UNIVERSIDAD TÉCNICA NACIONAL

VICERRECTORÍA DE INVESTIGACIÓN Y POSTGRADO
CENTRO DE FORMACIÓN PEDAGÓGICA Y TECNOLOGÍA EDUCATIVA

MAESTRÍA EN ENTORNOS VIRTUALES DE APRENDIZAJE

PROYECTO DE INTERVENCIÓN

TÍTULO DEL PROYECTO:

DISEÑO DEL CURSO DE PROGRAMACIÓN 1 PARA MODALIDAD VIRTUAL IMPLEMENTADO EN LA PLATAFORMA MOODLE

PREPARADO POR:

EDY DAVID ECHENIQUE RAMÍREZ

TUTOR DEL PROYECTO:

MARIELA DELAURO

AÑO

2019

ÍNDICE

RESUMEN TÉCNICO

Diseñar un curso para modalidad virtual representa todo un reto para los docentes y tutores, son muchos los aspectos que se deben tener presente para que el curso tenga éxito.

Este documento contiene una propuesta de diseño e implementación de un curso en modalidad virtual, el curso a desarrollar es de: “Programación 1” que se imparte actualmente en modalidad presencial como parte del programa de estudio académico de la carrera de ingeniería de sistemas computaciones en la Universidad Adventista de Centro América. El diseño de este documento está a cargo de un aspirante al postgrado de la maestría en entornos virtuales de aprendizaje en la Universidad Técnica Nacional de Costa Rica.

A través de este documento, compartiré con el lector la propuesta de diseño e implementación de un curso en modalidad virtual, que incluye una propuesta pedagógica, objetivos, resultados esperados, aspectos operativos y de administración, aspectos tecnológicos, evaluación y seguimiento del aprendizaje del estudiante, entre otros, todo esto tomando como marco de referencia el aprendizaje obtenido en el ciclo de especialización y maestría en entornos virtuales de aprendizaje. Por último, presento la planificación de una unidad que incluye tres clases con sus respectivas asignaciones y actividades, además, el diseño e implementación sobre el sistema Moodle como principal recurso de plataforma para la publicación del curso.

PROPUESTA DEL PROYECTO

1. EL PROBLEMA

1.1 El Problema

En la Universidad Adventista de Centro América (UNADECA) actualmente no se ofrecen programas formales de estudio en modalidad virtual. Sin embargo, existe una iniciativa de parte de cada director de facultad para pasar algunos de sus programas de educación formal presencial a una modalidad virtual a un mediano plazo. Uno de estos programas es el Bachillerato en Ingeniería de Sistemas Computacionales que se ofrece actualmente en la facultad de Ingeniería de Sistemas Computacionales. Este programa está aprobado por el Consejo Nacional de Enseñanza Superior Universitaria Privada (CONESUP) pero solo en modalidad presencial.

Un problema actual es que no existe un curso diseñado para impartirse en modalidad virtual que se pueda tomar como referencia en estructura para diseñar todos los cursos que componen el programa de Bachillerato en Ingeniería de Sistema Computacionales. Si se diseña un curso para modalidad virtual, se puede utilizar como punto de partida para diseñar todos los otros cursos que componen el programa de estudios. El curso propuesto en este documento es el de Programación 1 que se imparte en el segundo cuatrimestre del programa de estudio.

El diseño de un curso para modalidad virtual, aunque sea de una materia específica puede ser útil para el diseño de cursos virtuales en otros programas de estudio en diferentes facultadas ya que la estructura sería similar y sería una importante referencia.

1.2 Justificación

La UNADECA tiene como proyectos a mediano plazo cambiar de modalidad presencial a virtual algunos de sus programas de estudio en diferentes carreras, esto debido a que la UNADECA es un atractivo para muchos estudiantes adventistas en la región centroamericana y al ofrecer programas en modalidad virtual se estaría atendiendo esta demanda.

Ya hemos dicho que la UNADECA solo ofrece programas de estudio en modalidad presencial, sin embargo, hay mucho interés de parte de los estudiantes para

poder estudiar carreras en modalidad virtual. Muchos estudiantes desean venir y estudiar en la UNADECA, pero debido a las dificultades en temas de asuntos migratorios y costo de vida no lo pueden hacer, así que es de mucho interés de parte de la administración de la UNADECA abrir programas de estudio en modalidad virtual y atender esta demanda por parte de los estudiantes.

La UNADECA ya está haciendo esfuerzos importantes en capacitar a su personal docente en formación para trabajar en entornos virtuales de aprendizaje. Además, ya cuenta con una plataforma Open Source, en este caso es Moodle, que puede ser la herramienta ideal para el estudio en línea de estos programas de estudio en modalidad virtual.

Otra iniciativa de parte de los administradores de la universidad es ofrecer programas de estudio en educación continua, como cursos libres en diferentes áreas de interés y esto en modalidad virtual. De esta manera, la UNADECA estaría ampliando su oferta académica en educación formal y continua para una gran cantidad de futuros estudiantes que generalmente, son personas muy ocupadas y con falta de tiempo (trabajadores, personas que están en el exterior, o personas dedicadas al hogar) que no pueden asistir a un lugar de educación presencial; de esta manera, a través de la educación virtual es cómo pueden lograr tener un aprendizaje para desarrollarse profesionalmente.

1.3 Contexto del problema

El curso de Programación 1 solo existe en modalidad presencial actualmente, la idea es construir el curso para modalidad virtual desde una concepción pedagógica adaptada al tipo de estudiante. El curso se puede adaptar fácilmente en 12 unidades para cubrir los principales temas del contenido del curso y cada unidad puede durar 1 semana con su respectiva clase y sus actividades. Se estima que para la primera experiencia en impartir este curso tenga la participación de por lo menos 25 participantes. Con respecto al personal que pueden sostener el desarrollo de este curso, 2 personas con amplios conocimientos en la plataforma Moodle y un catedrático de la materia. Una persona será la encargada de desarrollar el contenido del curso junto a las actividades de aprendizaje, y la otra será el encargado de llevar el diseño

del curso a la plataforma Moodle.

La UNADECA, es una Universidad de larga trayectoria, más de 90 años de historia, actualmente tiene una oferta de 6 carreras: Administración de Empresas, Educación, Psicología, Enfermería, Teología e Ingeniería de Sistemas. Todos los docentes tienen el reto de formar profesionales para enfrentar los escenarios desafiantes de la sociedad actual, desarrollar en ellos competencias personales y profesionales que lo capacitan para: Ejercer liderazgo en proyectos de investigación, innovación y de servicio.

Por otro lado, está el contexto. Este es un mundo muy diferente al que hace unas pocas décadas; hay una revolución en el campo de las comunicaciones, el internet y toda el área de la informática hace que el mundo sea pequeño, sistemas políticos que parecían muy poderosos han caído, hay nuevos sistemas económicos que afectan toda la vida de la sociedad, la post modernización todas sus características y valores hacen que el hombre de hoy sea diferente. Sin duda este es un mundo de cambios vertiginosos y radicales en muchos aspectos que enfrenta al ser humano de hoy sea a situaciones muy difíciles. Por esta razón y otras, la UNADECA propone una oferta de programas de estudio formal y continuos en modalidad virtual.

La modalidad en línea. En esta modalidad, los estudiantes pueden tener una carga académica accesible que se ajuste a las posibilidades de tiempo. Además, organizar horarios de estudio, los salones virtuales están disponibles los 7 días de la semana, las 24 horas del día. Con una conexión a internet, ¡no hay límites!

De acuerdo a esto la UNADECA se ha propuesto estar a la vanguardia en el uso de recursos tecnológicos que permitan una educación más accesible, equitativa y justa en el contexto a nivel de instituciones de Educación Superior en nuestro país, capaz inclusive de trascender las fronteras que circunscriben actualmente los procesos educativos Universitarios tanto públicos como privados.

Una de las estrategias seleccionadas para tales fines es el uso de plataformas virtuales que apoyen el proceso enseñanza y aprendizaje y el cambio de modalidad presencial a virtual de algunas carreras.

2. PROSPECTIVA

En 6 meses de tiempo el curso de Programación 1 estará diseñado en un sitio virtual a través del sistema Moodle y se caracterizará de la siguiente manera: entorno amigable, material didáctico propio, mediado pedagógicamente y de acuerdo con lo que posibilita la web 2.0, con tiempo asignado para el seguimiento personalizado, evaluación del seguimiento de tal manera que se alcancen los objetivos propuestos.

El curso estará disponible desde su inicio quince semanas, y se le dará acceso a los estudiantes que van a tomar el curso. El contenido del curso será actualizado a medida que ocurran cambios significativos en el mismo cada vez que se vaya a ofrecer.

3. PROPUESTA PEDAGÓGICA

La propuesta pedagógica para este curso está basada en el constructivismo que proponen Vygotsky y Wenger, cuya característica principal de su teoría es el fomento de la reflexión en la experiencia, permitiendo que el contexto y el contenido sean dependientes de la construcción del conocimiento.

El curso de Programación 1 en modalidad virtual se diseñará bajo esta teoría ya que permite por un lado al docente ser un moderador, coordinador, facilitador y mediador y al mismo tiempo participativo a través de actividades en el proceso de aprendizaje. Estas actividades se pueden generar a través de múltiples herramientas Web 2.0. Por otro lado, el estudiante, siendo este el responsable de su propio proceso de aprendizaje y el procesador activo de la información es el que construye el conocimiento por sí mismo y con la utilización y aplicación de las TIC, potencia su compromiso en la participación, la interacción, la retroalimentación y conexión con el contexto real y estas son propicias para que el estudiante pueda controlar y ser conscientes de su propio proceso de aprendizaje.

4. OBJETIVOS

4.1 Generales

- Diseñar el curso de Programación 1 en modalidad virtual para el programa de Bachillerato en Ingeniería de Sistemas Computacionales y

que sirva como referencia para la creación de otros cursos en educación formal y continua.

Específicos

- Implementar el curso de Programación 1 en modalidad virtual utilizando la plataforma Moodle.
- Facilitar el acceso a los estudiantes al curso de Programación 1 a través de un entorno virtual por medio de la plataforma Moodle.
- Motivar a los directores de cada facultad para iniciar los procesos de crear programas de estudio en modalidad virtual.
- Seleccionar los recursos multimediales que se van a utilizar en cada una de las unidades de aprendizaje y enseñanza.

5. RESULTADOS ESPERADOS

De los resultados esperados en este proyecto podemos listar los siguientes:

- 1 curso de Programación 1 diseñado para ser impartido en modalidad virtual.
- 100% de estudiantes aprobados en este curso.
- 2 personas involucradas en el proceso del diseño y publicación de un curso en modalidad virtual
- El 90% de materiales multimediales necesarios creados y seleccionados para ser utilizados en el proceso de enseñanza y aprendizaje en el curso de Programación 1.
- 1 plataforma consolidada para la apertura de cursos en modalidad virtual.

6. ASPECTOS OPERATIVOS

6.1 Administración

En el proceso operativo de la administración del sistema estarán involucrados

los diferentes directores de facultades que son los que deberán conocer el proceso de diseño de un curso virtual. Por otro lado, es la directora académica en apoyo con la directora financiera de la universidad la que facilita los recursos necesarios para desarrollar el curso virtual.

Se debe contratar los servicios profesionales del tutor, del diseñador del curso y el espacio para el uso de la plataforma Moodle en un servidor que garantice la alta disponibilidad.

6.2 Aprendizaje y tecnologías

Al ser un curso en modalidad virtual debe desarrollarse a través de una plataforma que brinde las herramientas y que aporte en la construcción de contenidos, actividades de aprendizaje y recursos didácticos, en nuestro caso la propuesta es usar la plataforma Moodle.

Por ahora sugerimos una lista de prácticas de aprendizajes que acompañada de los contenidos será fundamental para la obtención del conocimiento:

Lecturas: no solo provistas por el docente en un formato de lectura adecuado sino a través de hipervínculos o enlaces hacia fuera de la plataforma.

Foros: espacios de intercambio de opiniones para debatir o enriquecer contenidos planteados por el tutor.

Laboratorios y ejercicios en software: facilitar al estudiante los softwares necesarios para que pueda realizar los diferentes laboratorios y prácticas relacionados con el contenido de cada unidad.

6.3 Tutoría

En cuanto a la tutoría considerando que no es lo mismo realizar una tutoría de una materia presencial a una virtual y que en esta propuesta hemos declarado que la primera experiencia al abrir este curso es para aproximadamente 15 personas. La persona encargada de la tutoría será la que ha tomado el curso de “La tutoría en entornos virtuales” en este mismo postgrado y que tiene un grado de Licenciatura en Ingeniería de Sistemas Computacionales, proponemos un tiempo de 10 a 15 horas semanales dedicados a la tutoría.

Es una gran responsabilidad para el tutor mantener la comunicación de cada participante desde principio a fin, pero si el tutor está capacitado para tal responsabilidad se pueden alcanzar los objetivos propuestos.

Otra tarea importante del tutor es la de redactar las clases y consignas para diferentes actividades, también habilitar las clases correspondientes a cada unidad, ser el moderador en cada foro manteniendo el rumbo correcto del debate planteado.

6.4 Materiales didácticos

En el marco de nuestro curso virtual propuesto, la producción del material didáctico no será una tarea fácil, pero se hará uso de muchos recursos, uno de ellos es el acceso a bibliografía digital accesible desde la base de datos ProQuest Ebook Central disponible en la siguiente dirección web: https://ebookcentral.proquest.com/lib/bvainteramericasp/home.action. El acceso a esta biblioteca digital es facilitado por el departamento de la biblioteca de la UNADECA que cuenta con una suscripción anual y es accesible para todos los estudiantes.

Otro recurso importante es la disponibilidad de software libre para acompañar el proceso de aprendizaje, algunos de estos softwares a utilizar son los siguientes: DIA Diagrama Editor disponible para instalar en Windows, Linux y Mac OS, otro programa es Draw disponible en internet a través del sitio web https://draw.io compatible con múltiples navegadores web.

También material referenciado producido en vídeo que servirá para reforzar los conceptos que se estudien en cada unidad. El material estará disponible para embeber en las clases ya que está almacenado en plataformas como YouTube, este tipo de vídeos es producido por diferentes autores especialistas en el área de programación.

A continuación, presentamos una lista detalladas de materiales didácticos que se desarrollarán y seleccionarán:

- Videos: para presentar información y reforzar contenidos de alguna temática.
- Guía didáctica: donde se presentarán los objetivos, contenidos, metodología de trabajo y modalidad de evaluación.
- Módulos: de cada una de las unidades. Constituirán la bibliografía básica y obligatoria de las mismas.

- Instructivos: de procedimientos que ayudarán a orientar a los estudiantes a realizar diferentes actividades dentro de la plataforma.

7. EVALUACIÓN Y SEGUIMIENTO DEL PROYECTO

7.1 Seguimiento

El proceso de seguimiento se llevará a cabo por medio de la plataforma, antes de iniciar el curso el tutor podrá visualizar la lista de sus estudiantes y conocer un poco acerca de ellos por medio de su perfil, durante el curso el seguimiento se hará mediante herramientas tradicionales, como trabajos prácticos, exámenes, evaluación de participación en actividades colectivas entre otras. Al finalizar el curso el seguimiento se hará mediante encuestas de retroalimentación respondidas por los participantes.

Cada proyecto se evalúa diferente, éste, al ser un proyecto relacionado con la educación virtual es difícil de evaluar, por ello categorizamos una serie de indicadores que marcarán las pautas a seguir para una evaluación objetiva:

7.2 Indicadores de evaluación

7.2.1 Modelo pedagógico general

1. Es adecuada la metodología usada por los facilitadores o tutores virtuales.
2. El diseño curricular cumple con los objetivos del proyecto.

7.2.2 Prácticas de aprendizaje y tecnologías

1. Fomenta el trabajo en equipo entre los participantes.
2. Facilita herramienta de software modernas y de libre distribución.

7.2.3 Material didáctico

1. Dispone de información actualizada relacionada con los temas a presentar.
2. Hay disponibilidad de materiales a nivel interno y externo.

3. El formato accesible para el estudiante de los materiales didácticos.
4. La calidad de los contenidos es aceptable.

7.2.4 Tutoría

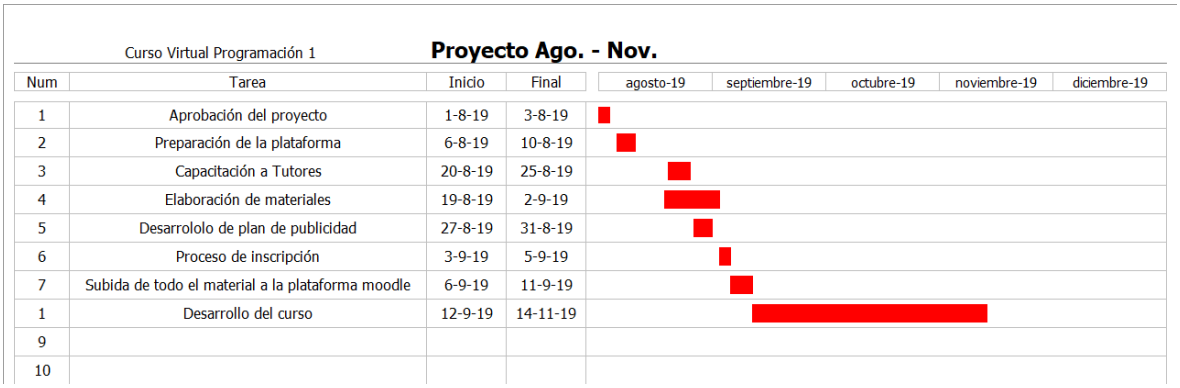
1. Dispone de una metodología implementada en la comunicación estudiante-tutor.
2. Hay compromiso de parte del tutor en atención a los estudiantes.
3. Las devoluciones son a tiempo de las actividades de los estudiantes.
4. Cómo califican los estudiantes al tutor a través de opiniones o encuestas.

7.2.5 Administración

1. El personal académico es adecuado para la educación a distancia.
2. Organización de reuniones de planificación con todos los involucrados en el proyecto.
3. Selección de personal como tutores con un perfil adecuado para ejercer esta función.
4. Cantidad de estudiantes proporcionan sostenibilidad al proyecto.

8. CRONOGRAMA PARA EJECUCIÓN DEL PROYECTO

El cronograma de actividades representa una buena planificación del proyecto estimando los tiempos adecuados para cada tarea, hay tareas que se pueden ejecutar simultáneamente para avanzar a buen ritmo con la preparación y desarrollo del proyecto. A continuación, presentamos un cuadro con las tareas y sus respectivas fechas para la ejecución.



Cuadro con de tareas

9. PRESUPUESTO

Todo proyecto de educación debe tener un presupuesto que facilite su desarrollo, y aunque un programa de formación virtual es más “reducido” que uno presencial hay que analizar claramente todas las variables involucradas. Para este proyecto se estima una inversión inicial más los costos de mantenimiento para sostenibilidad y funcionamiento, esto hay que contraponerlo con los ingresos que se van a generar para poder determinar las ganancias para la Universidad.

A continuación, presentamos una proyección de elementos que computan para un presupuesto y que se deben de tomar en cuenta para el inicio del proyecto y sostenibilidad del mismo.

9.1 Ingresos:

La proyección del número de estudiante que se estima que inicien el curso en modalidad virtual es de 15, y el costo para cada estudiante será de $150.00 USD.

15 estudiantes x $150.00 USD = $2,250 USD

9.2 Gastos:

Entre los gastos presentamos una lista de costos fijos y variables que se deben considerar:

9.2.1 Costos fijos:

- Instalaciones de oficina

- Equipo de cómputo
- Impresora
- Servicio de Internet
- Software para producción de material didáctico
- Hospedaje de la plataforma en un servidor arrendado

En costos fijos se presupuesta un monto de $300.00 USD por cohorte (1 bimestre)

9.2.2 Costos variables:

- Honorarios para la producción del material didáctico
- Honorarios para los tutores
- Costos administrativos

En costos fijos se presupuesta un monto de $350.00 USD (se toma en cuenta que tanto los productores de material didáctico como los tutores son de planta y lo que se presupuesta es un incentivo por su apoyo al proyecto)

Si a los ingresos ($2,250.00 USD) le restamos los costos fijos ($300.00 USD) y variables ($350.00 USD) resulta una utilidad de $1,600.00 para la universidad.

10. BIBLIOGRAFÍA

Acerca de Moodle - MoodleDocs. (2019). Recuperado el 12 de marzo de 2019, de https://docs.moodle.org/all/es/Acerca_de_Moodle

Bo, R., Sáez, A. y Belloch, C. (1999). Sistemas de evaluación de contenidos en teleformación: el proyecto CFV. Revista Electrónica de Investigación y Evaluación.

Bousaz, P. (2004). El constructivismo de Vigotsky. Buenos Aires: Lonseller.

Educativa. 5(2_2). Disponible en www.uv.es/RELIEVE/v5n2/RELIEVEv5n2_2.htm Consultado el 18 de marzo de 2019.

García, L. (2001). La educación a distancia. De la teoría a la práctica. Barcelona: Ariel. 287-305.

Prieto Castillo, D. (2017). Planificación, seguimiento y evaluación de proyectos, Aprende Virtual

DESARROLLO DEL PROYECTO

1. Nombre del curso virtual

Programación I

2. Selección y justificación de las herramientas tecnológicas

He seleccionado Moodle como plataforma de formación en línea por muchas razones. Primero, quiero mencionar que este software es de libre distribución y eso es una gran ventaja para iniciar a reducir costos en implementación de proyectos de formación en línea. Esta plataforma pone en manos de los docentes y tutores muchas de las herramientas necesarias para diseñar e implementar cursos virtuales profesionales.

En la Universidad Adventista de Centro América (UNADECA) ya se promueve el uso de esta plataforma como una herramienta de apoyo en la mayoría de los cursos presenciales y es casi un hecho de que esta plataforma será la oficial para ofrecer cursos 100% en línea a mediano plazo. La UNADECA ya cuenta con una instalación de Moodle en la siguiente dirección: https://unadeca.ac.cr/moodle/ y se capacita a los docentes en el uso de este sistema.

Al ser un curso en modalidad virtual debe desarrollarse a través de una plataforma que brinde las herramientas y que aporte en la construcción de contenidos, actividades de aprendizaje y recursos didácticos, en nuestro caso la propuesta es usar la plataforma Moodle.

Moodle ofrece distintos formatos para crear cursos virtuales donde cada uno de ellos delimitará la disposición del curso, el modo de exponer la información y toda la distribución de los contenidos y se debe seleccionar un formato que se adapte mejor para la interfaz que se desea proporcionar al curso.

Moodle le permite al docente utilizar el recurso página, donde puede agregar fácilmente texto, con diferente tipo de letra, color y tamaño, además, agregar elementos multimedia como audio y vídeo.

En Moodle tenemos a disposición múltiples herramientas de comunicación como mensajería interna, chat, foros, wikis entre otras. De manera nativa también hay herramientas para creación presentación de contenidos en diversos formatos, desde simples enlaces y páginas de texto hasta materiales con un mayor nivel de interactividad como son los cuestionarios, lecciones, Tareas, entre otros.

Moodle también dispone de una colección de utilidades para la configuración y edición de cursos. Podemos insertar, modificar y eliminar contenido dentro de un curso, adaptarlo a nuestras necesidades docentes o, simplemente modificar su aspecto visual. Otro aspecto importante a nivel pedagógico que nos ofrece esta plataforma es el seguimiento que se le puede dar a los estudiantes en todas las actividades de un curso individualmente o de manera colectiva, también incorpora la posibilidad de calificar las diversas tareas que se les asigne.

Moodle promueve el modelo de aprendizaje constructivista ya que incorpora como parte central de la experiencia formativa la realización de trabajos en grupo, participación en foros, discusiones, etc.

Para el diseño de este curso tengo a disposición de una instalación de Moodle en la versión 3.6 en un servidor propio alojado en la siguiente dirección: https://dominamoodle.net/proyectoeva de la cual yo soy el administrador, esto con el fin de garantizar el acceso total a la administración de esta plataforma y tener la libertad de agregar módulos externos según la necesidad del diseño del curso.

La estructura del aula virtual para el curso de Programación 1 se diseñará utilizando componentes y herramientas que ofrece la plataforma Moodle, a continuación, describimos los componentes que se van a utilizar:

Secciones del curso: Las secciones dentro de un curso en Moodle es donde se muestran los materiales de aprendizaje, a través de este elemento se acomodará el contenido por tópicos y cada tópico tendrá sus recursos y actividades.

En la primera sección o llamada también sección general, se utilizará el recurso libro para publicar la guía didáctica del curso, también en esta sección se colocará el recurso de novedades que ya viene por defecto en cada curso dentro de Moodle. En

esta sección también se habilitarán dos foros, uno para presentación personal de los estudiantes y otro para preguntas y respuestas generales del curso.

Recursos: Los recursos son elementos que permiten facilitar el acceso a materiales didácticos y serán utilizados en cada sección o tópico.
Entre los recursos que se utilizará están:

- Etiquetas: para colocar títulos o íconos.
- Archivos: para facilitar archivos como presentaciones, documentos pdf, imágenes entre otras.
- Página: para escribir las clases de cada unidad y agregar contenido multimedia como vídeos o archivos de sonido junto con texto explicativo.
- URL: para facilitar enlaces externos a sitios web, imágenes, documentos etc.

Actividades: Las actividades son las que los estudiantes harán y que interactúan con otros estudiantes y con el profesor o tutor. Entre las actividades más utilizadas serán:

- Tareas: para el envío y corrección de trabajos prácticos.
- Wikis: para realizar trabajos colaborativos.
- Foros: para tener discusiones asincrónicas durante un prolongado tiempo.
- Cuestionarios: para diseñar evaluaciones y exámenes con diferentes tipos de preguntas.

Bloques laterales: Otro elemento en la estructura del curso será el uso de bloques laterales, se habilitará el bloque de eventos próximos para mostrar un resumen a los estudiantes de lo que tienen que hacer según la fecha programada, el bloque de calendario para facilitar al estudiante la ubicación de las actividades en las fechas asignadas. El bloque de personas también estará disponible para facilitar el acceso a los participantes del curso.

3. Planificación de las clases

CLASE 1

Título de la clase 1: Instrucciones de selección

Objetivos de la clase:

1. Entender las técnicas básicas para la solución de problemas
2. Escribir algoritmos mediante un proceso secuencial usando pseudocódigo
3. Implementar las instrucciones if e if…else para decidir entre distintas acciones alternativas.

Contenidos:

- Introducción a las instrucciones de selección
- Algoritmos y Pseudocódigo
- El lenguaje del pseudocódigo
- Instrucciones if e if…else en Java

Bibliografía:

Echenique E. (2019). Unidad 3. Instrucciones de control en Java. Recuperado de: https://dominamoodle.net/proyectoeva/mod/resource/view.php?id=178

Deitel P. y H. (2016) Cómo programar en java. Editorial Pearson. Recuperado de http://jeisson.ecci.ucr.ac.cr/tmp/books/java/Java%20c%C3%B3mo%20programar,%2010ed%20-%20Deitel%20&%20Deitel%20%5B2016%5D.pdf

Vieties L. (2008). The Java Tutorials. Recuperado de http://www.codexion.com/tutorialesjava/java/

Recursos multimedia

Foto del profesor, al inicio de cada clase. URL: https://photos.app.goo.gl/NHREkGLd6manjr7m6

Imagen de la firma del profesor. URL: https://photos.app.goo.gl/Hd7fK5Y42rCnizjXA

Imagen: Diagrama de actividad para la instrucción if. URL: https://photos.app.goo.gl/ToiMzVhPFsCWG7dJ8

Imagen: Diagrama de actividad para la instrucción if…else URL: https://photos.app.goo.gl/APG7yk69uWdTjNHA6

Curso de java: Sentencias de control 1. Vídeo que explica el uso de las instrucciones de selección (if, else…if y switch) creado por: Códigos de Programación disponible en la siguiente dirección https://www.youtube.com/playlist?list=PL-Mlm_HYjCo9vojnWbPTs6nV51J8WV_O5

Instrucciones de selección en Java: Edy Echenique. Disponible desde la siguiente dirección: https://es.slideshare.net/EdyEchenique/echenique-objeto-digital

Actividades:

Consigna asignación 1:

Escriba un algoritmo con pseudocódigo que determine si alguno de los clientes de una tienda de departamentos se ha excedido del límite de crédito en una cuenta. Para cada cliente se tienen los siguientes datos:

a. El número de cuenta.

b. El saldo al inicio del mes.

c. El total de todos los artículos cargados por el cliente en el mes.

d. El total de todos los créditos aplicados a la cuenta del cliente en el mes.

e. El límite de crédito permitido.

En el algoritmo se debe especificar como datos de entrada cada uno de los ítems listados anteriormente en forma de números enteros, su algoritmo debe calcular el nuevo saldo de la siguiente manera:

nuevo saldo = (saldo inicial + cargos - créditos)

después de mostrar el nuevo saldo debe determinar si este excede el límite del crédito del cliente. Para los clientes cuyo límite de crédito sea excedido se debe mostrar un mensaje que diga: “se excedió el límite de crédito” en caso contrario debe mostrar un mensaje que diga: “Su límite de crédito no se ha excedido”.

Luego de escribir su algoritmo con pseudocódigo realice los siguiente:

- El programa en lenguaje Java con su IDE (entorno de desarrollo integrado) de preferencia.
- El diagrama de actividad

Objetivo de la actividad:

- Desarrollar un programa en el lenguaje Java, aplicando las técnicas de programación para el manejo de estructuras de control dentro de un programa.

Evaluación:

Para valorar a conciencia el esfuerzo y tiempo del estudiante empleado en la realización de su trabajo, se utilizará los siguientes criterios.

- Utilización de símbolos adecuados para la elaboración del diagrama de actividad
- Funcionalidad del programa
- Aplicación del razonamiento lógico
- Estructura del código
- Documentación del código

Plazos de entrega:

Esta actividad deberá ser entregada en dos formatos; el algoritmo con el pseudocódigo y el diagrama de actividad deberá entregarlo en un documento con formato .doc o .pdf

El programa en Java con todos los archivos de código fuente debe ser entregado en una carpeta comprimida con formato .zip

Se dispone de un máximo de 7 días para realizar este trabajo a partir de la fecha de publicación del mismo.

Foro

Consigna:

Este foro corresponde a la clase 1 de la unidad 3, sabemos que existen dos estructuras de selección, if simple e if…else de selección doble, ambas se pueden utilizar para dirigir el flujo de acciones a ejecutar dentro de un programa.

Compare y contraste la instrucción if de selección simple y la instrucción if…else de selección doble. Responda a las siguientes preguntas ¿Cuál es la similitud en las dos instrucciones? ¿Cuál es su diferencia? ¿En cuáles casos puedo o no utilizar una de las dos? Justifique su respuesta.

Primero debe hacer su aporte de acuerdo a la consigna de este foro, segundo debatir colaborativamente aportando sus acuerdos o desacuerdos y tercero contar su experiencia si ya ha implementados alguna de estas dos instrucciones en sus proyectos de programación.

Objetivo del foro

Promover la participación colaborativa entre estudiantes para la solución de situaciones del mundo real.

Plazo de participación en el foro

Para poder tener tiempo suficiente en realizar sus aportes y participar colaborativamente con sus compañeros este foro estará abierto durante 7 días a partir de la fecha que sea publicado.

CLASE 2

Título de la clase 2: Instrucciones de repetición

Objetivos de la clase:

1. Entender los fundamentos de los ciclos controlados por un contador
2. Implementar las instrucciones de repetición for, while y do…while en un programa
3. Resolver problemas del mundo real aplicando estructuras de repetición.

Contenidos:

- Introducción a las instrucciones de repetición
- Fundamentos de los ciclos controlados por un contador
- La instrucción for
- La instrucción while
- La instrucción do…while

Bibliografía:

Echenique E. (2019). Unidad 3. Instrucciones de control en Java. Recuperado de: https://dominamoodle.net/proyectoeva/mod/resource/view.php?id=178

Deitel P. y H. (2016) Cómo programar en java. Editorial Pearson. Recuperado de http://jeisson.ecci.ucr.ac.cr/tmp/books/java/Java%20c%C3%B3mo%20programar,%2010ed%20-%20Deitel%20&%20Deitel%20%5B2016%5D.pdf

Vieties L. (2008). The Java Tutorials. Recuperado de http://www.codexion.com/tutorialesjava/java/

Recursos multimedia

Foto del profesor, al inicio de cada clase. URL: https://photos.app.goo.gl/NHREkGLd6manjr7m6

Imagen de la firma del profesor. URL: https://photos.app.goo.gl/Hd7fK5Y42rCnizjXA

Curso de java: Sentencias de control 2. Vídeo que explica el uso de las instrucciones de repetición (while, do…wile, for) por: Códigos de Programación disponible en la siguiente dirección https://www.youtube.com/playlist?list=PL-Mlm_HYjCo9vojnWbPTs6nV51J8WV_O5

Instrucciones de control parte 2: Edy Echenique. Disponible desde la siguiente dirección: https://www.slideshare.net/secret/4Y7vaf0xB1zA5P

Actividades:

Consigna asignación 2:

(Programa para imprimir gráficos de barra) Una aplicación interesante de las computadoras es la visualización de gráficos convencionales y de barra. Aplique el

razonamiento lógico para resolver la siguiente situación: Escriba un programa en el lenguaje Java que lea cinco números, cada uno entre 1 y 30. Por cada número leído, su programa debe mostrar el mismo número de asteriscos adyacentes. Por ejemplo, si su programa lee el número 7, debe mostrar *******. Muestre las barras de asteriscos después de leer los cinco números.

Objetivo de la actividad:

- Desarrollar un programa en el lenguaje Java, aplicando la estructura de repetición for para resolver una situación específica.

Evaluación:

Para valorar a conciencia el esfuerzo y tiempo del estudiante empleado en la realización de su trabajo, se utilizará los siguientes criterios.

- Funcionalidad del programa
- Aplicación del razonamiento lógico
- Estructura del código
- Documentación del código

Plazos de entrega:

El programa en Java con todos los archivos de código fuente debe ser entregado en una carpeta comprimida con formato .zip

Se dispone de un máximo de 7 días para realizar este trabajo a partir de la fecha de publicación del mismo.

Foro

Consigna:

Este foro corresponde a la clase 2 de la unidad 3, según lo visto en la clase, entendemos que el lenguaje Java cuenta con dos estructuras de repetición muy similares, una es while y la otra do…while.

Responda a las siguientes preguntas ¿Cuál es la similitud en las dos instrucciones? ¿Cuál es su diferencia? ¿En cuáles casos puedo o no utilizar una de las dos? Justifique su respuesta.

Primero debe hacer su aporte de acuerdo a la consigna de este foro, segundo debatir colaborativamente aportando sus acuerdos o desacuerdos y tercero contar su experiencia si ya ha implementados alguna de estas dos instrucciones en sus proyectos de programación.

Objetivo del foro

Promover la participación colaborativa entre estudiantes para la solución de situaciones del mundo real.

Plazo de participación en el foro

Para poder tener tiempo suficiente en realizar sus aportes y participar colaborativamente con sus compañeros este foro estará abierto durante 7 días a partir de la fecha que sea publicado.

CLASE 3

Título de la clase 3: Instrucción de selección múltiple switch e instrucciones de control break

Objetivos de la clase:

1. Entender la selección múltiple implementando la instrucción switch
2. Utilizar instrucciones de control de un programa para alterar su flujo mediante la palabra break

Contenidos:

- Introducción a la instrucción switch
- Aplicación de la instrucción de selección múltiple switch
- Instrucciones break y continue

Bibliografía:

Echenique E. (2019). Unidad 3. Instrucciones de control en Java. Recuperado de: https://dominamoodle.net/proyectoeva/mod/resource/view.php?id=178

Deitel P. y H. (2016) Cómo programar en java. Editorial Pearson. Recuperado de http://jeisson.ecci.ucr.ac.cr/tmp/books/java/Java%20c%C3%B3mo%20programar,%2010ed%20-%20Deitel%20&%20Deitel%20%5B2016%5D.pdf

Vieties L. (2008). The Java Tutorials. Recuperado de http://www.codexion.com/tutorialesjava/java/

Recursos multimedia

Foto del profesor, al inicio de cada clase. URL: https://photos.app.goo.gl/NHREkGLd6manjr7m6

Imagen de la firma del profesor. URL: https://photos.app.goo.gl/Hd7fK5Y42rCnizjXA

Curso de programación en java: La sentencia Switch. Vídeo que explica el uso de las instrucciones de selección múltiple (switch) creado por: Programación ATS disponible en la siguiente dirección https://www.youtube.com/watch?v=T4X0C5bLM0A

Tutorial java: Las instrucciones Break y Continue. Vídeo que explica el uso de las instrucciones de control de flujo de programa (break y continue) creado por: Código Facilito disponible en la siguiente dirección https://www.youtube.com/watch?v=C1W9nfdYlnI

Actividades:

Consigna asignación 3:

Planteamiento del problema

Un almacén de pedidos por correo vende cinco productos cuyos precios de venta son los siguientes, producto 1: $2.98, producto 2: $4.50, producto 3: $9.98, producto 4: $4.99, producto 5: $6.87. Escriba un programa en Java que lea una serie de pares de números, como se muestra a continuación:

a. Número de producto

b. Cantidad vendida

Su programa debe utilizar una instrucción switch para determinar el precio de venta de cada producto debe calcular y mostrar el valor total de venta de todos los productos vendidos. Use un ciclo controlado por centinela para determinar cuándo debe dejar de iterar para mostrar los resultados finales.

Objetivo de la actividad:

- Desarrollar un programa en el lenguaje Java, aplicando las técnicas de programación para el manejo de instrucción de selección múltiple switch.

Evaluación:

Para valorar a conciencia el esfuerzo y tiempo del estudiante empleado en la realización de su trabajo, se utilizará los siguientes criterios.

- Funcionalidad del programa
- Aplicación del razonamiento lógico
- Estructura del código
- Documentación del código

Plazos de entrega:

El programa en Java con todos los archivos de código fuente debe ser entregado en una carpeta comprimida con formato .zip

Se dispone de un máximo de 7 días para realizar este trabajo a partir de la fecha de publicación del mismo.

Foro

Consigna:

Este foro corresponde a la clase 3 de la unidad 3, Responda a las siguientes preguntas ¿Qué sucede cuando la instrucción break se ejecuta dentro de una instrucción while, for, do…while o switch? ¿Cuál es la función principal de la instrucción continue? Justifique su respuesta.

Primero debe hacer su aporte de acuerdo a la consigna de este foro, segundo debatir colaborativamente aportando sus acuerdos o desacuerdos y tercero contar su experiencia si ya ha implementados alguna de estas dos instrucciones en sus proyectos de programación.

Objetivo del foro

Promover la participación colaborativa entre estudiantes para comprender el uso correcto de las instrucciones break y continue.

Plazo de participación en el foro

Para poder tener tiempo suficiente en realizar sus aportes y participar colaborativamente con sus compañeros este foro estará abierto durante 7 días a partir de la fecha que sea publicado.

4. Redacción de las clases

Clase 1 Instrucciones de selección

Edy Echenique

Bienvenidos(as) a la primera clase virtual de la unidad 3, es importante leer toda la clase y, además, leer y consultar las lecturas obligatorias y opcionales para enriquecer más el contenido de nuestra clase.
En esta unidad habilitaré 3 clases, una por semana los días miércoles, así que, es muy importante que ingresen una vez por semana a la plataforma para leer y revisar las clases publicadas.
Es muy importante que lea bien las consignas de la asignación y el foro para que pueda realizar sus actividades cumpliendo con lo solicitado.
Los materiales de lectura están disponibles en formato PDF desde la sección de recursos para la unidad 3 (página principal del curso).

Comencemos con nuestra primera clase:

Instrucciones de selección

Ya hemos recorrido parte de nuestro camino de este curso: **Introducción a la programación con JAVA**, hemos ido aprendiendo como crear programas básicos para computadoras utilizando el lenguaje de programación Java, aun nos falta mucho que aprender pues este lenguaje es extenso y requiere muchos meses de práctica, llegamos a la clase No. 1 de la unidad 3, vamos a dedicar esta clase a comprender las técnicas generales para la solución de problemas y aprender a cómo podemos utilizar las instrucciones de selección **`if`** e **`if…else`** para tomar decisiones entre diferentes alternativas dentro de un programa.

> "*Antes de escribir un programa que dé solución a un problema, es imprescindible tener una comprensión detallada de todo el problema, además de una metodología cuidadosamente planeada para resolverlo. Al escribir un programa, es igualmente esencial comprender los tipos de bloques de construcción disponibles, y emplear las técnicas comprobadas para construir programas.*" **(2012)**, **Paul y Harvey Deitel, Java: Cómo programar**

Algoritmos y Pseudocódigo

Antes de conocer y poner en práctica el uso de las instrucciones **`if`** e **`if…else`**, vamos a repasar algunos conceptos de **algoritmos y pseudocódigo**. Un problema de computadora puede resolverse según las instrucciones ejecutadas y el orden en que éstas se ejecutan. Nosotros los humanos cada día implementamos algoritmos para resolver situaciones fáciles y complejas del mundo que nos rodea, por ejemplo, considere el "**algoritmo para pintar una habitación**":

(1) preparar los materiales para pintar; **(2)** lavar la superficie a pintar; **(3)** preparar la pintura a utilizar; **(4)** pintar las paredes; **(5)** esperar a que seque la pintura; **(6)** limpiar manchas no deseas de pintura en el piso. Esta rutina o secuencia de pasos hará que una habitación quede bien pintada.

Ahora, suponga que estos pasos se lleven a cabo en un orden ligeramente distinto:

(1) preparar los materiales para pintar; **(2)** pintar las paredes; **(3)** preparar la pintura a utilizar; **(4)** lavar la superficie a pintar; **(5)** esperar a que seque la pintura; **(6)** limpiar manchas no deseas de pintura en el piso, en este caso seguramente las paredes de la habitación no quedarían pintadas.

Detallar un orden en que se ejecutan una serie de acciones en un programa se le conoce como: **control de ejecución del programa**. Las instrucciones **`if`** e **`if…else`** nos permiten aplicar un control a nuestros programas, lo vamos a mostrar en un momento.

El lenguaje del pseudocódigo

El **pseudocódigo** se conoce como un lenguaje informal y permite construir algoritmos sin tener que preocuparse por todos los detalles de la sintaxis del lenguaje de programación Java. El pseudocódigo es como el lenguaje humano, pero no es un lenguaje de programación de computadoras, ya veremos un ejemplo de pseudocódigo.

Instrucciones `if e if…else` en java

En el **lenguaje Java** vamos a encontrar tres tipos de instrucciones de selección de las cuales vamos a estudiar 2 (**`if`** e **`if…else`**). La instrucción **`if`** ejecuta una o varias acciones si el resultado de la condición es verdadera, o salta la ejecución si el resultado de la condición es falsa. La instrucción **`if…else`** a diferencia de la anterior ésta va a ejecutar una o varias instrucciones si el resultado de la condición es verdadera o va a ejecutar otra serie de instrucciones si el resultado de la condición es falsa. La instrucción se selección **`if`** se le conoce como: **instrucción de selección simple** y a la instrucción **`if…else`** se le conoce como **instrucción de selección doble**.

Instrucción `if`

Recuerda que los programas usan las instrucciones **if** para elegir entre ejecutar un conjunto de acciones o no ejecutarlas. Veamos un ejemplo:

Supongamos que la edad de una persona para ser mayor de edad es de 18 años, este ejemplo lo podemos expresar en pseudocódigo de la siguiente manera:

Si la edad de una persona es mayor o igual que 18 años

Imprimir "Es mayor de edad"

Estamos utilizando un lenguaje natural para expresar esta situación, note que hay una condición que se debe determinar si es verdadera y en caso de que así lo sea se va a imprimir en pantalla "**Es mayor de edad**" el programa debería continuar en orden a la siguiente instrucción en el pseudocódigo. En caso de que la condición sea falsa se estaría **omitiendo** la instrucción Imprimir y continuaría en orden la siguiente instrucción en el pseudocódigo.

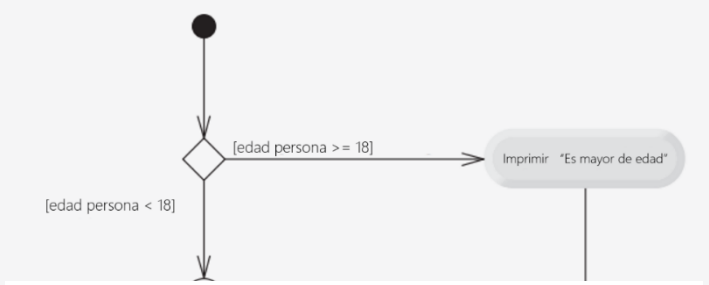


Diagrama de actividad para la instrucción ***if***

La instrucción ***Si*** que acabamos de ver a través de un pseudocódigo se escribiría en el lenguaje Java de la siguiente manera:

```
if (edadPersona >= 18)
      System.out.println("Es mayor de edad")
```

Observe que el código Java es equivalente en gran medida con el pseudocódigo, por eso es que el pseudocódigo es una herramienta muy útil para el desarrollo de programas, ayuda que puedas especificar los pasos de un algoritmo en lenguaje natural y luego esa misma estructura se adapta a un lenguaje de programación.

El siguiente **diagrama de actividad** muestra la instrucción **if** simple del ejemplo que estamos utilizando. Este diagrama contiene un símbolo que es muy importante en los diagramas de actividad como lo es el rombo el cual representa que se tomará una decisión.

En el diagrama si la edad de la persona es mayor o igual a 18 años, el programa imprimirá: "**Es mayor de edad**" y luego se dirige al estado final de la actividad. Si la edad

de la persona es menor que 18 años, el programa se dirige de inmediato al estado final sin mostrar un mensaje.

"Los diagramas de actividad forman parte de UML", **(2012)**, **Paul y Harvey Deitel**, los describen de la siguiente manera:

- *"Un diagrama de actividad modela el flujo de trabajo (también conocido como la actividad) de una parte de un sistema de software.*
- *Los diagramas de actividad se componen de símbolos de propósito especial, como los símbolos de estados de acción, rombos y pequeños círculos. Estos símbolos se conectan mediante flechas de transición, las cuales representan el flujo de la actividad.*
- *Los estados de acción representan las acciones a realizar. Cada estado de acción contiene una expresión de acción, la cual especifica una acción específica a realizar.*
- *Las flechas en un diagrama de actividad representan las transiciones, que indican el orden en el que ocurren las acciones representadas por los estados de acción.*
- *El círculo relleno que se encuentra en la parte superior de un diagrama de actividad representa el estado inicial de la actividad: el comienzo del flujo de trabajo antes de que el programa realice las acciones modeladas.*
- *El círculo sólido rodeado por una circunferencia, que aparece en la parte inferior del diagrama, representa el estado final: el término del flujo de trabajo después de que el programa realiza sus acciones.*
- *Los rectángulos con las esquinas superiores derechas dobladas se llaman notas en UML: comentarios que describen el propósito de los símbolos en el diagrama."*

Instrucción `if…else`

Esta instrucción permite especificar una o varias acciones para ejecutar cuando la condición es verdadera, y otras cuando el resultado de la condición es falsa. Veamos una variante del ejemplo anterior.

```
Si la edad de una persona es mayor o igual que 18 años
        Imprimir "Es mayor de edad"
De lo contrario
        Imprimir "Es menor de edad"
```

Observe que la instrucción que se va a ejecutar si la condición es verdadera será: "**Es mayor de edad**" y "**Es menor de edad**" si la edad de la persona es menor que 18 años.

La instrucción ***Si…de lo contrario*** que acabamos de ver a través de un pseudocódigo se escribiría en el lenguaje Java de la siguiente manera:

```
if (edadPersona >= 18)
      System.out.println(“Es mayor de edad”)
else
      System.out.println(“Es menor de edad”)
```

Un aspecto importante que debemos tener siempre en cuenta es el de usar sangría ya que además de permitir una mejor comprensión y visibilidad del código enfatiza la estructura inherente de todos los programas que son estructurados.

Buena práctica de programación.

Recuerde utilizar siempre sangría en ambos cuerpos de instrucciones de una estructura ***`if...else`***

Ahora veamos el diagrama de actividad aplicando una instrucción **`if…else`**, los símbolos en el diagrama de actividad representan estados de ejecución y decisión.

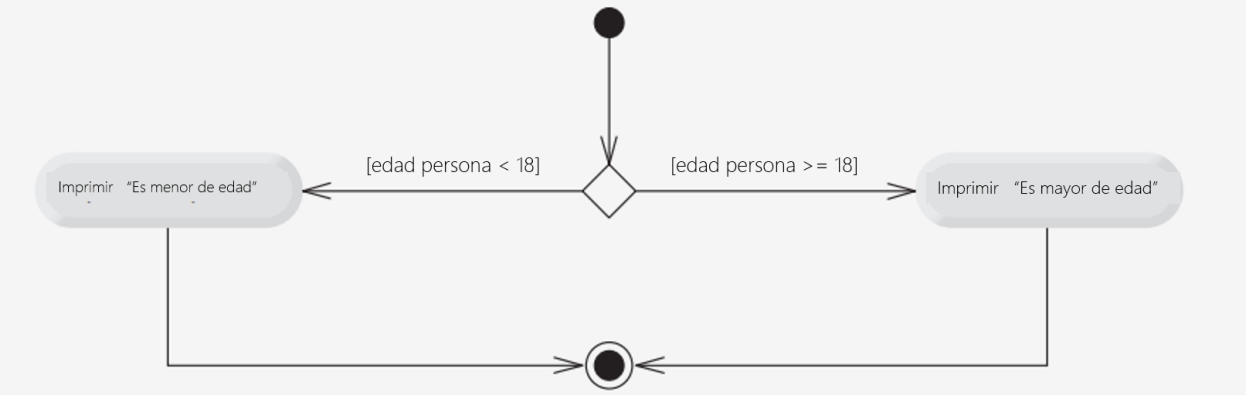


Diagrama de actividad para la instrucción if…else

A continuación, comparto una presentación que contiene un resumen de las estructuras de selección y algunos ejemplos:

Esta presentación será incrustada en la plataforma

URL: https://es.slideshare.net/EdyEchenique/echenique-objeto-digital.

Para finalizar te invito a ver el siguiente vídeo: Curso de java: Sentencias de control 1, que explica el uso de las instrucciones de selección (`if, else…if`) creado por: Códigos de Programación disponible en la siguiente dirección**:**

(Este vídeo será incrustado en la plataforma)

https://www.youtube.com/playlist?list=PL-Mlm_HYjCo9vojnWbPTs6nV51J8WV_O5

Conclusión

He demostrado el uso de la instrucción de selección simple **if** y la instrucción de selección doble **if…else** acompañado a esto, ahora sabemos que el procedimiento para resolver un problema en acciones a ejecutar y el orden en que éstas se ejecutan se conoce como **algoritmo** y para representar un algoritmo en un lenguaje natural se utiliza el **pseudocódigo** sin tener que preocuparse por los detalles estrictos de la sintaxis del lenguaje.

El uso de **diagramas de actividad** modela el flujo de trabajo de una parte de un programa de computadora. Cada diagrama de actividad se compone de símbolos con propósito especial y todos estos símbolos están conectados mediante flechas de transición y que representan los flujos de actividades.

Actividades

- Descargar y leer Unidad 3: Instrucciones de control (https://dominamoodle.net/proyectoeva/mod/resource/view.php?id=178)

- Ir y leer tutorial de Java en la sección de instrucciones de control: (http://www.codexion.com/tutorialesjava/java/)

- Vamos a pasar de la teoría a la práctica, y para esto, usted deberá resolver un problema para computadora utilizando los elementos que acabamos de estudiar, a saber, algoritmo con pseudocódigo, programa en Java y diagrama de actividad.

El planteamiento del problema es el siguiente:

Escriba un algoritmo con pseudocódigo que determine si alguno de los clientes de una tienda de departamentos se ha excedido del límite de crédito en una cuenta. Para cada cliente se tienen los siguientes datos:

a. *El número de cuenta.*
b. *El saldo al inicio del mes.*
c. *El total de todos los artículos cargados por el cliente en el mes.*
d. *El total de todos los créditos aplicados a la cuenta del cliente en el mes.*
e. *El límite de crédito permitido.*

En el algoritmo se debe especificar como datos de entrada cada uno de los ítems listados anteriormente en forma de números enteros, su algoritmo debe calcular el nuevo saldo de la siguiente manera:

`nuevo saldo = (saldo inicial + cargos - créditos)`

después de mostrar el nuevo saldo debe determinar si este excede el límite del crédito del cliente. Para los clientes cuyo límite de crédito sea excedido se debe mostrar un mensaje que diga: "***se excedió el límite de crédito***" en caso contrario debe mostrar un mensaje que diga: "***Su límite de crédito no se ha excedido***".

Luego de escribir su algoritmo con pseudocódigo realice los siguiente:

- ✓ El programa en lenguaje Java con su IDE (entorno de desarrollo integrado) de preferencia.
- ✓ El diagrama de actividad

Formato de entrega:

Esta actividad deberá ser entregada en dos formatos; el algoritmo con el pseudocódigo y el diagrama de actividad deberá entregarlo en un documento con formato **.doc o .pdf**

El programa en Java con todos los archivos de código fuente debe ser entregado en una carpeta comprimida con formato **.zip y debe establecer el nombre de la carpeta comprimida a Apellido_Nombre_Asignaciones.zip**

El plazo de entrega es el 15/06/19, y su entrega debe ser a través de la actividad tareas que se encuentra en la sección de Asignaciones (página principal del curso en moodle)

FORO

Saludos cordiales a tod@s,

Este foro corresponde a la clase 1 de la unidad 3, sabemos que existen dos estructuras de selección, if simple e if…else de selección doble, ambas se pueden utilizar para dirigir el flujo de acciones a ejecutar dentro de un programa.

Antes de poder participar en este foro es necesario realizar la actividad de resolución de problemas asignada al final de la clase 1 de esta unidad.

*Compare y contraste la instrucción **if** de selección simple y la instrucción **if…else** de selección doble. ¿Cuál es la similitud en las dos instrucciones? ¿Cuál es su diferencia? ¿En cuáles casos puedo o no utilizar una de las dos? Justifique su respuesta.*

Primero debe hacer su aporte personal de acuerdo a la consigna de este foro, segundo debatir colaborativamente aportando sus acuerdos o desacuerdos y tercero contar su experiencia si ya ha implementados alguna de estas dos instrucciones en sus proyectos de programación.

Recomendaciones:

- Favor no esperar hasta último día para participar, así puede intercambiar con sus compañeros.
- Responder de una manera clara, concisa y puntual a lo solicitado en la consigna.
- Revisar lo escrito antes de publicarlo, sobre todo la redacción y la ortografía.
- Publicar la opinión directamente en el foro y no a través de archivos adjuntos.

Este foro estará disponible en la sección Actividades de la unidad 3 (página principal del curso en Moodle)

Plazo de participación en el foro

Para poder tener tiempo suficiente en realizar sus aportes y participar colaborativamente con sus compañeros este foro estará abierto durante 7 días a partir de la fecha que sea publicado.

-Edy Echenique

Clase 2: Instrucciones de repetición

Edy Echenique

Bienvenidos(as) a la segunda clase virtual de la unidad 3, es importante leer toda la clase y, además, leer y consultar las lecturas obligatorias y opcionales para enriquecer más el contenido de nuestra clase.

En esta clase continuaremos estudiando las estructuras de control, y toca el turno para las estructuras repetitivas.

Les recuerdo que lean bien las consignas de la asignación y el foro en esta clase para que pueda realizar sus actividades cumpliendo con lo solicitado.

Los materiales de lectura están disponibles en formato PDF desde la sección de recursos para la unidad 3 (página principal del curso).

Comencemos con nuestra segunda clase:

Instrucciones de repetición

En el lenguaje Java podemos utilizar tres instrucciones de repetición, que permiten a los programas ejecutar bloques de código de manera repetida dependiendo de la condición. Estas instrucciones son las siguientes:

While, do while y for

Instrucción de repetición `while`

En una instrucción de repetición `while` un programa puede repetir una acción mientras la condición sea verdadera y se detendrá hasta que la condición sea falsa.

Veamos un ejemplo de esta instrucción de repetición en Java. El siguiente ejemplo es de un programa que encuentra la primera potencia de 3 que sea mayor a 60 asumiendo que la variable producto de tipo `int` se inicia en 3.

```
while (producto <= 80)
      producto = 3 * producto;
```

Vamos a explicar este ejemplo, en cada repetición de la instrucción **`while`** se multiplicara a la variable producto por **`3`**, por lo que la variable producto tomara los valores de **`9`**, **`27`** y **`81`** sucesivamente, note que cuando la variable producto toma el valor de **`81`** la condición que mantiene la ejecución del ciclo (**`producto <= 80`**) dará como resultado el valor de falso y entonces se detendrá la repetición, así que el valor final de la variable producto será de **`81`**. Es en este punto donde la ejecución del programa continuará con la siguiente instrucción después de la instrucción **`while`**.

Normalmente esta instrucción se utiliza en las situaciones en donde no se conoce la cantidad de repeticiones que se deben ejecutar para producir un resultado.

Repetición controlada por un contador

En ocasiones necesitaremos ejecutar un ciclo controlando las veces que queremos ejecutarlo, para esto es necesario el uso de un contador de lo cual vamos a hablar en este momento.

Existen varios elementos que son requeridos para llevar a cabo una repetición controlada por un contador:

1. Variable de control (contador de ciclo)
2. Valor inicial de la variable de control
3. Incremento con el que se modifica la variable de control cada vez que pasa por el ciclo.

4. Condición de continuación del ciclo. (determina si el ciclo debe continuar)

Para comprender estos elementos vamos a ver un ejemplo en código:

```
1 public class ContadorWhile
2 {
3     public static void main(String[] args) {
4           int contador = 1; // declara e inicializa la variable de control
5           while (contador <= 10) // condición de continuación de ciclo
6           {
7                 System.out.printf("%d ", contador);
8                 ++contador; // incrementa la variable de control
9           }
10          System.out.println();
11    }
12}
```

La salida de este programa sería:

```
1 2 3 4 5 6 7 8 9 10
```

En el código del ejemplo anterior todos los elementos de la repetición controlada por un contador se definen en la línea 4, 5 y 8, en la línea 4 declara la variable de control contador como un **`int`** y se establece el valor inicial en 1.

En la línea 13 incrementa la variable de control en 1, por cada repetición en el ciclo, observe que la condición de continuidad del ciclo en la instrucción **`while`** (línea 5) evalúa si el valor de la variable de control es menor o igual que 10 que sería el valor final para que la condición sea verdadera. Note que el cuerpo del **`while`** se ejecuta aun cuando la variable de control es igual a 10. Solo cuando el valor de la variable es mayor que 10 el ciclo termina o cuando el contador se convierte en 11.

Instrucción de repetición `do while`

Esta instrucción es similar a la instrucción **`while`**. Recordemos que la instrucción **`while`** el programa evalúa la condición de continuación de ciclo al principio, antes de ejecutar el cuerpo del ciclo; si la condición es falsa, el cuerpo nunca se va a ejecutar. En el caso de la instrucción do **`while`** evalúa la condición de repetición del ciclo después de ejecutar el cuerpo del ciclo; por lo tanto, el cuerpo siempre se va a ejecutar por lo menos 1 vez.

Veamos un ejemplo donde un programa implementa una instrucción **do while** para imprimir los números del 1 al 10:

```
1 public class PruebaDoWhile
2 {
3     public static void main(String[] args) {
4           int contador = 1;
5           do
6           {
7                 System.out.printf(“%d ”, contador);
8                 ++contador;
9           } while (contador <= 10)
10          System.out.println();
11    }
12}
```

La salida de este programa sería:

```
1 2 3 4 5 6 7 8 9 10
```

Cuál es la diferencia entonces entre la instrucción `while` y `do while`:

Observe que en la línea 4 se declara e inicializa la variable de control contador. Cuando se ejecuta el cuerpo de la instrucción do **while** la línea 7 imprime el valor del contador y la 8 incrementa el contador. Note que después el programa evalúa la prueba de continuación del ciclo al final de este en la línea 9. Si la condición es verdadera, el ciclo continúa repitiéndose, pero si la condición es falsa el ciclo termina el programa y sigue a la siguiente instrucción después del ciclo.

Instrucción de repetición `For`

Hasta aquí hemos visto dos instrucciones donde no conocíamos la cantidad de ciclos que se ejecutaban hasta cumplir con una condición. Ahora vamos a ver cómo utilizar una instrucción donde ya se sabe cuántos ciclos se deben cumplir para terminar la ejecución.

Imaginemos que queremos imprimir la tabla de multiplicar de un número hasta el factor 10. Sin utilizar ninguna instrucción de repetición tendríamos que escribir 9 líneas de código de la siguiente manera:

```
System.out.println("3 x 1 = 3");
```

.
.
.
System.out.println("3 x 10 = 27");

Para realizar estas operaciones de una manera más rápida podríamos utilizar unas de las instrucciones que ya hemos visto antes, pero ahora toca el turno a la instrucción `For`

Observe el siguiente código:

```
for ( int factor = 1; factor <= 9; factor ++ ) {
      System.out.println("3 x " + factor + " = " + 3*factor);
}
```

la sentencia **`for`** permite repetir un ciclo **`n`** cantidad de veces, en donde se debe determinar el valor inicial y cuantas veces se repetirá.

La estructura de la instrucción **`for`** es muy simple:

```
for({valor inicial};{condición de termino};{factor de incremento del valor inicial}){
  //En el cuerpo va lo que se repetirá n veces de acuerdo a la condición de termino
}
```

La instrucción **`For`** ahora tiempo y código para ejecutar muchas veces una acción.

A continuación, comparto una presentación: que contiene un resumen de las estructuras de repetición y algunos ejemplos:

Esta presentación será incrustada en la plataforma

URL: https://www.slideshare.net/secret/4Y7vaf0xB1zA5P.

Para finalizar te invito a ver el siguiente vídeo: Curso de java: Sentencias de control 2, que explica el uso de las instrucciones de selección (**`while, do…wile, for`**) creado por: Códigos de Programación disponible en la siguiente dirección**:**

(Este vídeo será incrustado en la plataforma)

https://www.youtube.com/playlist?list=PL-Mlm_HYjCo9vojnWbPTs6nV51J8WV_O5

Conclusión

Hemos visto el uso de las instrucciones de repetición en Java, recordemos que la instrucción **`while`** suele utilizarse para implementar cualquier ciclo controlado por un contador, en el caso de la instrucción **`For`** especifica en su encabezado todos los detalles sobre la repetición controlada por un contador. Cuando esta instrucción comienza a ejecutarse, se declara e inicializa la variable de control, si la condición es verdadera entonces el cuerpo del **`For`** se ejecuta, después de ejecutar el cuerpo del ciclo, se ejecuta la expresión de incremento. Luego, se lleva a cabo otra vez la prueba de continuación de ciclo para determinar si el programa debe continuar con la siguiente ejecución del ciclo.

Actividades

- Descargar y leer Unidad 3: Instrucciones de control (https://dominamoodle.net/proyectoeva/mod/resource/view.php?id=178)

- Ir y leer tutorial de Java en la sección de instrucciones de repetición: (http://www.codexion.com/tutorialesjava/java/)

- (Crear un programa para imprimir gráficos de barra) Una aplicación interesante de las computadoras es la visualización de gráficos convencionales y de barra. Aplique el razonamiento lógico para resolver la siguiente situación: Escriba un programa en el lenguaje Java que lea cinco números, cada uno entre 1 y 30. Por cada número leído, su programa debe mostrar el mismo número de asteriscos adyacentes. Por ejemplo, si su programa lee el número 7, debe mostrar *******. Muestre las barras de asteriscos después de leer los cinco números. Utilice la estructura de repetición más adecuada para resolver esta situación.

Evaluación:

Para valorar a conciencia el esfuerzo y tiempo del estudiante empleado en la realización de su trabajo, se utilizará los siguientes criterios.

- Funcionalidad del programa
- Aplicación del razonamiento lógico
- Estructura del código
- Documentación del código.

Formato de entrega:

El programa en Java con todos los archivos de código fuente debe ser entregado en una carpeta comprimida con formato **.zip y debe establecer el nombre de la carpeta comprimida a Apellido_Nombre_Asignaciones_3_2.zip**

El plazo de entrega es el 21/06/19, y su entrega debe ser a través de la actividad tareas que se encuentra en la sección de Asignaciones (página principal del curso en moodle)

FORO

Saludos cordiales a tod@s,

Este foro corresponde a la clase 2 de la unidad 3, según lo visto en la clase, entendemos que el lenguaje Java cuenta con dos estructuras de repetición muy similares, una es **`while`** y la otra **`do`**…**`while`**.

Responda a las siguientes preguntas ¿Cuál es la similitud en las dos instrucciones? ¿Cuál es su diferencia? ¿En cuáles casos puedo o no utilizar una de las dos? Justifique su respuesta.

Primero debe hacer su aporte de acuerdo a la consigna de este foro, segundo debatir colaborativamente aportando sus acuerdos o desacuerdos y tercero contar su experiencia si ya ha implementados alguna de estas dos instrucciones en sus proyectos de programación.

Plazo de participación en el foro

Para poder tener tiempo suficiente en realizar sus aportes y participar colaborativamente con sus compañeros este foro estará abierto durante 7 días a partir de la fecha que sea publicado.

-Edy Echenique

Clase 3 Instrucción de selección múltiple `switch`

Edy Echenique

Bienvenidos(as) a la última clase virtual de la unidad 3, es importante leer toda la clase y, además, leer y consultar las lecturas obligatorias y opcionales para enriquecer más el contenido de nuestra clase.

En esta clase continuaremos estudiando las estructuras de control, y toca el turno para las estructuras de selección **`Switch`**.

Les recuerdo que lea bien las consignas de la asignación y el foro en esta clase para que pueda realizar sus actividades cumpliendo con lo solicitado.

Los materiales de lectura están disponibles en formato PDF desde la sección de recursos para la unidad 3 (página principal del curso).

Comencemos con nuestra tercera clase:

Instrucción `Switch`

En la clase 1 de esta unidad hablamos de la instrucción de selección simple **`if`** y la doble **`if`** … **`else`**. Hoy vamos a mostrar la instrucción de selección múltiple llamada **`switch`** y

esta puede realizar distintas acciones con base a los posibles valores de una expresión. Para entender el uso de esta estructura vamos a ver un ejemplo práctico, primero lo vamos a solucionar utilizando una estructura de selección doble **`if`**…**`else`** para poder ver la diferencia y comparar la utilidad entre una y otra.

Este ejemplo está basado en un programa que simula una calculadora elemental con las 4 operaciones básicas: suma, resta, multiplicación y división utilizando dos operandos. En este caso vamos a utilizar una variable de tipo **`char`** para indicar el tipo de operación que se va a realizar, también después de realizar la operación vamos a mostrar el resultado en pantalla.

Utilizando una instrucción **`if`**…**`else`** quedaría como el siguiente código:

```
public class MiniCalculadora {
   public static void main(String args[]){
      int a = 1;
      int b = 1;
      char op = '/';
      System.out.print("El resultado es : ");
      if ( op == '+' ) {
          System.out.println( a + b);
      } else if ( op == '-') {
          System.out.println( a - b);
      } else if ( op == '*') {
          System.out.println( a * b);
      } else if ( op == '/') {
          System.out.println( a / b);
      }
   }
}
```

En este ejemplo ya no vemos situaciones de selección simple, podemos ver que hay una cadena de instrucciones **`if…else`** para realizar una selección múltiple, es decir la condición general va más allá de dos alternativas, por lo tanto, es en este tipo de situaciones que podemos utilizar una instrucción de selección múltiple como lo es la instrucción **`switch`**, veamos este mismo ejemplo como se implementaría con esta instrucción:

```
public class MiniCalculadora{
   public static void main(String args[]){
      int a = 1;
```

```
      int b = 1;
      char op = '/';
      System.out.print("El resultado es : ");
      switch ( op ) {
      case '+':
           System.out.println( a + b );
           break;
      case '-':
           System.out.println( a - b );
           break;
      case '*':
           System.out.println( a * b );
           break;
      case '/':
           System.out.println( a / b );
           break;
      default:
           System.out.println("error" );
           break;
      }
   }
}
```

Podemos ver que la instrucción **`switch`** maneja muy bien la selección múltiple a diferencia de la instrucción **`if`**…**`else`** que se limita a dos alternativas, además podemos ver un código más legible y ordenado para este tipo de situaciones, puede darse el caso en que la lista de selección múltiple sea extensa así que esta sería la mejor manera de manejarlo.

Algunos otros aspectos que es necesario aclarar en el uso de esta instrucción son los siguientes: cuando se evalúa la expresión de **`switch`** Java busca una constante con ese mismo valor. Si la encuentra, ejecutará todas las instrucciones asociadas a esa constante hasta que llegue a la expresión **`break`**.

La instrucción **`break`** se utiliza para finalizar la ejecución de una estructura o separar las alternativas. Puede darse el caso de que la constante no coincida con la expresión del **`switch`**, si es este el caso, pasará a ejecutar las instrucciones contenidas en el caso **`default`**.

A continuación, comparto una presentación: que contiene un resumen de las estructuras de repetición y algunos ejemplos:

Esta presentación será incrustada en la plataforma

URL: https://www.slideshare.net/secret/4Y7vaf0xB1zA5P

Para finalizar te invito a ver el siguiente vídeo: Curso de programación en java: La sentencia **`Switch`**, que explica el uso de las instrucciones de selección múltiple (**`switch`**) creado por: Programación ATS disponible en la siguiente dirección https://www.youtube.com/watch?v=T4X0C5bLM0A

(Este vídeo será incrustado en la plataforma)

Tutorial java: Las instrucciones **`Break`** y **`Continue`**. Vídeo que explica el uso de las instrucciones de control de flujo de programa (**`break`** y **`continue`**) creado por: Código Facilito disponible en la siguiente dirección https://www.youtube.com/watch?v=C1W9nfdYlnI

Este recurso se habilitara como enlace externo

Conclusión

La instrucción **`switch`** como vimos en esta clase realiza distintas acciones, con base en los posibles valores de una expresión. La instrucción **`switch`** contiene una secuencia de palabras case y un default. En esta instrucción el programa evalúa la expresión de control y compara su valor con cada palabra case. Si se da una coincidencia se van a ejecutar las instrucciones para ese case. Cada palabra case puede tener n cantidad de instrucciones y no es necesario que estén entre llaves.

Actividades

- Descargar y leer Unidad 3: Instrucciones de control (https://dominamoodle.net/proyectoeva/mod/resource/view.php?id=178)

- Ir y leer tutorial de Java en la sección de instrucciones de control: (http://www.codexion.com/tutorialesjava/java/)

- Planteamiento del problema

Un almacén de pedidos por correo vende cinco productos cuyos precios de venta son los siguientes, producto 1: $2.98, producto 2: $4.50, producto 3: $9.98, producto 4: $4.99, producto 5: $6.87. Escriba un programa en Java que lea una serie de pares de números, como se muestra a continuación:

a. Número de producto

b. Cantidad vendida

Su programa debe utilizar una instrucción **`switch`** para determinar el precio de venta de cada producto debe calcular y mostrar el valor total de venta de todos los productos vendidos. Use un ciclo controlado por centinela para determinar cuándo debe dejar de iterar para mostrar los resultados finales.

Evaluación:

Para valorar a conciencia el esfuerzo y tiempo del estudiante empleado en la realización de su trabajo, se utilizará los siguientes criterios.

- Funcionalidad del programa
- Aplicación del razonamiento lógico
- Estructura del código
- Documentación del código

Formato de entrega:

El programa en Java con todos los archivos de código fuente debe ser entregado en una carpeta comprimida con formato **.zip y debe establecer el nombre de la carpeta comprimida a Apellido_Nombre_Asignaciones_3_3.zip**

El plazo de entrega es el 28/06/19, y su entrega debe ser a través de la actividad tareas que se encuentra en la sección de Asignaciones (página principal del curso en moodle)

FORO

Saludos cordiales a tod@s,

Este foro corresponde a la clase 3 de la unidad 3, Responda a las siguientes preguntas ¿Qué sucede cuando la instrucción break se ejecuta dentro de una instrucción **`while`**, **`for`**, **`do`**…**`while`** o **`switch`**? ¿Cuál es la función principal de la instrucción **`continue`**? Justifique su respuesta.

Primero debe hacer su aporte de acuerdo a la consigna de este foro, segundo debatir colaborativamente aportando sus acuerdos o desacuerdos y tercero contar su experiencia si ya ha implementados alguna de estas dos instrucciones en sus proyectos de programación.

Plazo de participación en el foro

Para poder tener tiempo suficiente en realizar sus aportes y participar colaborativamente con sus compañeros este foro estará abierto durante 7 días a partir de la fecha que sea publicado.

-Edy Echenique

5. Capturas de pantalla de las clases

Capturas de la Clase 1

Clase 1

Edy Echenique

Instrucciones de selección

Bienvenidos(as) a la primera clase virtual de la unidad 3, es importante leer toda la clase y, además, leer y consultar las lecturas obligatorias y opcionales para enriquecer más el contenido de nuestra clase.

En esta unidad habilitaré 3 clases, una por semana los días miércoles, así que, es muy importante que ingresen una vez por semana a la plataforma para leer y revisar las clases publicadas.

Es muy importante que lea bien las consignas de la asignación y el foro para que pueda realizar sus actividades cumpliendo con lo solicitado.

Los materiales de lectura están disponibles en formato PDF desde la sección de recursos para la unidad 3 (página principal del curso).

Comencemos con nuestra primera clase:

Instrucciones de selección

Ya hemos recorrido parte de nuestro camino de este curso: Introducción a la programación con JAVA, hemos ido aprendiendo como crear programas básicos para computadoras utilizando el lenguaje de programación Java, aun nos falta mucho que aprender pues este lenguaje es extenso y requiere muchos meses de práctica, llegamos a la clase No. 1 de la unidad 3, vamos a dedicar esta clase a comprender las técnicas generales para la solución de problemas y aprender a cómo podemos utilizar las instrucciones de selección **`if e if…else`** para tomar decisiones entre diferentes alternativas dentro de un programa.

> *"Antes de escribir un programa que dé solución a un problema, es imprescindible tener una comprensión detallada de todo el problema, además de una metodología cuidadosamente planeada para resolverlo. Al escribir un programa, es igualmente esencial comprender los tipos de bloques de construcción disponibles, y emplear las técnicas comprobadas para construir programas."* **(2012)**, **Paul y Harvey Deitel, Java: Cómo programar**

Algoritmos y Pseudocódigo

Antes de conocer y poner en práctica el uso de las instrucciones **`if e if…else`**, vamos a repasar algunos conceptos de **algoritmos y pseudocódigo**. Un problema de computadora puede resolverse según las instrucciones ejecutadas y el orden en que éstas se ejecutan. Nosotros los humanos cada día implementamos algoritmos para resolver situaciones fáciles y complejas del mundo que nos rodea, por ejemplo, considere el "**algoritmo para pintar una habitación**":

(1) preparar los materiales para pintar; **(2)** lavar la superficie a pintar; **(3)** preparar la pintura a utilizar; **(4)** pintar las paredes; **(5)** esperar a que seque la pintura; **(6)** limpiar manchas no deseas de pintura en el piso. Esta rutina o secuencia de pasos hará que una habitación quede bien pintada.

Ahora, suponga que estos pasos se lleven a cabo en un orden ligeramente distinto:

(1) preparar los materiales para pintar; **(2)** pintar las paredes; **(3)** preparar la pintura a utilizar; **(4)** lavar la superficie a pintar; **(5)** esperar a que seque la pintura; **(6)** limpiar manchas no deseas de pintura en el piso, en este caso seguramente las paredes de la habitación no quedarían pintadas.

Detallar un orden en que se ejecutan una serie de acciones en un programa se le conoce como: **control de ejecución del programa**. Las instrucciones **if** e **if...else** nos permiten aplicar un control a nuestros programas, lo vamos a mostrar en un momento.

El lenguaje del pseudocódigo

El **pseudocódigo** se conoce como un lenguaje informal y permite construir algoritmos sin tener que preocuparse por todos los detalles de la sintaxis del lenguaje de programación Java. El pseudocódigo es como el lenguaje humano, pero no es un lenguaje de programación de computadoras, ya veremos un ejemplo de pseudocódigo.

Instrucciones if e if...else en java

En el **lenguaje Java** vamos a encontrar tres tipos de instrucciones de selección de las cuales vamos a estudiar 2 **(if e if... else)**. La instrucción **if** ejecuta una o varias acciones si el resultado de la condición es verdadera, o salta la ejecución si el resultado de la condición es falsa. La instrucción if...else a diferencia de la anterior ésta va a ejecutar una o varias instrucciones si el resultado de la condición es verdadera o va a ejecutar otra serie de instrucciones si el resultado de la condición es falsa. La instrucción se selección if se le conoce como: instrucción de selección simple y a la instrucción if...else se le conoce como instrucción de selección doble.

Instrucción if

Recuerda que los programas usan las instrucciones **if** para elegir entre ejecutar un conjunto de acciones o no ejecutarlas. Veamos un ejemplo:

Supongamos que la edad de una persona para ser mayor de edad es de 18 años, este ejemplo lo podemos expresar en pseudocódigo de la siguiente manera:

```
Si la edad de una persona es mayor o igual que 18 años

   Imprimir "Es mayor de edad"
```

Estamos utilizando un lenguaje natural para expresar esta situación, note que hay una condición que se debe determinar si es verdadera y en caso de que así lo sea se va a imprimir en pantalla "**Es mayor de edad**" el programa debería continuar en orden a la siguiente instrucción en el pseudocódigo. En caso de que la condición sea falsa se estaría **omitiendo** la instrucción Imprimir y continuaría en orden la siguiente instrucción en el pseudocódigo.

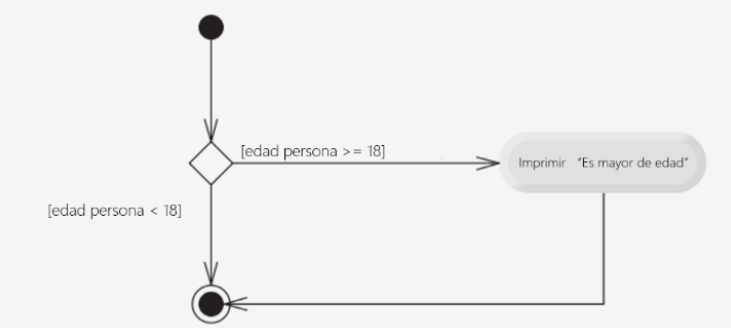


Diagrama de actividad para la instrucción If

La instrucción ***Si*** que acabamos de ver a través de un pseudocódigo se escribiría en el lenguaje Java de la siguiente manera:

```
if (edadPersona >= 18)

      System.out.println("Es mayor de edad")
```

Observe que el código Java es equivalente en gran medida con el pseudocódigo, por eso es que el pseudocódigo es una herramienta muy útil para el desarrollo de programas, ayuda que puedas especificar los pasos de un algoritmo en lenguaje natural y luego esa misma estructura se adapta a un lenguaje de programación.

El siguiente **diagrama de actividad** muestra la instrucción **if** simple del ejemplo que estamos utilizando. Este diagrama contiene un símbolo que es muy importante en los diagramas de actividad como lo es el rombo el cual representa que se tomará una decisión.

En el diagrama si la edad de la persona es mayor o igual a 18 años, el programa imprimirá: "**Es mayor de edad**" y luego se dirige al estado final de la actividad. Si la edad de la persona es menor que 18 años, el programa se dirige de inmediato al estado final sin mostrar un mensaje.

"Los diagramas de actividad forman parte de UML", **(2012)**, **Paul y Harvey Deitel**, los describen de la siguiente manera:

- *"Un diagrama de actividad modela el flujo de trabajo (también conocido como la actividad) de una parte de un sistema de software.*
- *Los diagramas de actividad se componen de símbolos de propósito especial, como los símbolos de estados de acción, rombos y pequeños círculos. Estos símbolos se conectan mediante flechas de transición, las cuales representan el flujo de la actividad.*
- *Los estados de acción representan las acciones a realizar. Cada estado de acción contiene una expresión de acción, la cual especifica una acción específica a realizar.*
- *Las flechas en un diagrama de actividad representan las transiciones, que indican el orden en el que ocurren las acciones representadas por los estados de acción.*
- *El círculo relleno que se encuentra en la parte superior de un diagrama de actividad representa el estado inicial de la actividad: el comienzo del flujo de trabajo antes de que el programa realice las acciones modeladas.*
- *El círculo sólido rodeado por una circunferencia, que aparece en la parte inferior del diagrama, representa el estado final: el término del flujo de trabajo después de que el programa realiza sus acciones.*
- *Los rectángulos con las esquinas superiores derechas dobladas se llaman notas en UML: comentarios que describen el propósito de los símbolos en el diagrama."*

Instrucción if...else

Esta instrucción permite especificar una o varias acciones para ejecutar cuando la condición es verdadera, y otras cuando el resultado de la condición es falsa. Veamos una variante del ejemplo anterior.

Si la edad de una persona es mayor o igual que 18 años

Imprimir "Es mayor de edad"

De lo contrario

Imprimir "Es menor de edad"

Observe que la instrucción que se va a ejecutar si la condición es verdadera será: "**Es mayor de edad**" y "**Es menor de edad**" si la edad de la persona es menor que 18 años.

*La instrucción **Si...de lo contrario** que acabamos de ver a través de un pseudocódigo se escribiría en el lenguaje Java de la siguiente manera:*

```
if (edadPersona >= 18)

   System.out.println("Es mayor de edad")

else

   System.out.println("Es menor de edad")
```

Un aspecto importante que debemos tener siempre en cuenta es el de usar sangría ya que además de permitir una mejor comprensión y visibilidad del código enfatiza la estructura inherente de todos los programas que son estructurados.

> 👉 Buena práctica de programación
>
> *Recuerde utilizar siempre sangría en ambos cuerpos de instrucciones de una estructura **if...else***

Ahora veamos el diagrama de actividad aplicando una instrucción **if...else**, los símbolos en el diagrama de actividad representan estados de ejecución y decisión.

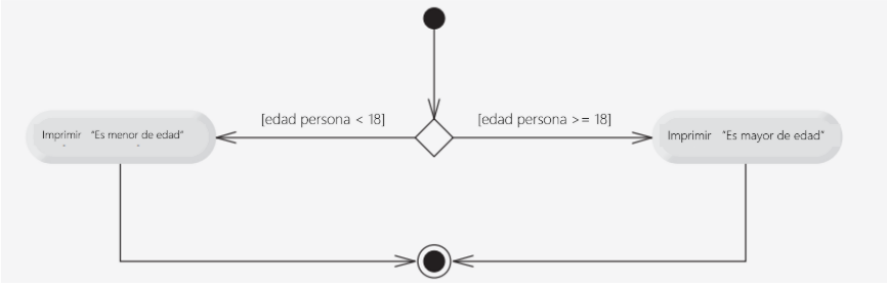
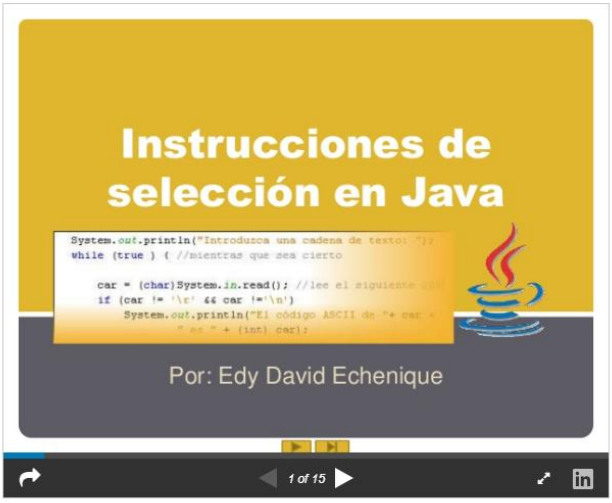


Diagrama de actividad para la instrucción if...else

A continuación, comparto una presentación que contiene un resumen de las estructuras de selección y algunos ejemplos:



Echenique objeto digital de **Edy Echenique**

Para finalizar te invito a ver el siguiente vídeo: Curso de java: Sentencias de control 1, que explica el uso de las instrucciones de selección (if, else...if) creado por: Códigos de Programación**:**

Conclusión

He demostrado el uso de la instrucción de selección simple **if** y la instrucción de selección doble **if...else** acompañado a esto, ahora sabemos que el procedimiento para resolver un problema en acciones a ejecutar y el orden en que éstas se ejecutan se conoce como **algoritmo** y para representar un algoritmo en un lenguaje natural se utiliza el **pseudocódigo** sin tener que preocuparse por los detalles estrictos de la sintaxis del lenguaje.

El uso de **diagramas de actividad** modela el flujo de trabajo de una parte de un programa de computadora. Cada diagrama de actividad se compone de símbolos con propósito especial y todos estos símbolos están conectados mediante flechas de transición y que representan los flujos de actividades.

Asignaciones

1.- Descargar y leer Unidad 3: Instrucciones de control (PDF)

2.- Ir y leer el manual de Java en la sección de instrucciones de control (Enlace externo)

3.- Participar en el foro 1 para esta clase (Foro 1)

4.- Ejercicio Práctico

Vamos a pasar de la teoría a la práctica, y para esto, usted deberá resolver un problema para computadora utilizando los elementos que acabamos de estudiar, a saber, algoritmo con pseudocódigo, programa en Java y diagrama de actividad.

El planteamiento del problema es el siguiente:

Escriba un algoritmo con pseudocódigo que determine si alguno de los clientes de una tienda de departamentos se ha excedido del límite de crédito en una cuenta. Para cada cliente se tienen los siguientes datos:

1. *El número de cuenta.*
2. *El saldo al inicio del mes.*
3. *El total de todos los artículos cargados por el cliente en el mes.*
4. *El total de todos los créditos aplicados a la cuenta del cliente en el mes.*
5. *El límite de crédito permitido.*

En el algoritmo se debe especificar como datos de entrada cada uno de los ítems listados anteriormente en forma de números enteros, su algoritmo debe calcular el nuevo saldo de la siguiente manera:

nuevo saldo = (saldo inicial + cargos - créditos)

después de mostrar el nuevo saldo debe determinar si este excede el límite del crédito del cliente. Para los clientes cuyo límite de crédito sea excedido se debe mostrar un mensaje que diga: "*se excedió el límite de crédito*" en caso contrario debe mostrar un mensaje que diga: "*Su límite de crédito no se ha excedido*".

Luego de escribir su algoritmo con pseudocódigo realice los siguiente:

- El programa en lenguaje Java con su IDE (entorno de desarrollo integrado) de preferencia.
- El diagrama de actividad

Formato de entrega

Esta actividad deberá ser entregada en dos formatos; el algoritmo con el pseudocódigo y el diagrama de actividad deberá entregarlo en un documento con formato **.doc o .pdf**

El programa en Java con todos los archivos de código fuente debe ser entregado en una carpeta comprimida con formato **.zip y debe establecer el nombre de la carpeta comprimida a Apellido_Nombre_Asignaciones_Clase1.zip**

El plazo de entrega es hasta el 29/05/19, y su entrega debe ser a través de la Asignación 1 que se encuentra en la sección de Asignaciones (página principal del curso en moodle)

Capturas de la Clase 2

Clase 2

Edy Echenique

Bienvenidos(as) a la segunda clase virtual de la unidad 3, es importante leer toda la clase y, además, leer y consultar las lecturas obligatorias y opcionales para enriquecer más el contenido de nuestra clase.

En esta clase continuaremos estudiando las estructuras de control, y toca el turno para las estructuras repetitivas.

Les recuerdo que lean bien las consignas de la asignación y el foro en esta clase para que pueda realizar sus actividades cumpliendo con lo solicitado.

Los materiales de lectura están disponibles en formato PDF desde la sección de recursos para la unidad 3 (página principal del curso).

Comencemos con nuestra primera clase:

Instrucciones de repetición

En el lenguaje Java podemos utilizar tres instrucciones de repetición, que permiten a los programas ejecutar bloques de código de manera repetida dependiendo de la condición. Estas instrucciones son las siguientes:

`While, do while y for`

Instrucciones de repetición while

En una instrucción de repetición **while** un programa puede repetir una acción mientras la condición sea verdadera y se detendrá hasta que la condición sea falsa.

Veamos un ejemplo de esta instrucción de repetición en Java. El siguiente ejemplo es de un programa que encuentra la primera potencia de 3 que sea mayor a 60 asumiendo que la variable producto de tipo int se inicia en 3.

```
while (producto <= 80)
  producto = 3 * producto;
```

Vamos a explicar este ejemplo, en cada repetición de la instrucción **while** se multiplicara a la variable producto por **3**, por lo que la variable producto tomara los valores de **9, 27 y 81** sucesivamente, note que cuando la variable producto toma el valor de **81** la condición que mantiene la ejecución del ciclo (**producto <= 80**) dará como resultado el valor de falso y entonces se detendrá la repetición, así que el valor final de la variable producto será de **81**. Es en este punto donde la ejecución del programa continuará con la siguiente instrucción después de la instrucción **while**.

Normalmente esta instrucción se utiliza en las situaciones en donde no se conoce la cantidad de repeticiones que se deben ejecutar para producir un resultado.

Repetición controlada por un contador

En ocasiones necesitaremos ejecutar un ciclo controlando las veces que queremos ejecutarlo, para esto es necesario el uso de un contador de lo cual vamos a hablar en este momento.

Existen varios elementos que son requeridos para llevar a cabo una repetición controlada por un contador:

1. Variable de control (contador de ciclo)
2. Valor inicial de la variable de control
3. Incremento con el que se modifica la variable de control cada vez que pasa por el ciclo.
4. Condición de continuación del ciclo. (determina si el ciclo debe continuar)

Para comprender estos elementos vamos a ver un ejemplo en código:

```
1 public class ContadorWhile
2 {
3      public static void main(String[] args) {
4           int contador = 1; // declara e inicializa la variable de control
5           while (contador <= 10) // condición de continuación de ciclo
6           {
7                System.out.printf("%d ", contador);
8                ++contador; // incrementa la variable de control
9           }
10          System.out.println();
11    }
12 }
```

La salida de este programa sería:

```
1 2 3 4 5 6 7 8 9 10
```

En el código del ejemplo anterior todos los elementos de la repetición controlada por un contador se definen en la línea **4, 5 y 8**, en la línea **4** declara la variable de control contador como un **int** y se establece el valor inicial en **1**.

En la línea **13** incrementa la variable de control en 1, por cada repetición en el ciclo, observe que la condición de continuidad del ciclo en la instrucción **while** (**línea 5**) evalúa si el valor de la variable de control es menor o igual que **10** que sería el valor final para que la condición sea verdadera. Note que el cuerpo del while se ejecuta aun cuando la variable de control es igual a **10**. Solo cuando el valor de la variable es mayor que **10** el ciclo termina o cuando el contador se convierte en **11**.

Instrucciones de repetición while

Esta instrucción es similar a la instrucción **while**. Recordemos que la instrucción **while** el programa evalúa la condición de continuación de ciclo al principio, antes de ejecutar el cuerpo del ciclo; si la condición es falsa, el cuerpo nunca se va a ejecutar. En el caso de la instrucción do **while** evalúa la condición de repetición del ciclo después de ejecutar el cuerpo del ciclo; por lo tanto, el cuerpo siempre se va a ejecutar por lo menos 1 vez.

Veamos un ejemplo donde un programa implementa una instrucción do **while** para imprimir los números del **1 al 10**:

```
1 public class PruebaDoWhile
2 {
3      public static void main(String[] args) {
4           int contador = 1;
5           do
6           {
7                System.out.printf("%d ", contador);
8                ++contador;
9           } while (contador <= 10)
10            System.out.println();
11    }
12}
```

La salida de este programa sería:

```
1 2 3 4 5 6 7 8 9 10
```

Cuál es la diferencia entonces entre la instrucción **while** y **do while**:

Observe que en la línea 4 se declara e inicializa la variable de control contador. Cuando se ejecuta el cuerpo de la instrucción do **while** la línea **7** imprime el valor del contador y la **8** incrementa el contador. Note que después el programa evalúa la prueba de continuación del ciclo al final de este en la línea **9**. Si la condición es verdadera, el ciclo continúa repitiéndose, pero si la condición es falsa el ciclo termina el programa y sigue a la siguiente instrucción después del ciclo.

Instrucción de repetición For

Hasta aquí hemos visto dos instrucciones donde no conocíamos la cantidad de ciclos que se ejecutaban hasta cumplir con una condición. Ahora vamos a ver como utilizar una instrucción donde ya se sabe cuántos ciclos se deben cumplir para terminar la ejecución.

Imaginemos que queremos imprimir la tabla de multiplicar de un número hasta el factor **10**. Sin utilizar ninguna instrucción de repetición tendríamos que escribir **9** líneas de código de la siguiente manera:

```
System.out.println("3 x 1 = 3");

.

.

.

System.out.println("3 x 10 = 27");
```

Para realizar estas operaciones de una manera más rápida podríamos utilizar unas de las instrucciones que ya hemos visto antes, pero ahora toca el turno ala instrucción **For**

Observe el siguiente código:

```
for ( int factor = 1; factor <= 9; factor ++ ) {

      System.out.println("3 x " + factor + " = " + 3*factor);

}
```

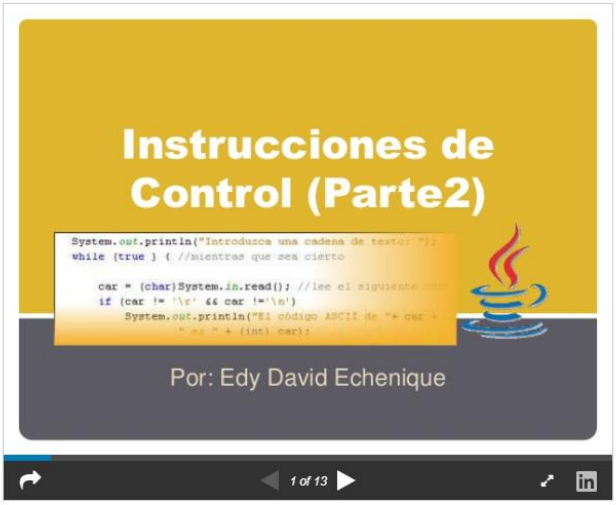
la sentencia **for** permite repetir un ciclo **n** cantidad de veces, en donde se debe determinar el valor inicial y cuantas veces se repetirá.

La estructura de la instrucción **for** es muy simple:

```
for({valor inicial};{condición de termino};{factor de incremento del valor inicial}){
  //En el cuerpo va lo que se repetirá n veces de acuerdo a la condición de termino
}
```

La instrucción **For** ahora tiempo y código para ejecutar muchas veces una acción.

A continuación, comparto una presentación: que contiene un resumen de las estructuras de repetición y algunos ejemplos:



Echenique objeto digital 2 from Edy Echenique

Para finalizar te invito a ver el siguiente vídeo: Curso de java: Sentencias de control 2, que explica el uso de las instrucciones de selección (while, do...wile, for) creado por: Códigos de Programación**:**

Conclusión

Hemos visto el uso de las instrucciones de repetición en Java, recordemos que la instrucción **while** suele utilizarse para implementar cualquier ciclo controlado por un contador, en el caso de la instrucción **For** especifica en su encabezado todos los detalles sobre la repetición controlada por un contador. Cuando esta instrucción comienza a ejecutarse, se declara e inicializa la variable de control, si la condición es verdadera entonces el cuerpo del **For** se ejecuta, después de ejecutar el cuerpo del ciclo, se ejecuta la expresión de incremento. Luego, se lleva a cabo otra vez la prueba de continuación de ciclo para determinar si el programa debe continuar con la siguiente ejecución del ciclo.

Asignaciones

1.- Descargar y leer Unidad 3: Instrucciones de control (PDF)

2.- Ir y leer manual de Java en la sección de instrucciones de control (Enlace externo)

3.- Participar en el foro 1 para esta clase (Foro 2)

4.- Ejercicio Práctico

(Crear un programa para imprimir gráficos de barra) Una aplicación interesante de las computadoras es la visualización de gráficos convencionales y de barra. Aplique el razonamiento lógico para resolver la siguiente situación: Escriba un programa en el lenguaje Java que lea cinco números, cada uno entre 1 y 30.

Por cada número leído, su programa debe mostrar el mismo número de asteriscos adyacentes. Por ejemplo, si su programa lee el número 7, debe mostrar *******. Muestre las barras de asteriscos después de leer los cinco números. Utilice la estructura de repetición más adecuada para resolver esta situación.

Criterios de evaluación:

Para valorar a conciencia el esfuerzo y tiempo del estudiante empleado en la realización de su trabajo, se utilizará los siguientes criterios:

1. Funcionalidad del programa
2. Aplicación del razonamiento lógico
3. Estructura del código
4. Documentación del código

Formato de entrega

El programa en Java con todos los archivos de código fuente debe ser guardados con la extensión .java y entregado en una carpeta comprimida con formato **.zip y debe establecer el nombre de la carpeta comprimida a Apellido_Nombre_Asignaciones_3_2.zip.**

El plazo de entrega es hasta el 05/06/19, y su entrega debe ser a través de la Asignación 2 que se encuentra en la sección de Asignaciones (página principal del curso en moodle)

Capturas de la Clase 3

Clase 3

Edy Echenique

Instrucciones de selección múltiple switch

Bienvenidos(as) a la última clase virtual de la unidad 3, es importante leer toda la clase y, además, leer y consultar las lecturas obligatorias y opcionales para enriquecer más el contenido de nuestra clase.

En esta clase continuaremos estudiando las estructuras de control, y toca el turno para las estructuras de selección **Switch**.

Les recuerdo que lea bien las consignas de la asignación y el foro en esta clase para que pueda realizar sus actividades cumpliendo con lo solicitado.

Los materiales de lectura están disponibles en formato PDF desde la sección de recursos para la unidad 3 (página principal del curso).

Comencemos con nuestra primera clase:

Instrucción switch

En el lenguaje Java podemos utilizar tres instrucciones de repetición, que permiten a los programas ejecutar bloques de código de manera repetida dependiendo de la condición. Estas instrucciones son las siguientes:

`While, do while y for`

Instrucciones de repetición while

En la clase 1 de esta unidad hablamos de la instrucción de selección simple **if** y la doble **if ... else.** Hoy vamos a mostrar la instrucción de selección múltiple llamada **switch** y esta puede realizar distintas acciones con base a los posibles valores de una expresión. Para entender el uso de esta estructura vamos a ver un ejemplo práctico, primero lo vamos a solucionar utilizando una estructura de selección doble **if...else** para poder ver la diferencia y comparar la utilidad entre una y otra.

Este ejemplo está basado en un programa que simula una calculadora elemental con las 4 operaciones básicas: suma, resta, multiplicación y división utilizando dos operandos. En este caso vamos a utilizar una variable de tipo char para indicar el tipo de operación que se va a realizar, también después de realizar la operación vamos a mostrar el resultado en pantalla.

Utilizando una instrucción **if...else** quedaría como el siguiente código:

```
public class MiniCalculadora {

   public static void main(String args[]){

      int a = 1;

      int b = 1;

      char op = '/';

      System.out.print("El resultado es : ");

      if ( op == '+' ) {

          System.out.println( a + b);

      } else if ( op == '-') {

          System.out.println( a - b);

      } else if ( op == '*') {

          System.out.println( a * b);

      } else if ( op == '/') {

          System.out.println( a / b);

      }

   }

}
```

En este ejemplo ya no vemos situaciones de selección simple, podemos ver que hay una cadena de instrucciones **if...else** para realizar una selección múltiple, es decir la condición general va más allá de dos alternativas, por lo tanto, es en este tipo de situaciones que podemos utilizar una instrucción de selección múltiple como lo es la instrucción switch, veamos este mismo ejemplo como se implementaría con esta instrucción:

```
public class MiniCalculadora{

   public static void main(String args[]){

      int a = 1;

      int b = 1;

      char op = '/';

      System.out.print("El resultado es : ");

      switch ( op ) {

      case '+':

           System.out.println( a + b );

           break;

      case '-':

           System.out.println( a - b );

           break;

      case '*':

           System.out.println( a * b );

           break;

      case '/':

           System.out.println( a / b );

           break;

      default:

           System.out.println("error" );

           break;

      }

   }

}
```

Podemos ver que la instrucción **switch** maneja muy bien la selección múltiple a diferencia de la instrucción **if...else** que se limita a dos alternativas, además podemos ver un código más legible y ordenado para este tipo de situaciones, puede darse el caso en que la lista de selección múltiple sea extensa así que esta sería la mejor manera de manejarlo.

Algunos otros aspectos que es necesario aclarar en el uso de esta instrucción son los siguientes: cuando se evalúa la expresión de **switch** Java busca una constante con ese mismo valor. Si la encuentra, ejecutará todas las instrucciones asociadas a esa constante hasta que llegue a la expresión **break**.

Echenique objeto digital de **Edy Echenique**

Para finalizar te invito a ver el siguiente vídeo: Curso de java: Sentencias de control 1, que explica el uso de las instrucciones de selección (if, else...if) creado por: Códigos de Programación**:**

Conclusión

La instrucción **switch** como vimos en esta clase realiza distintas acciones, con base en los posibles valores de una expresión. La instrucción **switch** contiene una secuencia de palabras case y un **default**. En esta instrucción el programa evalúa la expresión de control y compara su valor con cada palabra case. Si se da una coincidencia se van a ejecutar las instrucciones para ese case. Cada palabra case puede tener **n** cantidad de instrucciones y no es necesario que estén entre llaves.

Asignaciones

1.- Descargar y leer Unidad 3: Instrucciones de control (PDF)

2.- Ir y leer manual de Java en la sección de instrucciones de control (Enlace externo)

3.- Participar en el foro 1 para esta clase (Foro 3)

4.- Ejercicio Práctico

(Escribir un programa en Java) Planteamiento del problema: Un almacén de pedidos por correo vende cinco productos cuyos precios de venta son los siguientes, producto 1: $2.98, producto 2: $4.50, producto 3: $9.98, producto 4: $4.99, producto 5: $6.87. Escriba un programa en Java que lea una serie de pares de números, como se muestra a continuación:

- Número de producto
- Cantidad vendida

Su programa debe utilizar una instrucción switch para determinar el precio de venta de cada producto debe calcular y mostrar el valor total de venta de todos los productos vendidos. Use un ciclo controlado por centinela para determinar cuándo debe dejar de iterar para mostrar los resultados finales.

Criterios de evaluación:

Para valorar a conciencia el esfuerzo y tiempo del estudiante empleado en la realización de su trabajo, se utilizará los siguientes criterios:

1. Funcionalidad del programa
2. Aplicación del razonamiento lógico
3. Estructura del código
4. Documentación del código

Formato de entrega

El programa en Java con todos los archivos de código fuente debe ser guardados con la extensión .java entregado en una carpeta comprimida con formato **.zip y debe establecer el nombre de la carpeta comprimida a Apellido_Nombre_Asignaciones_3_3.zip.**

El plazo de entrega es hasta el 12/06/19, y su entrega debe ser a través de la actividad tareas que se encuentra en la sección de Asignaciones (página principal del curso en moodle)

DOCUMENTOS ELABORADOS

UNIVERSIDAD TÉCNICA NACIONAL
VICERRECTORÍA DE INVESTIGACIÓN Y POSTGRADO CENTRO DE FORMACIÓN PEDAGÓGICA Y TECNOLOGÍA EDUCATIVA

MAESTRÍA EN ENTORNOS VIRTUALES DE APRENDIZAJE

GUÍA DIDÁCTICA

PREPARADO POR:
EDY DAVID ECHENIQUE RAMÍREZ

TUTOR DEL PROYECTO:
MARIELA DELAURO

1. Fundamentación de la materia

El curso de Programación 1 forma parte del programa de estudio del bachillerato universitario en ingeniería de sistemas de la UNADECA.

Este curso permite explorar de forma sistemática el estudio de la programación para el discente. Inducirle un pensamiento crítico y mostrarle cómo encontrar formas metódicas, pero distintas, de resolver diversas situaciones de la vida real. Por ello, este curso de programación presenta las herramientas necesarias para desarrollar la habilidad en el discente para encontrar soluciones a problemas diversos utilizando lenguajes de programación de alto nivel.

2. Objetivos

2.1 Objetivo General

Construir soluciones a problemas diversos, utilizando estructuras metódicas de programación mediante pseudocódigo, diagramas de flujo y algoritmos propuestos en un lenguaje de programación.

2.2 Objetivos Específicos

- Ordenar en pasos lógicos, la solución a un problema, utilizando un algoritmo e implementándolo en un lenguaje de programación.
- Desarrollar soluciones de programación utilizando conceptos básicos de clases, reutilización de código y validación de datos.
- Proponer soluciones a problemas complejos utilizando herramientas de decisión y selección, así como el planteamiento del pseudocódigo.
- Separar un problema en partes pequeñas para su implementación mediante la utilización de funciones.
- Integrar soluciones prácticas utilizando arreglos a casos de estudio real.
- Proponer código de programación que utilice apuntadores para la solución de aplicaciones.
- Analizar la solución de un problema por medio de la implementación de clases.

3. Contenidos

- **Introducción a la Programación.** Lenguajes de Programación. Tecnologías de Software. Escribiendo programas: usos de memoria, enteros, aritmética, toma de decisiones.
- **Introducción a las Clases, Objetos y Cadenas.** Definición de clases. Definición de miembros y parámetros. Inicializar objetos con constructores. Utilizando clases en archivos separados. Separar la interfaz de la implementación. Validación de datos.
- **Sentencias de Control.** Algoritmos. Pseudocódigo. Estructuras de control. Usos de: if, if…else, while, do…while, switch. Formulación de algoritmos: controlados por contador, por centinela y semillas.
- **Funciones y una introducción a la Recursión.** Componentes de un programa. Librerías de funciones. Definición de funciones. Sobrecarga de funciones. Recursión.
- **Arreglos y Vectores.** Arreglos. Declaración de arreglos. Ejemplos de aplicación. Paso de arreglos a funciones. Búsqueda y ordenación de arreglos. Arreglos multidimensionales.
- **Apuntadores.** Declaración e inicialización de la variable apuntador. Operadores. Aritmética. Relación entre apuntadores y arreglos. Arreglos de punteros. Funciones de punteros.
- **Clases.** Separando la interfaz de la aplicación por medio de clases. Funciones de acceso y utilidad. Destructores. Objetos miembros de una clase.
- **Sobrecarga de Operadores.** Usos de la sobrecarga de operadores. Fundamentos. Sobrecarga de: operadores binarios, cadenas de inserción y extracción binarias, operadores unarios. Casos de estudio.
- **Programación Orientada a Objetos.** Clases base y derivada. Relación entre clases base y derivada. Constructor y destructor. Ingeniería del software.

4. Metodología de trabajo

Cada contenido de las unidades se desarrollará llevando a un diálogo la teoría, poniendo en práctica los conceptos a partir de los intercambios realizados en el

entorno virtual. Como parte de la metodología de trabajo aclaramos los siguientes aspectos:

 **Clases:**

Cada unidad contiene un máximo de tres clases, estás clases se habilitarán cada miércoles, en las unidades que tengan una duración de una semana se habilitará dos clases, la primera se publicará el día miércoles y la segunda el día domingo.

 **Comunicación:**

Para este curso la comunicación será asíncrona utilizando las diferentes herramientas que ofrece la plataforma moodle; mensajería instantánea, se habilitará un foro de presentación, el espacio para novedades y noticas que ya viene por defecto en cada aula virtual estará habilitado, se habilitará un foro de preguntas y respuestas para tratar aspectos generales del curso., adicional a esto la plataforma moodle tendrá un módulo externo para envió de correo interno que permite envío de texto con formato html y adjunto de archivos.

 **Envío de trabajos:**

Muchas veces los trabajos requeridos implican la realización de algún tipo de archivo (un documento, una imagen, una presentación, etc.). En esos casos, salvo expresa indicación del docente, deben ser enviados por media de la plataforma, adjuntados en los espacios de actividad tipo tarea.

Si los archivos a enviar exceden los 5Mb, recomendamos comprimirlos.

 **Fechas de entrega:**

Todos los trabajos pautados con fecha de entrega serán considerados correctos si, además de cumplir la consigna académica, se entregan dentro de los plazos.

Las entregas fuera de término serán un factor que tomar en cuenta en la calificación.

Es importante destacar que la participación en foros es una de las tareas que más importancia tiene en relación a ajustarse a los tiempos para el debate o la argumentación.

Gramática y Ortografía:

Se tendrá en cuenta en la evaluación la expresión escrita como un contenido transversal importante, el respeto por las normas gramaticales y ortográficas.

Registro de actividades cumplidas:

Es responsabilidad de cada participante llevar un registro de las asignaciones solicitadas y chequear su cumplimiento. Esas asignaciones se consignan claramente en la sección Clases y en las actividades propiamente dentro de la plataforma, incluyendo la fecha de vencimiento de cada una.

5. Evaluación de los aprendizajes

La evaluación de los aprendizajes se realizará a través de las siguientes actividades:

Foros. En los foros los aspectos a tomar en cuenta para la evaluación serán: la escritura de su respuesta a la consigna y comentar o retroalimentar los aportes de otros. Todas las participaciones en los foros deberán estar sustentadas en los materiales de apoyo, también deberán seguir un hilo en cuanto a las participaciones de los pares.

Las participaciones enviadas por mail al docente luego del cierre de los foros no serán tenidas en cuenta como participación.

Trabajos individuales. Cada trabajo individual debe cumplir con las pautas establecidas en la consigna de trabajo, se evaluará la estructura del trabajo, fecha de envío. En los trabajos que se identifique plagio o información de otros compañeros no serán tomados en cuenta para aplicar una calificación.

Exámenes. Habrá un tipo de exámenes, los exámenes parciales que se realizará en la fecha establecida en el calendario académico de la Universidad. El material para evaluar consistirá de las clases previamente estudiadas como parte del curso, hasta la semana anterior al mismo. Todos serán de carácter teórico y práctico a través de la actividad cuestionarios en Moodle.

La ponderación de la evaluación se incluye en la siguiente tabla:

7 forosCantidad	**20%**
9 actividades individuales	50%
3 exámenes parciales	30%
TOTAL	100%

El límite mínimo que se necesita para aprobar el curso es de **70%**

6. Cronograma de trabajo

A continuación, se muestra la duración en tiempo de las diferentes actividades de este curso:

Actividad	Tiempo de duración
unidad 1 - **Foro académico 1** - **Actividad individual 1**	1 semana
unidad 2 - **Foro académico 2** - **Actividad individual 2**	2 semanas
unidad 3 - **Foro académico 3** - **Actividad individual 3**	3 semanas
Examen parcial 1	3 días
unidad 4 - **Actividad individual 4**	1 semana
unidad 5 - **Foro académico 4** - **Actividad individual 5**	2 semanas
unidad 6 - **Actividad individual 6**	2 semanas
Examen parcial 2	3 días
unidad 7 - **Foro académico 5**	1 semana

- **Actividad individual 7**	
Unidad 8 - **Foro académico 6** - **Actividad individual 8**	2 semanas
unidad 9 - **Foro académico 7** - **Actividad individual 9**	2 semanas
Examen parcial 3	3 días

7. Presentación del tutor

Lic. Edy David Echenique Ramírez

@eder1210

Licenciado en Ingeniería de Sistemas Computacionales de la Universidad Adventista de Centro América (UNADECA). Obtuvo la especialización en Entornos Virtuales de Aprendizaje con el Instituto Latinoamericano de Desarrollo Profesional Docente. Actualmente cursa el ciclo de maestría en Entornos Virtuales de Aprendizaje ofrecida por el Instituto Latinoamericano de Desarrollo Profesional Docente.

Integrante del cuerpo docente de planta de la UNADECA para la facultad de Ingeniería de Sistemas Computacionales, impartiendo los cursos de: programación 1 y 2, base de datos, análisis y diseño de sistemas 1 y 2, sistemas operativos 1 y 2, introducción a las tecnologías de información, teoría de autómatas, compiladores e intérpretes.

Responsable de la administración de aulas virtuales basadas en el sistema Moodle para la UNADECA.

Asesor de proyectos de intervención para el grado de licenciatura en Ingeniería de Sistemas Computacionales de la UNADECA.

Formador: Publicación de cursos como: Moodle para profesores y Moodle para administradores en la plataforma Udemy.com

Lic. Edy Echenique

UNIVERSIDAD TÉCNICA NACIONAL
VICERRECTORÍA DE INVESTIGACIÓN Y POSTGRADO CENTRO DE FORMACIÓN PEDAGÓGICA Y TECNOLOGÍA EDUCATIVA

MAESTRÍA EN ENTORNOS VIRTUALES DE APRENDIZAJE

UNIDAD 3: ESTRUCTURAS DE CONTROL

PREPARADO POR:
EDY DAVID ECHENIQUE RAMÍREZ

TUTOR DEL PROYECTO:
MARIELA DELAURO

Índice

Sentencias de control en la programación

Cuando escribimos programas para computadoras es normal que las instrucciones o líneas de código se ejecuten una tras otra, en ese mismo orden en que fueron escritas. A este proceso se le conoce como ejecución de instrucciones secuenciales. Existen instrucciones que vamos a ver más adelante y que permiten a los programadores alterar la ejecución secuencial de un programa. A esto se le conoce con el nombre de transferencia de control. La transferencia de control hace que en un punto de la ejecución de un programa el control sea enviado a otro bloque de código y una vez que este termine de ejecutarse el control es devuelto a la última instrucción que cedió el control. Lo veremos más adelante con ejemplo concretos.

En el lenguaje de programación Java el término “Instrucciones de control” hace referencia a tres tipos de estructuras de control: las estructuras secuenciales, las estructuras de repetición y las estructuras de selección.

Estructuras de secuencia en lenguaje Java

En las computadoras las instrucciones del lenguaje de Java se ejecutan una después de otra, en el orden en que fueron escritas, es decir, en secuencia. A menos que se indique lo contrario.

En el siguiente diagrama de actividad de la figura 2.1 muestra una estructura de secuencia simple y donde se realizan dos cálculos en orden.

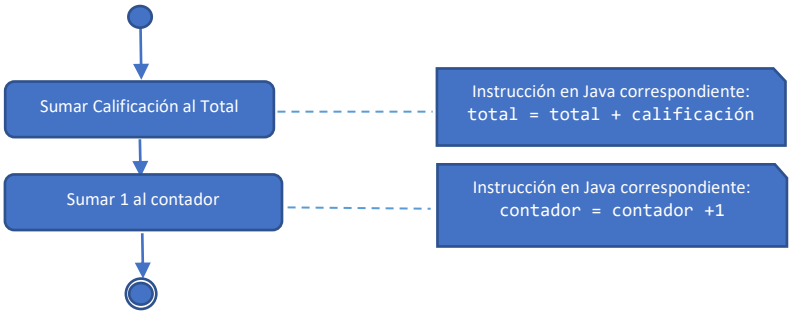


Fig. 2.1 | Diagrama de actividad de una estructura secuencial

En este punto vale la pena mencionar la definición de los diagramas de actividad ya que estaremos usando para representar diferentes procesos dentro de nuestros programas.

P. Deitel (2016: 104) lo define de esta manera: “Un diagrama de actividad de UML modela el flujo de trabajo (también conocido como la actividad) de una parte de un sistema de software. Dichos flujos de trabajo pueden incluir una porción de un algoritmo. Los diagramas de actividad

están compuestos por símbolos, como los símbolos de estado de acción (rectángulos cuyos lados izquierdo y derecho se reemplazan con arcos hacia fuera), rombos y círculos pequeños. Estos símbolos se conectan mediante flechas de transición, que representan el flujo de la actividad; es decir, el orden en el que deben ocurrir las acciones".

En la unidad 1 hablamos de seudocódigo, los diagramas de actividad son herramientas parecida a los seudocódigos que nos ayudan a representar algoritmos. En este caso utilizaremos diagramas de actividad para mostrar con más claridad cómo operan las estructuras de control.

En el diagrama de la figura 2.1 vemos dos estados de acción, donde cada uno contiene una expresión de acción; "sumar calificación a total" o "sumar 1 al contador". Las flechas que hay en el diagrama de actividad nos ayudan a representar la transición que indican el orden en el que ocurren las acciones representadas por estos estados de acción. En este caso como se ve en el diagrama de la figura 2.1 primero se suma calificación al total, y después se suma 1 al contador. Uno de los objetivos de los diagramas de actividad es mostrar con claridad como operan las estructuras de control. Estaremos usando UML para mostrar los flujos de control en las instrucciones de control.

Instrucciones de selección en lenguaje Java

En el lenguaje Java identificamos tres tipos de instrucciones de selección, analizaremos cada una de ellas a continuación.

Instrucciones `if` de selección simple

En términos muy simples podemos decir que una instrucción `if` de selección simple realiza (selecciona) una acción si el resultado de la condición es verdadera, o evita la acción si el resultado de la condición es falsa.

Los programas usan las instrucciones `if` para elegir entre ejecutar un conjunto de acciones o no ejecutarlas. Veamos un ejemplo:

Supongamos que la edad de una persona para ser mayor de edad es de 18 años, este ejemplo lo podemos expresar en pseudocódigo de la siguiente manera:

```
Si la edad de una persona es mayor o igual que 18 años
        Imprimir "Es mayor de edad"
```

Estamos utilizando un lenguaje natural para expresar esta situación, note que hay una condición que se debe determinar si es verdadera y en caso de que así lo sea se va a imprimir en pantalla "**Es mayor de edad**" el programa debería continuar en orden a la siguiente instrucción en el pseudocódigo. En caso de que la condición sea falsa se estaría **omitiendo** la instrucción Imprimir y continuaría en orden la siguiente instrucción en el pseudocódigo.

La instrucción ***Si*** que acabamos de ver a través de un pseudocódigo se escribiría en el lenguaje Java de la siguiente manera:

```
if (edadPersona >= 18)
     System.out.println("Es mayor de edad")
```

Observe que el código Java es equivalente en gran medida con el pseudocódigo, por eso es que el pseudocódigo es una herramienta muy útil para el desarrollo de programas, ayuda que puedas especificar los pasos de un algoritmo en lenguaje natural y luego esa misma estructura se adapta a un lenguaje de programación.

El siguiente **diagrama de actividad** de la figura 2.2 muestra la instrucción **`if`** simple del ejemplo que estamos utilizando. Este diagrama contiene un símbolo que es muy importante en los diagramas de actividad como lo es el rombo el cual representa que se tomará una decisión.

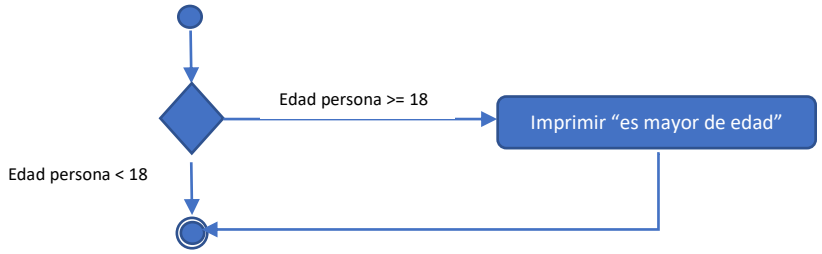


Fig. 2.2 | Diagrama de actividad para la instrucción `if` simple

En el diagrama si la edad de la persona es mayor o igual a 18 años, el programa imprimirá: "**Es mayor de edad**" y luego se dirige al estado final de la actividad. Si la edad de la persona es menor que 18 años, el programa se dirige de inmediato al estado final sin mostrar un mensaje.

Instrucciones if…else de selección doble

Esta instrucción permite especificar una o varias acciones para ejecutar cuando la condición es verdadera, y otras cuando el resultado de la condición es falsa. Veamos una variante del ejemplo anterior.

```
Si la edad de una persona es mayor o igual que 18 años
        Imprimir "Es mayor de edad"
De lo contrario
        Imprimir "Es menor de edad"
```

Observe que la instrucción que se va a ejecutar si la condición es verdadera será: "**Es mayor de edad**" y "**Es menor de edad**" si la edad de la persona es menor que 18 años.

La instrucción ***Si…de lo contrario*** que acabamos de ver a través de un pseudocódigo se escribiría en el lenguaje Java de la siguiente manera:

```
if (edadPersona >= 18)
     System.out.println("Es mayor de edad")
else
System.out.println("Es menor de edad")
```

Un aspecto importante que debemos tener siempre en cuenta es el de usar sangría ya que además de permitir una mejor comprensión y visibilidad del código enfatiza la estructura inherente de todos los programas que son estructurados.

Ahora veamos el diagrama de actividad en la figura 2.3 aplicando una instrucción **`if…else`**, los símbolos en el diagrama de actividad representan estados de ejecución y decisión.

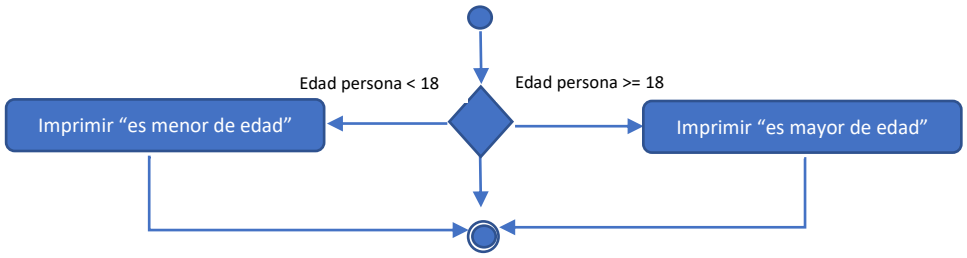


Fig. 2.3 | Diagrama de actividad para la instrucción `if…else`

Instrucciones if…else anidadas

Un programa puede evaluar varios casos utilizando instrucciones `if…else` anidadas, es decir, una instrucción `if…else` dentro de otra instrucción `if…else`. Veamos un ejemplo, en el siguiente seudocódigo vemos una representación de una instrucción `if…else` anidada que imprime "`Excelente`" para las calificaciones de exámenes con nota igual o mayor a `90`, "`Muy bueno`" para las notas en el rango de `80` a `89`, "`Bueno`" para las notas en el rango `70` a `79`, "`Fracaso`" para cualquier otra nota.

```
Si la nota del estudiante es igual o mayor que 90
        Imprimir "Excelente"
de lo contrario
        Si la nota del estudiante es igual o mayor que 80
                Imprimir "Muy bueno"
        de lo contrario
                Si la nota del estudiante es igual o mayor que 70
                        Imprimir "Bueno"
                de lo contrario
                        Imprimir "Fracaso"
```

En lenguaje Java hay dos formas en cómo podrimos escribir este seudocódigo:

```
if (notaEstudiante >= 90)
      System.out.println("Excelente");
else
      if (notaEstudiante >= 80)
            System.out.println("Muy Bueno");
      else
            if (notaEstudiante >= 70)
                  System.out.println("Bueno");
            else
                  System.out.println("Fracaso");
```

Si hubiera más casos que evaluar tuviéramos que seguir agregando instrucciones if…else, y sería un poco difícil comprender el bloque de código, por eso es que la mayoría de los programadores prefieren la forma más corta que mostramos a continuación:

```
if (notaEstudiante>= 90)
      System.out.println("Excelente");
```

```
else if (notaEstudiante>= 80)
     System.out.println(“Muy bueno”);
else if (notaEstudiante >= 70)
     System.out.println(“Bueno”);
else
     System.out.println(“Fracaso”);
```

Las dos formas de escribir el código está correcta, son idénticas, recordemos que en el lenguaje Java el compilador ignora las sangrías, pero la segunda forma es la más popular ya que permite no tener que estar dejando muchas sangrías a la derecha.

Bloques de código

En la instrucción `if`, por lo general, solo se espera una instrucción en su cuerpo. Si queremos añadir más de una instrucción en el cuerpo de un `if` o en el cuerpo de una instrucción `else` debe encerrar las instrucciones entre llaves ({}). Todas las instrucciones encerradas en un par de llaves forman un bloque. A continuación, vemos un ejemplo que incluye un bloque en la instrucción else de un `if…else`.

```
if (calificacion >= 60)
     System.out.println(“Aprobado”);
else{
     System.out.println(“Reprobado”);
     System.out.println(“Debe tomar este curso otra vez.”);
    }
```

Al ejecutar este código anterior si la calificación es menor que 60 el programa imprimirá ambas instrucciones en el cuerpo del else:

```
Reprobado.
Debe tomar este curso otra vez.
```

El uso de las llaves es importante ya que sin ellas solo se ejecutaría la instrucción que imprime “Reprobado” y la instrucción que imprime “Debe tomar este curso otra vez” siempre se imprimiría sin importar que la calificación fuera menor que 60

Instrucciones de selección múltiple switch

Esta instrucción constituye otra forma de usar muchas instrucciones `if` anidadas. Podríamos realizar todo lo que necesitemos con la ayuda de las instrucciones `if…else`, pero la instrucción `switch` puede ser una alternativa más eficiente en circunstancias apropiadas.

Por ejemplo, supongamos que necesitamos una pieza de código para mostrar el día de la semana como una cadena de texto. Imagine que el programa representa el día de la semana como una variable **`int`** llamada **`númeroDía`** con uno de los valores del 1 al 7 para representar los días de lunes a domingo. Queremos convertir la versión numérica del día en una versión de cadena de texto llamada **`nombreDía`**.

Ahora veamos el diagrama de actividad en la figura 2.4 que representa una instrucción `switch` para el ejemplo anterior.

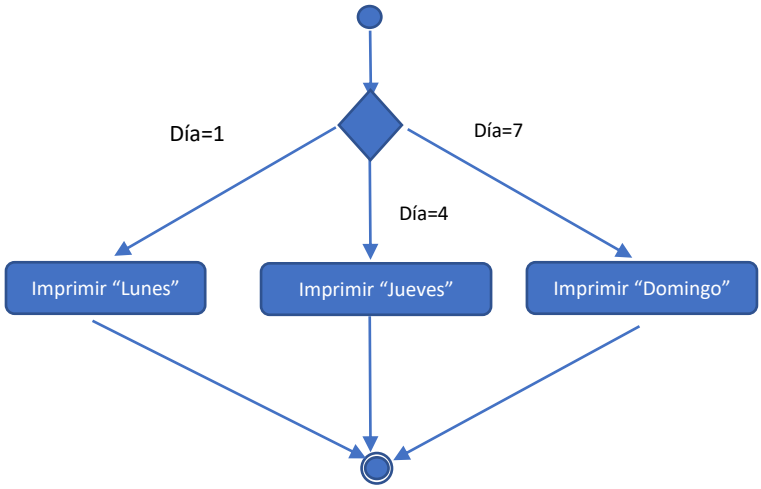


Fig. 2.4 | Diagrama de actividad para la instrucción `switch`

El siguiente bloque de código muestra toda una estructura `switch` para el ejemplo anterior:

```
switch (día) {
      case 1:
           System.out.println(“El día es Lunes”);
           break;
      case 2:
```

```
            System.out.println(“El día es Martes”);
            break;
       case 3:
            System.out.println(“El día es Miércoles”);
            break;
       case 4:
            System.out.println(“El día es Jueves”);
            break;
      case 5:
            System.out.println(“El día es Viernes”);
            break;
      case 6:
            System.out.println(“El día es Sábado”);
            break;
      case 7:
            System.out.println(“El día es Domingo”);
            break;
       default:
            System.out.println("El número dado no corresponde a ningún
día");
            break;
       }
```

En el código Java mostrado anteriormente cuando el flujo de control llega a la instrucción `switch` el programa evaluará la variable “`día`” que está entre paréntesis después de la palabra `switch`, esto en Java es conocido como “expresión de control” en una instrucción `switch`. Lo que va a realizar el programa es comparar el valor de la “expresión de control” con cada una de las opciones “`case`”, por ejemplo, cuando un usuario ingresa un número entero entre 1 al 7

esta se almacena en la variable "`día`" que es la "expresión de control" si el número es 4 la instrucción `switch` compara con cada opción "`case`". Si hay una coincidencia (`case: 4`) el programa ejecuta las instrucciones para esa opción `case`.

Cada vez que hay una coincidencia en las opciones `case`, se ejecutan las instrucciones para ese `case` y las siguientes hasta encontrar una instrucción `break` o el final del `switch`. La instrucción `break` permite terminar la ejecución de una instrucción `switch`.

Al final de una instrucción `switch` vemos que esta el caso `default`, este se ejecuta si no ocurre una coincidencia entre el valor de la expresión de control y una de las opciones `case`. En este ejemplo se ejecutaría el caso `default` si el número ingresado por el usuario no está comprendido entre 1 y 7, cualquier valor fuera de estos llevaría directamente a la ejecución del caso `default`.

No es necesario utilizar una etiqueta `break` para el último case del `switch` ni para el caso `default`, ya que la ejecución va a continuar con la siguiente instrucción que va después del `switch`.

Instrucciones de repetición en lenguaje Java

En el lenguaje de programación Java encontramos tres diferentes instrucciones de repetición (también conocidas como ciclos) que permiten a los programas ejecutar un conjunto de instrucciones de manera repetida, siempre que la condición llamada "condición de ejecución del ciclo" siga en un estado verdadera.

Las instrucciones de repetición que vamos a ver a continuación son: `while`, `for` y `do…while`.

Instrucciones de repetición while

En una instrucción de repetición o de ciclo puede especificar que se repitan varias acciones mientras una condición sea verdadera. Esta instrucción en seudocódigo sería de la siguiente manera

Mientras existan más artículos en mi lista de compras
    Comprar el siguiente artículo y sacarlo de la lista

En este seudocódigo se está describiendo lo que ocurre cuando va de compras al supermercado. Notemos que la condición "*¿Existen más artículos en mi lista de compras?*" podría ser Si o No (`verdadera o falsa`) en caso de ser verdadera se realiza la acción "Comprar el siguiente artículo y sacarlo de la lista" esta acción se va a realizar de manera repetida mientras la condición sea verdadera. Debemos mencionar también que lo que

constituye el cuerpo de la instrucción `while` puede tener de una a muchas instrucciones en caso de ser necesarias. El ciclo terminará cuando la condición sea `falsa` o cuando el último artículo de la lista de compras sea adquirido y eliminado.

La sintaxis de la instrucción `while` es la siguiente:

```
While (condición) {
     Instrucción 1;
     Instrucción 2;
     …
     Instrucción n;
}
```

Consideremos el siguiente ejemplo: Escribir un programa que imprima los números del 1 al 5. Para resolver esta situación es necesario utilizar una instrucción `while` ya que lo podríamos hacerlo con instrucciones de impresión regular (`System.out.println(1);`), pero sería escribir una línea de código por cada número impreso, imaginémonos entonces si quisiéramos imprimir los números del 1 al 500, nuestro programa necesitaría 500 líneas de código de impresión regular. El código en Java para solucionar esta situación sería de la siguiente manera:

```
int i = 1;
while (i <=5) {
     System.out.println(i);
     i ++;
}
```

De esta manera la instrucción `while` se ejecutaría 5 veces imprimiendo el valor de la variable `i`, notemos que la variable `i` sufre un incremento en 1 después de imprimirse el número correspondiente almacenado en la variable `i`, esto es importante ya que si la variable `i` no se incrementa su valor nunca cambiaría y nuestro ciclo se ejecutaría de manera infinita ocasionando un error lógico.

*Instrucciones de repetición con **for***

El lenguaje Java también cuenta con la instrucción de repetición `for`, a diferencia de la instrucción `while`, la instrucción `for` declara los detalles de la repetición en una sola línea.

A continuación de muestra una figura 2.5 con una vista más detallada del encabezado de la instrucción for:

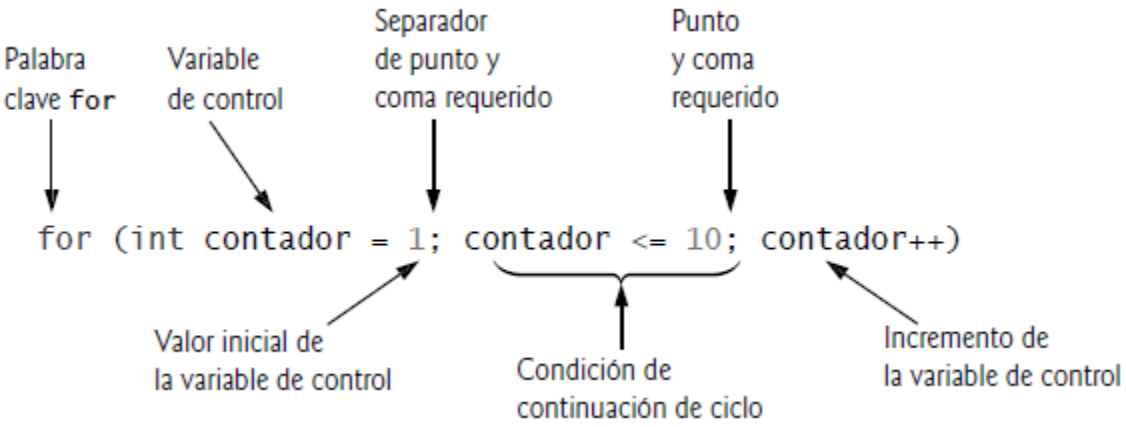


Fig. 2.5 | Encabezado de una instrucción `for`

El encabezado de `for` prácticamente "se encarga de todo", es aquí donde se detallan cada uno de los elementos necesarios para la repetición incluyendo un contador y una variable de control. Debemos tomar en cuenta de que si hay más de una instrucción en el cuerpo del `for` es necesario encerrar entre llaves todas estas instrucciones.

El formato básico para la instrucción `for` es el siguiente:

```
for (inicialización; condiciónDeContinuaciónDeCiclo; incremento)
        instrucciones…
```

Donde la expresión de inicialización declara la variable de control del ciclo y establece opcionalmente su valor inicial, la condición de continuación del ciclo establece si el ciclo debe seguir ejecutándose, y el incremento de la variable de control la modifica para que pueda ir cambiando en cada repetición.

Veamos el siguiente código en Java que incluye una instrucción `for` para imprimir los numero del 1 al 10.

```
1 //Programa de ejemplo de For
2 //Repetición controlada con contador, con la instrucción de repetición for.
3
4 public class ContadorFor
5 {
```

```
6 public static void main(String[] args)
7 {
8 // el encabezado de la instrucción for incluye la inicialización,
9 // la condición de continuación de ciclo y el incremento
10 for (int contador = 1; contador <= 10; contador++)
11    System.out.printf("%d ", contador);
12
13 System.out.println();
14 }
15 } // fin de la clase ContadorFor
```

La salida del programa sería de la siguiente manera:

```
1 2 3 4 5 6 7 8 9 10
```

Notemos en el programa que cuando la instrucción `for` (línea 10 y 11) se ejecuta por primera vez, la variable contadora se inicializa en 1, luego el programa verifica la condición de continuidad del ciclo (`contador <= 10`), observemos que al inicio la condición es verdadera, por lo tanto, la instrucción de la línea 11 imprimirá el valor de la variable contadora que es 1, luego esta variable es incrementada en 1, así que para el segundo ciclo el valor de la variable es 2 y luego es incrementada en 1 una vez más, así sucesivamente hasta que el valor del contador sea 11.

El diagrama de actividad para la instrucción `for` de muestra en la siguiente figura 2.6.

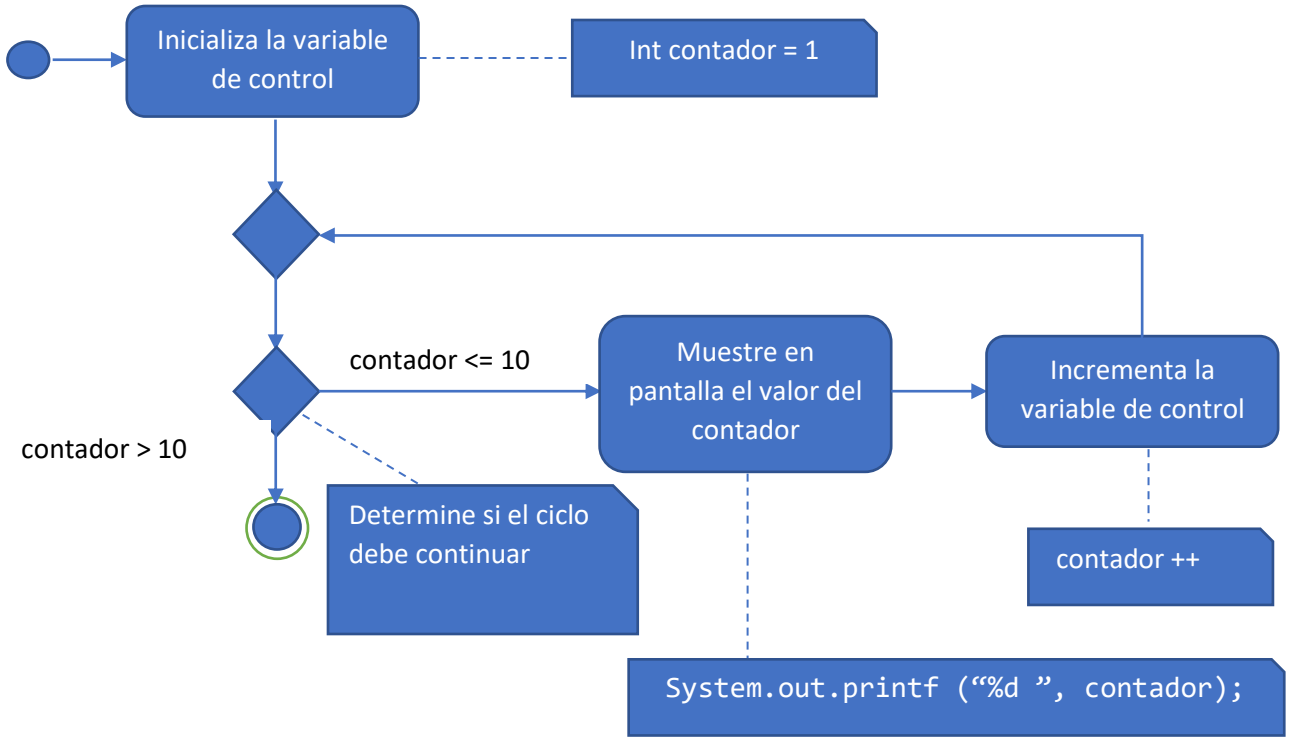


Fig. 2.6 | Diagrama de actividad para la instrucción for

Instrucción de repetición ***do…while***

La instrucción de repetición do `while` al igual que `while` define un procedimiento iterativo. La diferencia consiste en que la instrucción do `while` primero ejecuta lo que tiene dentro del cuerpo y después consulta la condición de continuidad. La instrucción `while` primero consulta y después ejecuta.

La sentencia básica de la instrucción do while es la siguiente:

```
do {
      Instrucción 1;
      Instrucción 2;
      …
} while (condición);
```

A continuación, mostramos un ejemplo de un programa en lenguaje Java que imprime los números del 1 al 10 con esta instrucción.

```
1 //Programa de ejemplo de while
2 //Instrucción de repetición do…while
3
4 public class PruebaDoWhile
5 {
6 public static void main(String[] args)
7 {
8     int contador = 1;
9     do {
10          System.out.printf(“%d”, contador);
11          ++Contador;
12    } while (contador <= 10); //Fin de do…while
13    System.out.println();
14 }
15 } // fin de la clase PruebaDoWhile
```

La salida del programa sería de la siguiente manera:

```
1 2 3 4 5 6 7 8 9 10
```

En el programa anterior en la línea 8 vemos que estamos inicializando la variable de control contador. Al ingresar al do…`while`, en la línea 10 se está imprimiendo el valor del contador y en la línea 11 se está incrementando el contador, luego, el programa evalúa la condición de continuidad del ciclo. En caso de que la condición sea verdadera, el ciclo continúa a partir de la línea 10, si la condición es falsa, el ciclo termina y el programa continúa su ejecución en la siguiente instrucción después del ciclo.

El diagrama de actividad para la instrucción do `while` de muestra en la siguiente figura 2.7.

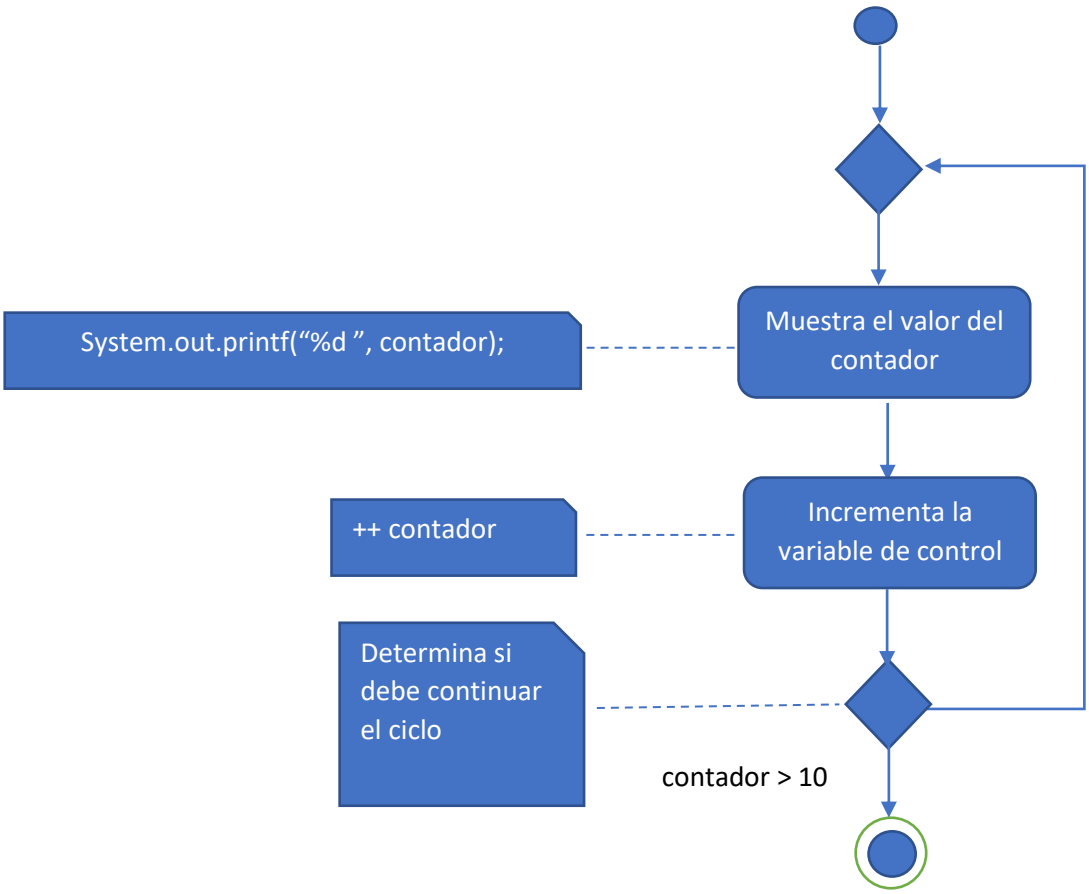


Fig. 2.7 | Diagrama de actividad para la instrucción `do while`

Este diagrama de actividad muestra la instrucción do…`while`. Podemos ver que la condición de continuidad del ciclo no se evalúa sino hasta después que el ciclo ejecuta el estado de acción, por lo menos una vez.

Hasta este punto hemos podido conocer un poco más acerca de las estructuras de control condicionales y repetitivas en Java, todas estas instrucciones son muy útiles en la programación ya que facilitan la ejecución de procesos de una manera más eficiente. Debemos analizar con cuidado cuál instrucción se adapta mejor a cada situación que vamos a implementar.

Estas estructuras son básicas y se utilizan comúnmente en los programas que escribimos sin importar el lenguaje de programación, las instrucciones condicionales nos permiten elegir ciertos caminos de ejecución dependiendo si se cumple o no cierta condición, y las instrucciones repetitivas ayudar a ejecutar acciones ***n*** cantidad de veces hasta que la condición a evaluar sea falsa.

Bibliografía

Ceballos, S. F. J. (2010). *Java 2 : Curso de programación* (4a. ed.). Publicado en Web, https://ebookcentral.proquest.com

Cero, T. los O. en J. con E.-J. desde. (2019, febrero 8). Estructuras Condicionales Java: if, if-else, switch, break, continue, jump. Recuperado el 17 de abril de 2019, de Java desde Cero website: https://javadesdecero.es/basico/if-else-switch-break-continue-jump-java/

Deitel, P. (2016). *Cómo programar en Java*. Ed. Pearson Educación. México

Flórez, F. H. A. (2012). *Programación orientada a objetos usando java*. Publicado en Web, https://ebookcentral.proquest.com

CONCLUSIÓN

Planificar, diseñar y publicar un curso en modalidad virtual ha sido una experiencia enriquecedora. A continuación, resalto algunas conclusiones del proyecto.

El curso de Programación 1 en modalidad virtual fue diseñado bajo la teoría basada en el constructivismo que proponen Vygotsky y Wenger, cuya característica principal de su teoría es el fomento de la reflexión en la experiencia ya que permite por un lado al docente ser un moderador, coordinador, facilitador y mediador y al mismo tiempo participativo a través de actividades en el proceso de aprendizaje. Por otro lado, el estudiante, siendo éste el responsable de su propio proceso de aprendizaje y el procesador activo de la información es el que construye el conocimiento por sí mismo y con la utilización y aplicación de las TIC.

Parte de los resultados obtenidos en este proyecto se desarrolló la guía didáctica para el curso a virtualizar y un módulo como principal recurso bibliográfico para una unidad en específico.

También se hizo la planificación de toda una unidad que incluye el diseño de 3 clases con sus respectivas actividades de aprendizaje, estas actividades incluyen consignas de trabajo y criterios de evaluación.

Por último, quiero resaltar el uso de la plataforma Moodle. Para la propuesta de implementación de este curso, seleccioné este sistema por ser uno de los más populares en ambientes virtuales de aprendizaje en la actualidad, las clases fueron diseñadas en el recurso página que ofrece todas las posibilidades para formatear texto e incluir una gran variedad de recursos multimedia, se utilizaron las actividades de tipo foro y tarea para la comunicación y entrega de asignaciones respectivamente.