INFORMÁTICA: ***automatización de la información***. *[información de entrada]-(Proceso)→[salida].*

PROGRAMACIÓN: ***secuencia*** *de instrucciones, una detrás de otra, para conseguir la salida deseada.*

PROGRAMACIÓN ESTRUCTURADA: *hay sentencias que puede* ***alterar el orden*** *normal de ejecución.*

PROGRAMACIÓN ORIENTADA A OBJETOS: *la información es tratada como* ***clases*** *y objetos de esas clases.*

JAVA: *lenguaje de programación* ***estructurado*** *y* ***orientado a objetos***. *No es compilado, se pasa de los archivos ".java" con el código fuente a los pre-compilados ".class" con los bytecodes entendibles por la Maquina Virtual de Java. En java normalmente el nombre de las variables empieza por una* ***minúscula***, *esta desaconsejado usar en todo lo que no sea cadena de texto la "ñ" y las tildes. En principio se usará solo la clase "Integer" para valores* ***enteros*** *y la clase "String" para* ***cadenas*** *de texto. Hay que* ***definir*** *las variable antes de usarlas, diciendo el tipo o clase y opcionalmente se le puede dar un valor inicial. Una variables solo será* ***visible*** *y se podrá usar en el bloque de instrucciones donde fue definida.*

```
public class Hola {
    public static void main(String[] args) {
    Integer x=5;    // Comentario de una sola linea
    Integer y;
    String txt="Hola";
    String txt=null;
    ...
    <instrucción>;
    <instrucción>;
    ...

System.out.println(txt+" que tal. x="+x);
    /* Comentario varias líneas empieza con barra
    asterisco y termina con asterisco barra */
    }
}
```

Una sentencia puede ser solo una instrucción o un bloque de instrucciones {instrucción1; instrucción2; instrucción3; ...}

Los operadores ***aritméticos*** *son: "+","-","*","/" y "%" resto.* ***Relacionales****:">","<","==" y "!=" distinto. En las* ***cadenas*** *hay que usar "s1.equal(s2)" en vez de "s1==s2" y "!s1.equal(s2)" en vez de "s1!=s2". Op.* ***Lógicos****: y → "&&", o → "||", no → "!". Una expresión lógica o* ***condición*** *puede ser "true" o "false" por Ej. "x>=5" o "y==6" y pueden ser* ***compuestas*** *como por ejemplo "x<=9 && x>6" o "x==5 || y>7".*

ESTRUCTURAS DE CONTROL: *permiten alterar el orden normal de ejecución. Y pueden* ***anidarse****.*

- *"****Si****" sintaxis "****if (<Condición>) <Sentencia>****" se ejecuta la sentencia si se la condición es verdadera.*
- *"****Sino****" sintaxis "****if (<Condición>) <Sentencia_A> else <Sentecia_B>****"...si es falsa ejecuta "<B>".*
- *"****Bucle****" sintaxis "****for (<IniciaVariable>;<Condición>;<CadaIteración>) <Sentencia>****" mientras la condición sea verdadera ejecuta: "<IniciaVariable>" solo antes de la primera iteración, ejecuta "<Sentencia>", después de cada iteración ejecutará "<CadaIteración>" (en este lugar se suele modificar una variable que afecte a "<Condición>") y si la condición sigue siendo verdadera vuelve a hacerlo.*

FALLOS: *para evitar que un programa en el que se produzca un fallo cuando se está ejecutando sea abortado directamente por el sistema operativo perdiendo información, etc. En java se intentan gestionar todos los fallos posibles. Para ello ser rodea las instrucciones propensas a dar fallos en el momento de ejecución con:*
try <Sent.PosibleFallo> catch (<TipoExcepción>) <Sent.GestionFallo> finally <Sent.FalloONo>
Pudiendo haber varias "catch" una para cada tipo de excepción.

ENTRADA POR TECLADO: *permite introducir una cadena de texto por teclado. (Puede haber un fallo de teclado)*

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Hola {
    public static void main(String[] args) {
        String linea=""; // para leer por teclado quedando almacenado en la variable linea
        System.out.print("Intruduzca un número:"); // print sin 'ln' final no salta a la siguiente linea
        try{ // este bloque debe estar donde queramos leer por teclado
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            linea = br.readLine();
        }catch(Exception e){ e.printStackTrace();}
        // ahora la variable linea contendrá el valor introducido por teclado
        …
    }
}
```

CONVERSIÓN DE CLASES: *(Puede haber fallo, por ejemplo al convertir letras en un entero)*

```
Integer i=43;
String s="";
s=i.toString();
```

```
String s="43";
int i=0;
i=Integer.parseInt(s);
```

MÉTODO: *es parte del código, que por realizar una tarea que se hace varias veces, por organización o por otro motivo se agrupa dándole un nombre:* ***<TipoDevuelto> <Nombre> (<Parámetros>) <Sentencia>***

Donde "<TipoDevuelto>" es la clase o tipo ("void" si no devuelve nada) del valor devuelto usando "return <Expresion>;". Y "<Parámetros>" serán la definición de las variable que contendrá el valor pasado. El método será llamado cada vez que se quiera ejecutar ese código: ***<Nombre> (<Expresiones>);***

Hacer un programa que …

1. *Escriba "Hola mundo".*
2. *Almacene en una variable de cadena de texto "Hola mundo" y lo escriba.*
3. *Almacene en una variable entero el valor de 5 y escriba el valor de la variable x.*
4. *Almacene en una variable entero el valor de 5 y escriba "x*2=" y el valor de la variable x por 2.*
5. *Almacene en x un valor entero cualquiera (cambiarlo para probarlo) y escriba "aprobado" si x>=5.*
6. *Almacene en x un valor entero cualquiera y escriba "notable" si x>=7 y x<=8.*
7. *Almacene en x un valor entero cualquiera y escriba "aprobado" si x>=5, o "suspenso" sino.*
8. *Almacene en x un valor y escriba "suficiente" → 5, "bien" → 6, "notable" → 7o8, "sobre" → 9o10.*
9. *Cuente de 1 a 10, escribiendo "x=" y valor de la variable x para cada iteración (1,2,3,4...10).*
10. *Cuente de 10 a 1, escribiendo "x=" y valor de la variable x para cada iteración (10,9,8,7...1).*
11. *Cuente de 0 a 9 con incrementos de 3, escribiendo el valor de x para cada iteración (0,3,6 y 9).*
12. *Cuente de 1 a 10, escribiendo el valor x solo si es mayor que 5 (6,7,8,9 y 10).*
13. *Cuente de 1 a 10, escribiendo el valor x solo si x>5 y x<9 (6,7 y 8).*
14. *Cuente de 1 a 10, escribiendo el valor x solo si x<4 o x>8 (1,2,3,9 y 10).*
15. *Cuente de 1 a 10, escribiendo el valor x solo si es múltiplo de 3 (resto de dividir por 3 es 0) (3,6 y 9).*
16. *Represente la tabla de multiplicar del 2.*
17. *Cuente de 1 a 10, escribiendo "suficiente" → 5, "bien" → 6, "notable" → 7o8, "sobre" → 9o10.*
18. *Lea desde teclado un nombre y escriba "Hola " seguido del nombre introducido.*
19. *Lea por teclado un entero y diga "suficiente" → 5, "bien" → 6, "notable" → 7o8, "sobre" → 9o10.*
20. *Defina y utilice un método que al pasarle un valor entero escriba la tabla de multiplicar.*
21. *Lea desde teclado un número entero y llame al método definido en el ejercicio anterior con ese valor.*

Hacer un programa que …

1. *Defina un método que devuelva el doble del valor que se le pase como parámetro. El programa debe solicitar un número por teclado, llamar al método e imprimir el resultado.*

2. *Defina un método que devuelva la cadena "Hola <nombre>" siendo <nombre> una cadena pasada como parámetro. El programa debe solicitar un nombre por teclado, llamar al método e imprimir el resultado.*

3. *Lea por teclado dos números enteros y se los pase a un método que devuelva el mayor de los dos (o cualquiera si son iguales), por último el programa debe escribir en pantalla "el mayor es...".*

4. *Lea por teclado tres números enteros y se los pase a un método que devuelva el mayor de los tres (o uno de los mayores si hay iguales), por último el programa debe escribir en pantalla "el mayor es...".*

5. *Defina un método que devuelva la cadena "<cadena>, " repetida <n> veces, siendo <cadena> una cadena pasada como parámetro y <n> un valor entero pasado como parámetro. El programa debe solicitar una cadena y un número entero, llamar al método e imprimir el resultado.*

6. *Defina un método que compruebe si el valor entero pasado como parámetro esta comprendido entre 1 y 10, y si es así, devuelva una cadena con la secuencia de números desde el pasado como parámetro hasta el 10 (ej. "4,5,6,7,8,9,10"). Si el valor pasado como parámetro no estuviera entre 1 y 10, que devuelva la cadena "fuera de rango". El programa debe permitir al usuario introducir un número por teclado e imprimir el resultado.*

7. *Lea por teclado un número "n", a continuación pida n-veces un numero y por último imprima cuantos números se han introducido, la suma de todos los número introducidos, el número mayor de todos los introducidos y el menor de todos los introducidos.*

8. *Definir un método que lea por teclado un número "n", a continuación pida n-veces un valor numérico que sera una nota de clase una vez introducidas la n-notas devolverá la media. Definir otro método que pasándole una nota devuelva una cadena que represente dicha nota numérica con letras. El programa deberá llamar primero al método que genera la media y con el valor devuelto llamar al segundo método y escribir en pantalla lo que devuelve el segundo método.*

9. *Defina un método que vaya pidiendo números por teclado hasta que la suma de todos los números introducidos sea igual o mayor que 10, momento en el que devolverá dicha suma. El programa debe llamar al método e imprimir por pantalla el valor devuelto.*

10. *Defina dos métodos llamados "imprimir", los dos métodos se llamarán igual pero uno tendrá como parámetro un entero y el otro tendrá como parámetro una cadena. Lo único que harán será imprimir en pantalla "El número es X" para el método que acepta un entero y "La cadena es 'X'" para el método que acepta una cadena. Donde la "X" será el valor del parámetro pasado. El programa debe hacer uso de los dos métodos.*

11. *Calcule el valor factorial (x!=x*(x-1)*(x-2)*(x-3)...hasta que x=1) primero mediante un bucle con un acumulador y después con un método que se llame a si mismo*

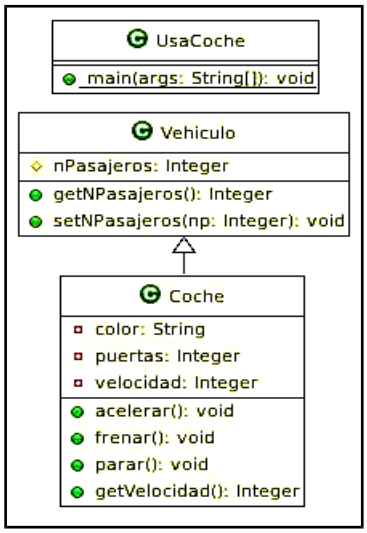
INTRODUCCIÓN: *Cuando se desarrolla una aplicación normalmente se hace para realizar una tarea que ya se realiza en el mundo real. Y lo que se intenta es* ***modelar*** *(crear una representación del mundo real dentro de la computadora) la* ***información*** *y los* ***procesos*** *de esta tarea en el ordenador. Hasta que llegó la POO la información iba por un lado y los procesos que se aplicaban a ella iban por otro lado. Con la POO la información va a tener asociada los procesos. A esta unión se le llama* ***clase****. Cuando se* ***define una clase*** *se definirá la información contenida por dicha clase (a cada variable que contendrá parte de dicha información se le llamará* ***atributo****) y los procesos (o* ***métodos****) que podemos realizar sobre esa información. Cuando quiero usar una clase debo crear una variable de esa clase, a esta variable donde ya puedo almacenar información se llama objeto. Para poder* ***usar un objeto*** *primero debo crearlo a partir de una clase. La POO se usa por que esta forma de trabajar se acerca más al mundo real. Lo que* ***diferencia*** *a un objeto de otro son sus valores de los atributos.*

CONCEPTOS BÁSICOS DE LA POO:

Herencia*: esta consiste en que se puede crear una subclase que herede (los atributos y métodos) de otra clase.*
Polimorfismo*: puede haber varios métodos con un mismo identificador pero diferentes parámetros.*
Encapsulación*: cada clase debe recoger todos los aspectos relacionados con una entidad en el mundo real, siendo cada entidad una parte bien diferenciada y más o menos independiente de las demás partes.*
Ocultación*: cada clase debe ser accesible solo en aspecto necesario, ocultando los demás atributos y métodos.*

JAVA: *lenguaje de programación* ***estructurado*** *y* ***orientado a objetos****.*

En java siempre se trabaja con ***clases y objetos****. Cuando se crea una aplicación lo primero será crear una clase y a esta se le suele definir un método especial llamado "****main****" que será ejecutado cuando se intente ejecutar la clase. Por este motivo el* ***punto de partida*** *de la aplicación será siempre el método "main" de la clase principal. Una aplicación puede definir varias clases y manejar varios objetos para cada clase.*
Cuando manejamos un objeto se hará ***referencia*** *primero al nombre con el que se creo (o* ***identificador****) seguido de un* ***punto*** *y a continuación el atributo o método al que queramos referirnos.*
*En java los atributos y métodos puede llevar delante un modificador "****static****", este indica que ese atributo o método puede ser usado directamente a través del nombre de la clase sin tener que crear ningún objeto. De esta forma serán* ***atributos o métodos de la clase*** *y no de un objeto como suele ser lo normal (sin modificador). Por este motivo, por ejemplo, el método "main" puede ser ejecutado sin haber creado un objeto de la clase principal que lo contiene.*




```
public class UsaCoche {
  public static void main(String[] args) {
    Coche coche1 = new Coche();
    coche1.acelerar();
    coche1.setNPasajeros(3);
    System.out.println(
      "Velocidad="+coche1.getVelocidad()+
      " Pasajeros="+coche1.getNPasajeros());
  }
}
class Vehiculo {
  protected Integer nPasajeros;
  public Integer getNPasajeros()
    {return this.nPasajeros;}
  public void setNPasajeros(Integer np)
    {this.nPasajeros=np;}
}
class Coche extends Vehiculo{
  private String color;
  private Integer puertas;
  private Integer velocidad;
  public void acelerar(){this.velocidad++;}
  public void frenar(){this.velocidad--;}
  public void parar(){this.velocidad=0;}
  public Integer getVelocidad(){return this.velocidad;}
}
```

Además tenemos unos modificadores de acceso utilizados para los atributos y métodos que son:

- *"public": que lo hará visible al resto de clases.*
- *"private": visible solo desde el interior de la clase.*
- *"protected": visible desde dentro de la clase y subclases que hereden de ella.*

*Si una clase hereda de otra clase se definirá como "****class Coche extends Vehiculo****" de esta manera un objeto coche tendrá también los atributos y métodos de vehículo. Si creáramos una clase bicicleta que heredara de vehículo esta también tendría esos atributos y métodos. Existe una clase llamada* ***Vector*** *que contendrá varios objetos ordenados por un índice de una misma clase. Crea un objeto "****Vector<Coche> coches = new Vector<Coche>();****". "****coches.add(coche1);****" añadirá el objeto coche1 al vector, "****coches.size()****" dará el número de elementos y "****coches.get(3)****" dará el cuarto elemento. Hay un recolector de basura (objetos en memoria que ya no se utilizan) que comprueba que objetos han perdido todas las referencias hacia él y los elimina. Una clase puede contener uno o varios* ***constructores****, es decir un método que se llame igual que la clase que*

se ejecutará al crear un nuevo objeto.

Crear una clase ejecutable en un archivo que además …

1. *Defina una clase "Coche" con los atributos: color y año de fabricación. Y cree dos coches "cocheA" y "cocheB", de color rojo y verde respectivamente, y del año 2009 y 2010 respectivamente. Por último debe mostrar la información de cada coche por pantalla.*
2. *Defina una clase "Vehiculo" con el atributo número de pasajeros y haga que la clase del ejercicio anterior herede de esta. Además crear una clase "Bicicleta" que también heredará de la clase "Vehiculo" con los atributos: número de cambios y peso, creando los objetos "biciA" y "biciB" dándolo respectivamente los valores: 3,20; y 12, 33.*
3. *Defina una clase "Periferico" con los atributos: puerto y nombre. De la que herede una clase "Ratón" con el atributo número de botones. También herede otra clase "Escaner" con el atributo resolución. También herede otra clase "Monitor" con el atributo número pulgadas de la pantalla. Y cree al menos 2 objetos para cada subclase presentando los datos.*
4. *Pregunte varias cadenas de texto por teclado hasta que se introduzca una cadena vacía (sin caracteres, directamente dándole a "intro"), vaya almacenando estas cadenas en un vector de cadenas y al final las imprima en pantalla una detrás de otro.*
5. *Defina una clase "Jugador" de fútbol con los atributos: nombre, edad y dorsal. Cree un vector de jugadores y añada al menos tres jugadores. E imprima por pantalla la información de los tres jugadores.*
6. *Copiar la clase anterior "Jugador" llamándola "Futbolista" y añadir un atributo llamado goles que contendrá el número de goles marcados por ese futbolista. Además el vector de futbolistas debe ser definido como estático en la clase "Futbolista". Y cree un método estático de la clase "Futbolista" que recorra el vector de futbolistas y sume todos los goles marcados por los futbolistas devolviendo el ese valor. Crear varios futbolistas e imprimir en pantalla el número de goles totales.*

7. *Crear una clase ejecutable y una clase "Bombilla". La clase "Bombilla" debe contener un atributo "encendida" y otro "vatios". Además la clase "Bombilla" tendrá un vector estático que contendrá todas las bombilla y un método que devuelva el consumo total.*
8. *Copiar la clase anterior "Bombilla" llamándola "Pc" y poner el atributos "encendida" como "private" crear (como "public") los métodos "getEncencidad()" para obtener el valor del atributo y "setEncencidad()" para fijar el valor del atributo. Hacer lo mismo con el atributo "vatios".*
9. *Copiar la clase anterior "Pc" llamándola "Tv". Crear un constructor sin parámetros que no haga nada. Y otro constructor con dos parámetros, un para asignar el valor inicial del atributo "encendida" y otro parámetro para asignar el valor inicial del atributo "vatios".*
10. *Con "Integer x = (new Random()).nextInt(11);" el objeto x tendrá un valor entero aleatorio entre 0 y 10. A partir de esto crear una clase "Corredor" con dos atributos "nombre" y "distancia". Crear un vector estático en la clase "Corredor" llamado "corredores". Crear un constructor con un solo parámetro para definir "nombre" pero que además ponga a "0" el atributo "distancia" y añada en nuevo corredor al vector "corredores". Esta clase "Corredor" además debe contener un método estático llamado "unSegundo" que incremente de forma aleatoria la distancia de cada objeto de la clase corredor dentro del vector "corredores" y otro método estático llamado "hayGanador" que devuelva el nombre del ganador si algún corredor superó la distancia de 100. En la clase principal crear varios corredores y definir un bucle que imprima los segundos transcurridos y ejecute "unSegundo" y si el valor devuelto por "hayGanador" no es nulo imprima las distancias de todos los corredores y termine.*

Autor: Ismael PONCE GORDILLO
Fecha: 11 de Octubre de 2016