CURSO DE PROGRAMACIÓN FULL STACK

Introducción a JAVA

FUNDAMENTOS DEL LENGUAJE

EGG

Introducción a Java

Hasta el momento hemos aprendido los diferentes tipos de estructuras de control comunes a todos los lenguajes de programación, dentro del paradigma de programación imperativa, haciendo uso del pseudo intérprete PSeInt. A partir de esta guía comenzaremos a introducir cada uno de los conceptos vistos hasta el momento, pero haciendo uso de un lenguaje de programación de propósito general como lo es Java.

Una plataforma es un entorno de software o de hardware sobre el cual se ejecutan aplicaciones. La mayoría de las plataformas pueden describirse como una combinación entre el sistema operativo y el hardware subyacente. En este aspecto Java es una plataforma distinta al resto, es una plataforma únicamente de software que corre en la parte superior de otras plataformas basadas en hardware.

JAVA

La Máquina virtual Java (JVM) es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java. La JVM es una máquina implementada en software y cuya emulación se realiza sobre una máquina real.

EJECUCIÓN DE UNA APLICACIÓN

En Java, todo el código fuente se escribe en archivos de texto plano cuya extensión es .java, en el ejemplo: MyProgram.java. Posteriormente, al compilar el código fuente, el compilador realiza un análisis de sintaxis del código escrito en los archivos fuente de Java (con extensión *.java). Si no encuentra errores en el código genera los archivos compilados con extensión .class. En caso de encontrar errores muestra la línea o líneas erróneas en el código. Los archivos .class no contienen código nativo del procesador, sino que contienen un conjunto de instrucciones que se denominan bytecodes, el lenguaje máquina de la JVM. Las instrucciones en bytecodes son independientes del tipo de computadora. Luego, el lanzador de aplicaciones java corre su aplicación con una instancia de la JVM en una computadora específica (Windows, Unix, MacOS, etc).

ARCHIVO FUENTE EN JAVA

Un archivo fuente de la tecnología Java tiene la siguiente forma:

```
[<declaración_paquete>]
<declaración_importacion>*
<modificador>* <declaración_clase>{
      <atributos>*
      <constructores>*
      <metodos>*
}
```

El orden de estos puntos es importante. Cualquier sentencia de importación debe preceder todas las declaraciones de clases. Si usa una declaración de paquetes, el mismo debe preceder tanto a las clases como a las importaciones. El nombre del archivo fuente tiene que ser el mismo que el nombre de la declaración de la clase pública en ese archivo. Un archivo fuente puede incluir más de una declaración de clase, pero *sólo una* puede ser declarada pública. Si un archivo fuente no contiene clases públicas declaradas, el nombre del archivo fuente no está restringido. Sin embargo, es una buena práctica tener un archivo fuente para cada clase declarada y que el nombre del archivo sea idéntico al nombre de la clase.

PAQUETES

Los paquetes son contenedores de clases y su función es la de organizar la distribución de las clases. Los paquetes y las clases son análogos a las carpetas y archivos utilizadas por el sistema operativo, respectivamente.
El lenguaje de programación de la tecnología Java le provee la sentencia package como la forma de agrupar clases relacionadas. La sentencia package tiene la siguiente forma:

```
package <nombre_paq_sup>[.<nombre_sub_paq>]*;
```

La declaración package, en caso de existir, debe estar al principio del archivo fuente y sólo la declaración de un paquete está permitida. Los nombres de los paquetes son jerárquicos (al igual que una organización de directorios en disco) y están separados por puntos. Es usual que sean escritos completamente en minúscula.

IMPORTACIÓN DE CLASES

Si una clase utiliza clases de otros paquetes, estas clases deben ser importadas. Las sentencias de importación van entre la declaración del paquete y la declaración de clase.

```
import <nombre del paquete>[.<nombre del subpaquete>].<nombre de clase>;

import <nombre de paquete>[.<nombre  del subpaquete>].*;
```

Cuando se necesita usar paquetes hay que usar la sentencia *import* para decirle al compilador dónde encontrar las clases. En efecto, el nombre del paquete forma parte del nombre de las clases dentro del paquete.

CLASES

En el mundo de orientación a objetos, todos los programas se definen en término de objetos y sus relaciones. Las clases sirven para modelar los objetos que serán utilizados por nuestros programas. Los objetos y las clases son conceptos que serán abordados con profundidad más adelante en el curso.

Una clase está formada por una parte correspondiente a la declaración de la clase, y otra correspondiente al cuerpo de la misma:

```
Declaración de clase {
Cuerpo de clase
}
```

En la plantilla de ejemplo se ha simplificado el aspecto de la Declaración de clase, pero sí que puede asegurarse que la misma contendrá, como mínimo, la palabra reservada *class* y el nombre que recibe la clase. El cuerpo de las clases comienza con una llave abierta ({) y termina con una llave cerrada (}).

MÉTODOS

Dentro del cuerpo de la clase se declaran los atributos y los métodos de la clase. Los métodos son rutinas que contienen una o más sentencias que una vez ejecutadas deberían llevar adelante un objetivo concreto. Si bien es un tema sobre el que se profundizará más adelante en el curso, los métodos son de vital importancia para los objetos y las clases. En un principio, para dar los primeros pasos en Java nos alcanza con esta definición.

MÉTODO MAIN()

Para que un programa se pueda ejecutar, debe contener una clase que tenga un método *main()* con la siguiente declaración:

```
public static void main(String[] args){
```

A continuación describirnos cada uno de los modificadores y componentes que se utilizan *siempre* en la declaración del método *main():*

public: es un tipo de acceso que indica que el método *main()* es público y, por tanto, puede ser llamado desde otras clases. Todo método *main()* debe ser público para poder ejecutarse desde el intérprete Java (JVM).

static: es un modificador el cual indica que la clase no necesita ser instanciada para poder utilizar el método. También indica que el método es el mismo para todas las instancias que pudieran crearse.

void: indica que la función o método *main()* no devuelve ningún valor.

El método *main()* debe aceptar siempre, como parámetro, un vector de strings, que contendrá los posibles argumentos que se le pasen al programa en la línea de comandos, aunque como es nuestro caso, no se utilice.

Luego, al indicarle a la máquina virtual que ejecute una aplicación el primer método que ejecutará es el método *main()*. Si indicamos a la máquina virtual que corra una clase que no contiene este método, se lanzará un mensaje advirtiendo que la clase que se quiere ejecutar no contiene un método *main()*, es decir que dicha clase no es ejecutable.

Si no se han comprendido hasta el momento muy bien todos estos conceptos, los mismos se irán comprendiendo a lo largo del curso.

IDENTIFICADOR

Los identificadores son los nombres que se usan para identificar cada uno de los elementos del lenguaje, como ser, los nombres de las variables, nombres de las clases, interfaces, atributos y métodos de un programa. Si bien Java permite nombres de identificadores tan largos que se desee, es aconsejable crearlos de forma que tengan sentido y faciliten su interpretación. El nombre ideal para un identificador es aquel que no se excede en longitud (lo más corto posible) y que califique claramente el concepto que intenta representar en el contexto del problema que se está resolviendo.

COMENTARIOS

Los comentarios en los programas fuente son muy importantes en cualquier lenguaje. Sirven para aumentar la facilidad de comprensión del código y para recordar ciertas cosas sobre el mismo. Son porciones del programa fuente que *no serán compiladas*, y, por tanto, no ocuparán espacio en el archivo "ejecutable". Únicamente sirven para documentar.

```
numero = 0 // Le asigno un valor a la variable numero
```

ESTRUCTURAS DE CONTROL

Las estructuras de control son construcciones hechas a partir de palabras reservadas del lenguaje que permiten modificar el flujo de ejecución de un programa. De este modo, pueden crearse construcciones de alternativas mediante sentencias condicionales y bucles de repetición de bloques de instrucciones. Hay que señalar que un bloque de instrucciones se encontrará encerrado mediante llaves {.....} si existe más de una instrucción.

SENTENCIAS CONDICIONALES

Los condicionales son sentencias de control que cambian el flujo de ejecución de un programa de acuerdo a si se cumple o no una condición. Cuando el flujo de control del programa llega al condicional, el programa evalúa la condición y determina el camino a seguir. Existen dos tipos de sentencias condicionales, las sentencias *if / else* y la sentencia *switch*.

La *sentencia if* es la más básica de las sentencias de control de flujo. Esta sentencia le indica al programa que ejecute cierta parte del código sólo si la condición evaluada es verdadera («true»). La forma más simple de esta sentencia es la siguiente:

```
if(<condición>){
      <sentencias>
}
```

En donde, <*condición*> es una expresión condicional cuyo resultado luego de la evaluación es un dato booleano(lógico) verdadero o falso. El bloque de instrucciones <sentencias> se ejecuta si, y sólo si, la expresión (que debe ser lógica) se evalúa a true, es decir, se cumple la condición.

Luego, en caso de que la condición no se cumpla y se quiera ejecutar otro bloque de código, otra forma de expresar esta sentencia es la siguiente:

```
if(<condición>){
      <sentencias A>
} else {
      <sentencias B>
}
```

El flujo de control del programa funciona de la misma manera, cuando se ejecuta la sentencia if, se evalúa la expresión condicional, si el resultado de la condición es verdadero se ejecutan las sentencias que se encuentran contenidas dentro del bloque de código if (<sentencias A>). Contrariamente, se ejecutan las sentencias contenidas dentro del bloque else (<sentencias B>).

En muchas ocasiones, se anidan estructuras alternativas if-else, de forma que se pregunte por una condición si anteriormente no se ha cumplido otra y así sucesivamente.

```
if (<condicion1>) {
<sentencias A>
} else if(<condicion2>){
<sentencias B>
} else {
<sentencias C>
}
```

Al contrario de la sentencia if / else, la sentencia *switch* permite cualquier cantidad de rutas de ejecución posibles. Un switch funciona con los datos primitivos byte, short, char e int. También funciona con Enumeraciones, tema que se verá más adelante en el curso, y con unas cuantas clases especiales que «envuelven» a ciertos tipos primitivos: Character, Byte, Short, and Integer (tema que trataremos cuando se profundice en Orientación a Objetos).

```
switch(<variable){
case <valor1>:
<sentencias1>
break;
case <valor2>:
<sentencias2>
break;
default:
<sentencias3>
}
```

El bloque *switch* evalúa qué valor tiene la variable, y de acuerdo al valor que posee ejecuta las sentencias del bloque case correspondiente, es decir, del bloque case que cumpla con el valor de la variable que se está evaluando dentro del switch.

El uso de la sentencia break que va detrás de cada case termina la sentencia switch que la envuelve, es decir que el control de flujo del programa continúa con la primera sentencia que se encuentra a continuación del cierre del bloque switch. Si el programa comprueba que se cumple el primer valor (valor1) se ejecutará el bloque de instrucciones <sentencias1>, pero si no se encuentra inmediatamente la sentencia break también se ejecutarían las instrucciones <sentencias2>, y así sucesivamente hasta encontrarse con la palabra reservada break o llegar al final de la estructura.

Las instrucciones dentro del bloque default se ejecutan cuando la variable que se está evaluando no coincide con ninguno de los valores case. Esta sentencia equivale al else de la estructura if-else.

SENTENCIAS DE ITERACIÓN

Las sentencias de Iteración ejecutan un conjunto de sentencias tantas veces como lo establezca una condición determinada.

While

La sentencia *while* ejecuta un bloque de instrucciones mientras se cumple una condición. La condición se comprueba antes de empezar a ejecutar por primera vez el bucle, por lo tanto, si la condición se evalúa a «false» en la primera iteración, entonces el bloque de instrucciones no se ejecutará ninguna vez.

```
while (<condición>) {
        <sentencias>
}
```

Do / While

En este tipo de bucle, el bloque instrucciones se ejecuta siempre al menos una vez. El bloque de instrucciones se ejecutará mientras la condición se evalúe a «true». Por lo tanto, entre las instrucciones que se repiten deberá existir alguna que, en algún momento, haga que la condición se evalúe a «false», de lo contrario el bucle será infinito.

```
do {
        <sentencias>
} while (<condición>);
```

La diferencia entre *do-while* y *while* es que do-while evalúa su condición al final del bloque en lugar de hacerlo al inicio. Por lo tanto, el bloque de sentencias después del "do" siempre se ejecutan al menos una vez.

For

La sentencia *for* proporciona una forma compacta de recorrer un rango de valores cuando la cantidad de veces que se deber iterar un bloque de código es conocida. La forma general de la sentencia for se puede expresar del siguiente modo:

```
for (<inicialización>; <terminación>; <incremento>) {
        <sentencias>
}
```

La expresión <inicialización> inicializa el bucle y se ejecuta una sola vez al iniciar el bucle. El bucle termina cuando al evaluar la expresión <terminación> el resultado que se obtiene es false. La expresión <incremento> se invoca después de cada iteración que realiza el bucle; esta expresión incrementa o decrementa un valor hasta que se cumpla la condición de <terminación> del bucle.

La sentencia for también ha sido mejorada para iterar de manera más compacta las colecciones y los arreglos, tema que se verá más adelante en este curso. Esta versión mejorada se conoce como for-each.

Como regla general puede decirse que se utilizará el bucle for cuando se conozca de antemano el número exacto de veces que ha de repetirse un determinado bloque de instrucciones. Se utilizará el bucle do-while cuando no se conoce exactamente el número de veces que se ejecutará el 3bucle, pero se sabe que por lo menos se ha de ejecutar una. Se utilizará el bucle while cuando es posible que no deba ejecutarse ninguna vez.

Sentencias de Salto

En Java existen dos formas de realizar un salto incondicional en el flujo "normal" de un programa. A saber, las instrucciones break y continue.

Break

La instrucción *break* sirve para abandonar una estructura de control, tanto de las condicionales (if-else y switch) como de las repetitivas (for, do-while y while). En el momento que se ejecuta la instrucción break, el control del programa sale de la estructura en la que se encuentra contenida.

Continue

La sentencia *continue* altera la ejecución del cuerpo de un bucle, pero en lugar de salir del bucle, continúa con la siguiente iteración. La instrucción continue transfiere el control del programa desde la instrucción continue directamente a la cabecera del bucle (for, do-while o while) donde se encuentra.

FUNCIONES

Las funciones o métodos son un conjunto de líneas de código (instrucciones), encapsulados en un bloque, usualmente reciben parámetros, cuyos valores utilizan para efectuar operaciones y adicionalmente retornan un valor. En otras palabras una función puede recibir parámetros o argumentos (algunas no reciben nada), hace uso de dichos valores recibidos como sea necesario y retorna un valor usando la instrucción return, si no retorna algo, entonces no es una función. Los tipos que pueden usarse en la función son: int, doble, boolean y String

```
[acceso][modificador][tipo] nombreFuncion([tipo] nombreArgumento, .....){

   /*
      * Bloque de instrucciones
   */

   return valor;
}
```

PROCEDIMIENTOS

Los procedimientos son básicamente un conjunto de instrucciones que se ejecutan sin retornar ningún valor, hay quienes dicen que un procedimiento no recibe valores o argumentos, sin embargo, en la definición no hay nada que se lo impida. En el contexto de Java un procedimiento es básicamente un método o función cuyo tipo de retorno es void que no nos obliga a utilizar una sentencia return.

```
[acceso][modificador] void nombreFuncion([tipo] nombreArgumento.){

   /*
      * Bloque de instrucciones
   */

}
```

Acerca de los argumentos o parámetros:

Hay algunos detalles respecto a los argumentos de un método, veamos:
* Una función, un método o un procedimiento pueden tener una cantidad infinita de parámetros, es decir pueden tener cero, uno, tres, diez, cien o más parámetros. Aunque habitualmente no suelen tener más de 4 o 5.
* Si una función tiene más de un parámetro cada uno de ellos debe ir separado por una coma.
* Los argumentos de una función también tienen un tipo y un nombre que los identifica. El tipo del argumento puede ser cualquiera y no tiene relación con el tipo del método.
* Al recibir un argumento nada nos obliga a hacer uso de éste al interior del método, sin embargo, para que recibirlo si no lo vamos a usar.

Consejos acerca de return:

* Cualquier instrucción que se encuentre después de la ejecución de return NO será ejecutada. Es común encontrar funciones con múltiples sentencias return al interior de condicionales, pero una vez que el código ejecuta una sentencia return lo que haya de allí hacia abajo no se ejecutará.
* El tipo de valor que se retorna en una función debe coincidir con el del tipo declarado a la función, es decir si se declara int, el valor retornado debe ser un número entero.
* En el caso de los procedimientos (void) podemos usar la sentencia return pero sin ningún tipo de valor, sólo la usaríamos como una manera de terminar la ejecución del procedimiento.

ARREGLOS: VECTORES Y MATRICES

Un arreglo es un contenedor de objetos que tiene un número fijo de valores del mismo tipo. La longitud del arreglo es establecida cuando el arreglo es creado y luego de su creación su longitud es fija. Cada una de las celdas de un arreglo es llamada elemento y cada elemento puede ser accedido por un índice numérico.

Al igual que la declaración de otros tipos de variables, la declaración de un arreglo tiene dos componentes: el tipo de dato del arreglo y su nombre. El tipo de dato del arreglo se escribe como *tipo[]*, en donde tipo es el tipo de dato de cada elemento contenido en él. Los corchetes son un símbolo especial que indica que esa variable almacena un arreglo. El tamaño del arreglo no es parte de su tipo (es por esto que los corchetes están vacíos).

Una vez declarado un arreglo hay que crearlo, es decir, hay que asignar al arreglo memoria suficiente para almacenar los valores. La creación de un arreglo se hace con el operador *new*.

Declaración y creación de un Vector

```
tipo[] arregloV = new tipo[Tamaño];
```

Declaración y creación de una Matriz

```
tipo[][] arregloM = new tipo[Filas][Columnas];
```

VECTORES Y MATRICES EN SUBPROGRAMAS

Los arreglos se pueden pasar como parámetros a un subprograma (función o procedimiento) del mismo modo que las variables, poniendo el tipo de dato delante del nombre del vector o matriz y sumandole las llaves para el vector o matriz. Sin embargo, hay que tener en cuenta que la diferencia entre los arreglos y las variables, es que los arreglos siempre se pasan por referencia.

```
public static void llenarVector(int [ ] vector){

}
public static void mostrarMatriz(int [ ][ ] matriz){

}
```

A diferencia de Pseint, en Java si podemos devolver un vector o una matriz en una función para usarla en otro momento en el algoritmo principal (main). Lo que hacemos es poner como tipo de dato de la función, el tipo de dato que tendra el vector y asi poder devolverlo.

```
public static int devolverVector(){

int[ ] vector = new int[5];

return vector;
```

PREGUNTAS DE APRENDIZAJE

1) El archivo de un programa en Java debe terminar con la extensión de archivo:

a) .class
b) .java
c) .jar
d) Ninguna de las anteriores

2) Cuando se compila un programa en Java, el archivo producido por el compilador termina con la extensión:

a) .class
b) .java
c) .jar
d) Ninguna de las anteriores

3) ¿Qué es el bytecode en Java?

a) El formato de intercambio de datos
b) El formato que obtenemos tras compilar un código fuente .java
c) Un tipo de variable
d) Un depurador de código

4) Una aplicación ejecutable que representa un procesador genérico sobre el cual se ejecutan los bytecodes de Java es:

a) La máquina virtual de Java (JVM)
b) Ambiente de ejecución de Java
c) Librerías de clases Java
d) Ninguna de las anteriores

5) ¿Con qué se comienza la ejecución de un programa Java?

a) Con package
b) Con códigos
c) Con el programa o método main()
d) Todas las anteriores

6) Para mostrar mensajes por pantalla se usa:

a) System.out.printer()
b) System.out.prin()
c) System.out.println()
d) Ninguna de las anteriores

7) Teniendo en cuenta que los paquetes y las clases son análogos a las carpetas y archivos utilizadas por el sistema operativo, ¿el nombre de la clase debe ser diferente al del paquete?

a) Siempre
b) Nunca
c) No importa
d) Ninguna de las anteriores

8) ¿Cuál es el comando que se utiliza para ejecutar un programa en Java desde la consola?

a) Java
b) Javac
c) Javax
d) Execute

9) El double se aplica para datos tipo:

a) Entero
b) Decimal
c) Carácter
d) Ninguno de los anteriores

10) ¿El llamado a una librería se hace haciendo uso de la sentencia:

a) String
b) Continue
c) Package
d) Import

11) ¿Cuáles son las sentencias de iteración?

a) El Bucle for, while y do/while
b) Bucle for e if/else
c) Bucle while y switch
d) Ninguna de las anteriores

12) ¿Cuáles son las sentencias de salto?

a) Try y break
b) Break y continue
c) Continue y switch
d) While y break

13) ¿Qué diferencia hay entre un bucle while y un bucle for?

a) El bucle for puede llegar a ejecutarse nunca pero el while siempre se ejecuta al menos una vez.
b) El bucle for se ejecuta un número determinado de veces y el while un número indeterminado de veces
c) El bucle for no puede convertirse en un bucle while, pero sí al contrario.
d) El El bucle while permite su inicialización, pero el bucle for no lo permite.

PREGUNTAS EXTRA:

1) Investigue al menos otros 5 tipos de proyectos que pueden generarse desde el entorno y para qué sirven esos tipos de proyectos.

2) Investigar la documentación de la clase System y responder:

- ¿Que representa la clase System?
- ¿Para qué sirve el atributo out de la clase System?
- ¿Para qué sirve el atributo in de la clase System?
- ¿Para qué sirve el atributo err de la clase System?

EJERCICIOS DE APRENDIZAJE

PRIMEROS PASOS

1. Instalando Java: La primera tarea que tenemos que llevar adelante es la instalación del entorno de desarrollo.

Kit de Desarrollo Java

Lo primero que haremos será instalar Java en nuestras computadoras. Para ello debemos descargarlo a través del siguiente link:

Java SE Development Kit 8 Downloads

Una vez instalado vamos a corroborar de que el entorno ha sido instalado de manera correcta. En la consola ejecutaremos el siguiente comando:

```
java -version
```

Si todo se instaló correctamente deberíamos obtener una respuesta como esta:

```
java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)
```

ENTORNOS DE DESARROLLO

Una vez instalado Java procedemos a instalar un entorno de desarrollo que proporciona una serie de servicios y herramientas integrales que tienen como fin facilitar las tareas de desarrollo, testing e implementación de software. Para llevar adelante el curso recomendamos instalar NetBeans, pero se puede utilizar cualquiera de los siguientes IDEs:

NetBeans IDE

Spring Tools Suite (STS)

Eclipse

A partir de ahora comenzaremos a aprender cómo los mismos algoritmos que diseñamos en PSeInt podemos escribirlos también en Java, simplemente haciendo una traducción de cada una de las estructuras de control vistas en PSeInt a Java.

Si bien en esta guía se proponen nuevos problemas, se sugiere que los mismos ejercicios ya implementados en PSeInt sean traducidos al lenguaje de programación Java.

VER VIDEO: Primer programa en Java con Netbeans

Para la realización cada uno de los siguientes ejercicios se debe definir una clase y colocar toda la implementación dentro del método *main()* de dicha clase, tal cual se indica en el video. Para conocer los tipos de datos y operadores disponibles en Java pueden consultar el "*Apéndice A*".

Dificultad Baja

Dificultad Media

Dificultad Alta

2. Escribir un programa que pida dos números enteros por teclado y calcule la suma de los dos. El programa deberá después mostrar el resultado de la suma

3. Escribir un programa que pida tu nombre, lo guarde en una variable y lo muestre por pantalla.

4. Escribir un programa que pida una frase y la muestre toda en mayúsculas y después toda en minúsculas. Nota: investigar la función toUpperCase() y toLowerCase() en Java.

5. Dada una cantidad de grados centígrados se debe mostrar su equivalente en grados Fahrenheit. La fórmula correspondiente es: F = 32 + (9 * C / 5).

6. Dada las horas trabajadas de una persona y el valor por hora. Calcular su salario e imprimirlo. Salario = Hs. trabajadas * valor por hora

7. Escribir un programa que lea un número entero por teclado y muestre por pantalla el doble, el triple y la raíz cuadrada de ese número. Nota: investigar la función Math.sqrt().

8. Dado un tiempo en minutos, calcular su equivalente en días, horas y minutos. Por ejemplo, si el usuario ingresa 1600 minutos, el sistema debe calcular su equivalente: 1 día, 2 horas y 40 minutos.

9. Declarar cuatro variables de tipo entero A, B, C y D y asignarle un valor diferente a cada una. A continuación, realizar las instrucciones necesarias para que: B tome el valor de C, C tome el valor de A, A tome el valor de D y D tome el valor de B. Mostrar los valores iniciales y los valores finales de cada variable. Utilizar sólo una variable auxiliar.

VER VIDEO: Condicionales en Java

10. Implementar un programa que dado dos números enteros determine cuál es el mayor y lo muestre por pantalla.

11. Crear un programa que dado un numero determine si es par o impar.

12. Crear un programa que pida una frase y si esa frase es igual a "eureka" el programa pondrá un mensaje de Correcto, sino mostrará un mensaje de Incorrecto. **Nota: investigar la función equals() en Java.**

13. Realizar un programa que pida introducir solo frases o palabras de 8 de largo. Si el usuario ingresa una frase o palabra de 8 de largo se deberá de imprimir un mensaje por pantalla que diga "CORRECTO", en caso contrario, se deberá imprimir "INCORRECTO". **Nota: investigar la función Lenght() en Java.**

14. Escriba un programa que pida una frase o palabra y valide si la primera letra de esa frase es una 'A'. Si la primera letra es una 'A', se deberá de imprimir un mensaje por pantalla que diga "CORRECTO", en caso contrario, se deberá imprimir "INCORRECTO". **Nota: investigar la función Substring y equals() de Java.**

15. Elaborar un algoritmo en el cuál se ingrese un número entre 1 y 10 y se muestre su equivalente en romano.

    ¿No te acordás los números romanos? Consultalos acá.

16. Elaborar un algoritmo en el cuál se ingrese una letra y se detecte si se trata de una vocal. Caso contrario mostrar un mensaje.

17. Una obra social tiene tres clases de socios:

    - Los socios tipo 'A' abonan una cuota mayor, pero tienen un 50% de descuento en todos los tipos de tratamientos odontológicos.
    - Los socios tipo 'B' abonan una cuota moderada y tienen un 35% de descuento para los mismos tratamientos que los socios del tipo A.
    - Los socios que menos aportan, los de tipo 'C', no reciben descuentos sobre dichos tratamientos.

    Solicite una letra (carácter) que representa la clase de un socio, y luego un valor real que represente el costo del tratamiento (previo al descuento) y determine el importe en efectivo a pagar por dicho socio.

18. Considera que estás desarrollando una web para una empresa que fabrica motores (suponemos que se trata del tipo de motor de una bomba para mover fluidos). Definir una variable tipoMotor y permitir que el usuario ingrese un valor entre 1 y 4. El programa debe mostrar lo siguiente:

    - Si el tipo de motor es 1, mostrar un mensaje indicando "La bomba es una bomba de agua".

- Si el tipo de motor es 2, mostrar un mensaje indicando “La bomba es una bomba de gasolina”.
- Si el tipo de motor es 3, mostrar un mensaje indicando “La bomba es una bomba de hormigón”.
- Si el tipo de motor es 4, mostrar un mensaje indicando “La bomba es una bomba de pasta alimenticia”.
- Si no se cumple ninguno de los valores anteriores mostrar el mensaje “No existe un valor válido para tipo de bomba”

VER VIDEO: Bucles y sentencias de salto break y continue

19. Escriba un programa que valide si una nota está entre 0 y 10, sino está entre 0 y 10 la nota se pedirá de nuevo hasta que la nota sea correcta.

20. Escriba un programa en el cual se ingrese un valor límite positivo, y a continuación solicite números al usuario hasta que la suma de los números introducidos supere el límite inicial.

21. Leer la altura de N personas y determinar el promedio de estaturas que se encuentran por debajo de 1.60 mts. y el promedio de estaturas en general.

22. Realice un programa que calcule y visualice el valor máximo, el valor mínimo y el promedio de n números (n>0). El valor de n se solicitará al principio del programa y los números serán introducidos por el usuario. Realice dos versiones del programa, una usando el bucle “while” y otra con el bucle “do - while”.

23. Realizar un programa que pida dos números enteros positivos por teclado y muestre por pantalla el siguiente menú:

```
MENU
1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. Salir
Elija opción:
```

El usuario deberá elegir una opción y el programa deberá mostrar el resultado por pantalla y luego volver al menú. El programa deberá ejecutarse hasta que se elija la opción 5. Tener en cuenta que, si el usuario selecciona la opción 5, en vez de salir del programa directamente, se debe mostrar el siguiente mensaje de confirmación: ¿Está seguro que desea salir del programa (S/N)? Si el usuario selecciona el carácter ‘S’ se sale del programa, caso contrario se vuelve a mostrar el menú.

24. Escriba un programa que lea 20 números. Si el número leído es igual a cero se debe salir del bucle y mostrar el mensaje "Se capturó el numero cero". El programa deberá calcular y mostrar el resultado de la suma de los números leídos, pero si el número es negativo no debe sumarse. Nota: recordar el uso de la sentencia break.

25. Escriba un programa que lea números enteros. Si el número es múltiplo de cinco debe detener la lectura y mostrar la cantidad de números leídos, la cantidad de números pares y la cantidad de números impares. Al igual que en el ejercicio anterior los números negativos no deben sumarse. **Nota: recordar el uso de la sentencia break.**

26. Realizar un programa que simule el funcionamiento de un dispositivo RS232, este tipo de dispositivo lee caracteres enviados por un sensor. Las lecturas se realizan de a 5 caracteres (buffer) por vez, los cuales deben llegar con un formato fijo: el primer carácter tiene que ser X y el último tiene que ser 0.
Las secuencias leídas que respeten el formato se consideran correctas, la secuencia especial "&&&&&" marca el final de los envíos (llamémosla FDE), y toda secuencia distinta de FDE que no sea correcta se considera inválida. Al finalizar el proceso, se imprime un informe indicando la cantidad de lecturas correctas e inválidas recibidas. Para resolver el ejercicio deberá investigar cómo se utilizan las siguientes funciones de Java **Substring() y Length().**

27. Simular la división usando solamente restas. Dados dos números enteros mayores que uno, realizar un algoritmo que calcule el cociente y el residuo usando sólo restas. Método: Restar el dividendo del divisor hasta obtener un resultado menor que el divisor, este resultado es el residuo, y el número de restas realizadas es el cociente. Por ejemplo: 50 / 13:

    50 – 13 = 37 una resta realizada
    37 – 13 = 24 dos restas realizadas
    24 – 13 = 11 tres restas realizadas
    dado que 11 es menor que 13, entonces: el residuo es 11 y el cociente es 3.

    ¿Aún no lo entendiste? Consultá acá.

28. Realice un programa para que el usuario adivine el resultado de una multiplicación entre dos números generados aleatoriamente entre 0 y 10. El programa debe indicar al usuario si su respuesta es o no correcta. En caso que la respuesta sea incorrecta se debe permitir al usuario ingresar su respuesta nuevamente. Para realizar este ejercicio investigue como utilizar la función **Math.random()** de Java.

29. Escribir un programa que lea un número entero y devuelva el número de dígitos que componen ese número. Por ejemplo, si introducimos el número 12345, el programa deberá devolver 5. Calcular la cantidad de dígitos matemáticamente utilizando el operador de división. **Nota: recordar que las variables de tipo entero truncan los números o resultados.**

30. Crear un programa que dibuje una escalera de números, donde cada línea de números comience en uno y termine en el número de la línea. Solicitar la altura de la escalera al usuario al comenzar. Ejemplo: si se ingresa el número 3:

    1
    12
    123

31. Necesitamos mostrar un contador con 3 dígitos (X-X-X), que muestre los números del 0-0-0 al 9-9-9, con la particularidad que cada vez que aparezca un 3 lo sustituya por una E. Ejemplo:

0-0-0
0-0-1
0-0-2
0-0-E
0-0-4
.
.
0-1-2
0-1-E

Nota: investigar función contains() y replace() de Java

32. Dibujar un cuadrado de N elementos por lado utilizando el carácter "*". Por ejemplo, si el cuadrado tiene 4 elementos por lado se deberá dibujar lo siguiente:

* * * *
* *
* *
* * * *

33. Se dispone de un conjunto de N familias, cada una de las cuales tiene una M cantidad de hijos. Escriba un programa que pida la cantidad de familias y para cada familia la cantidad de hijos para averiguar la media de edad de los hijos de todas las familias.

34. Realizar un programa que lea 4 números (comprendidos entre 1 y 20) e imprima el número ingresado seguido de tantos asteriscos como indique su valor. Por ejemplo:

5 *****
3 ***
11 **********
2 **

35. Crea una aplicación que le pida dos números al usuario y este pueda elegir entre sumar, restar, multiplicar y dividir. La aplicación debe tener una función para cada operación matemática y deben devolver sus resultados para imprimirlos en el main.

36. Crea una aplicación que a través de una función nos convierta una cantidad de euros introducida por teclado a otra moneda, estas pueden ser a dólares, yenes o libras. La función tendrá como parámetros, la cantidad de euros y la moneda a pasar que será una cadena, este no devolverá ningún valor y mostrará un mensaje indicando el cambio (void).

El cambio de divisas son:
* 0.86 libras es un 1 €
* 1.28611 $ es un 1 €
* 129.852 yenes es un 1 €

37. Diseñe una función que pida el nombre y la edad de n personas e imprima los datos de las personas ingresadas por teclado e indique si son mayores o menores de edad. El método debe preguntarle al usuario si quiere seguir mostrando personas y frenar cuando el usuario ingrese la palabra "No".

38. Crea una aplicación que nos pida un número por teclado y con una función se lo pasamos por parámetro para que nos indique si es o no un número primo, debe devolver true si es primo sino false.
Un número primo es aquel solo puede dividirse entre 1 y si mismo. Por ejemplo: 25 no es primo, ya que 25 es divisible entre 5, sin embargo, 17 si es primo.

¿Qué son los números primos?

VER VIDEO: Vectores y Matrices en Java

39. Realizar un algoritmo que rellene un vector con los 100 primeros números enteros y los muestre por pantalla en orden descendente.

40. Realizar un algoritmo que calcule la suma de todos los elementos de un vector de tamaño N, con los valores ingresados por el usuario.

41. Escriba un programa que averigüe si dos vectores de N enteros son iguales (la comparación deberá detenerse en cuanto se detecte alguna diferencia entre los elementos).

42. Recorrer un vector de N enteros contabilizando cuántos números son de 1 dígito, cuántos de 2 dígitos, etcétera (hasta 5 dígitos).

43. Realizar un algoritmo que rellene un vector de tamaño N con valores aleatorios y le pida al usuario un numero a buscar en el vector. El programa mostrará donde se encuentra el numero y si se encuentra repetido

44. Crear una función rellene un vector con números aleatorios, pasándole un arreglo por parámetro. Después haremos otra función o procedimiento que imprima el vector.

45. Los profesores del curso de programación de Egg necesitan llevar un registro de las notas adquiridas por sus 10 alumnos para luego obtener una cantidad de aprobados y desaprobados. Durante el periodo de cursado se obtienen 4 notas, 2 por trabajos prácticos evaluativos y 2 por parciales. Las ponderaciones de cada nota son:

Primer trabajo práctico evaluativo 10%

Segundo trabajo práctico evaluativo 15%

Primer Integrador 25%

Segundo integrador 50%

Una vez cargadas las notas, se calcula el promedio y se guarda en el arreglo. Al final del programa los profesores necesitan obtener por pantalla la cantidad de aprobados y desaprobados, teniendo en cuenta que solo aprueban los alumnos con promedio mayor o igual al 70% de sus notas del curso.

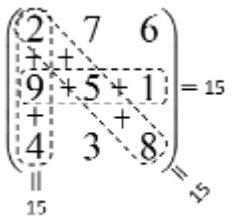
46. Realizar un programa que rellene un matriz de 4 x 4 de valores aleatorios y la muestre ordenada por sus columnas.

47. Realizar un programa que rellene una matriz de tamaño NxM con valores aleatorios y muestre la suma de sus elementos.

48. Realice un programa que compruebe si una matriz dada es anti simétrica. Se dice que una matriz A es anti simétrica cuando ésta es igual a su propia traspuesta, pero cambiada de signo. Es decir, A es anti simétrica si $A = -A^T$. La matriz traspuesta de una matriz A se denota por A^T y se obtiene cambiando sus filas por columnas (o viceversa).

¿Cómo es la transpuesta de una matriz?

49. Un cuadrado mágico 3 x 3 es una matriz 3 x 3 formada por números del 1 al 9 donde la suma de sus filas, sus columnas y sus diagonales son idénticas. Crear un programa que permita introducir un cuadrado por teclado y determine si este cuadrado es mágico o no. El programa deberá comprobar que los números introducidos son correctos, es decir, están entre el 1 y el 9.



50. Dadas dos matrices cuadradas de números enteros, la matriz M de 10x10 y la matriz P de 3x3, se solicita escribir un programa en el cual se compruebe si la matriz P está contenida dentro de la matriz M. Para ello se debe verificar si entre todas las submatrices de 3x3 que se pueden formar en la matriz M, desplazándose por filas o columnas, existe al menos una que coincida con la matriz P. En ese caso, el programa debe indicar la fila y la columna de la matriz M en la cual empieza el primer elemento de la submatriz P.

51. Construya un programa que lea 5 palabras de mínimo 3 y hasta 5 caracteres y, a medida que el usuario las va ingresando, construya una "sopa de letras para niños" de tamaño de 20 x 20 caracteres. Las palabras se ubicarán todas en orden horizontal en una fila que será seleccionada de manera aleatoria. Una vez concluida la ubicación de las palabras, rellene los espacios no utilizados con un número aleatorio del 0 al 9. Finalmente imprima por pantalla la sopa de letras creada.

Nota: Para resolver el ejercicio deberá investigar cómo se utilizan las siguientes funciones de Java substring(), Length() y Math.random().

52. Realizar un programa que complete un vector con los N primeros números de la sucesión de Fibonacci. Recordar que la sucesión de Fibonacci es la sucesión de los siguientes números:

1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Donde cada uno de los números se calcula sumando los dos anteriores a él. Por ejemplo:
La sucesión del número 2 se calcula sumando (1+1)
Análogamente, la sucesión del número 3 es (1+2),
Y la del 5 es (2+3),
Y así sucesivamente...
La sucesión de Fibonacci se puede formalizar de acuerdo a la siguiente fórmula:

$Fibonacci_n = Fibonacci_{n-1} + Fibonacci_{n-2}$ para todo $n>1$
$Fibonacci_n = 1$ para todo $n<=1$

Por lo tanto, si queremos calcular el término "n" debemos escribir una función que reciba como parámetro el valor de "n" y que calcule la serie hasta llegar a ese valor.

Para conocer más acerca de la serie de Fibonacci consultar el siguiente link: https://quantdare.com/numeros-de-fibonacci/