**Unidad didáctica 3: PROGRAMACIÓN ORIENTADA A OBJETOS CON JAVA I**

# Modo de entrega

***Crear un proyecto en Eclipse con nombre UT0301, una vez acabados los ejercicios, iréis donde se guardan vuestros proyectos Java ( Si seguisteis la guía de instalación, se encuentra en C:\ProyectosEclipse). Seleccionareis la carpeta donde se guardan el proyecto que contienen los ejercicios y los añadiréis a un archivo zip o rar con el siguiente formato de nombre:***

***DAM-PRO-UD03-Apellido1\_Apellido2-Nombre.zip***

***DAM-PRO-UD03-García\_García-Luís Regino.zip***

# Ejercicio 1:

Crea una clase llamada Contador que contenga un único atributo entero llamado *cont*.

La clase tendrá los siguientes constructores:

* Constructor por defecto
* Constructor con parámetros para inicializar el contador con un valor no negativo. Si el valor inicial que se recibe es negativo el contador tomará el valor cero como valor inicial.
* Constructor copia.

Además de los métodos getter y setter, la clase contendrá los métodos:

* incrementar: incrementa el contador en una unidad.
* decrementar: decrementa el contador en una unidad. El contador nunca podrá tener un valor negativo. Si al decrementar se alcanza un valor negativo el contador toma el valor cero.

Una vez creada la clase, escribe un método main para probar la clase.

# Ejercicio 2:

A) Crea una clase Cuenta para representar una cuenta bancaria. Los datos de la cuenta son:

* nombre del cliente (String)
* número de cuenta (String)
* tipo de interés (double)
* saldo (double).

La clase contendrá los siguientes métodos:

* Constructor por defecto
* Constructor con todos los parámetros
* Constructor copia. Este método crea una cuenta a partir de otra ya existente.
* Métodos setters/getters para asignar y obtener los datos de la cuenta.
* Métodos ingreso y reintegro.
  + Un ingreso consiste en aumentar el saldo en la cantidad que se indique. Esa cantidad no puede ser negativa.
  + Un reintegro consiste en disminuir el saldo en una cantidad pero antes se debe comprobar que hay saldo suficiente. La cantidad no puede ser negativa.

Los métodos ingreso y reintegro devuelven true si la operación se ha podido realizar o false en caso contrario.

* Método transferencia que permita pasar dinero de una cuenta a otra siempre que en la cuenta de origen haya dinero suficiente para poder hacerla. Ejemplo de uso del método transferencia:

cuentaOrigen.transferencia(cuentaDestino, importe);

que indica que queremos hacer una transferencia desde cuentaOrigen a cuentaDestino del importe indicado.

B) Crea una clase principal donde:

* Crea un menú con las siguientes opciones
  + Crear cuenta
  + Mostrar saldo cuenta
  + Hacer ingreso en cuenta
  + Hacer transferencia
  + Mostrar datos de la cuenta
* Realizar las operaciones que se indican en el menú utilizando la clase cuenta creada en el apartado A

# Ejercicio 3:

A) Crea una clase llamada Libro que guarde la información de cada uno de los libros de una biblioteca. La clase debe guardar:

* el título del libro
* autor
* número de ejemplares del libro
* número de ejemplares prestados

La clase contendrá los siguientes métodos:

* Constructor por defecto.
* Constructor con parámetros.
* Métodos Setters/getters
* Método *préstamo* que incremente el atributo correspondiente cada vez que se realice un préstamo del libro. No se podrán prestar libros de los que no queden ejemplares disponibles para prestar. Devuelve true si se ha podido realizar la operación y false en caso contrario.
* Método *devolución* que decremente el atributo correspondiente cuando se produzca la devolución de un libro. No se podrán devolver libros que no se hayan prestado. Devuelve true si se ha podido realizar la operación y false en caso contrario.
* Método *toString* para mostrar los datos de los libros. Este método se heredada de Object y lo debemos modificar (override) para adaptarlo a la clase Libro.

B) Crea una clase principal para probar el funcionamiento de la clase Libro que contenga un menú que realice las siguientes acciones:

* Inserción de libros
* Mostrar libros
* Realizar préstamo
* Realizar devolución

# Ejercicio 4:

Crearemos una superclase llamada Electrodoméstico con las siguientes características:

* Sus atributos son precio base, color, consumo energético (letras entre A y F) y peso. Indica que se podrán heredar.
* Por defecto, el color será blanco, el consumo energético será F, el precioBase es de 100 € y el peso de 5 kg. Usa constantes para ello.
* Los colores disponibles son blanco, negro, rojo, azul y gris. No importa si el nombre está en mayúsculas o en minúsculas.
* Los constructores que se implementarán serán
  + Un constructor por defecto.
  + Un constructor con el precio y peso. El resto por defecto.
  + Un constructor con todos los atributos.
* Los métodos que implementara serán:
  + Métodos get de todos los atributos.
  + **comprobarConsumoEnergetico(char letra)**: comprueba que la letra es correcta, sino es correcta usara la letra por defecto. Se invocará al crear el objeto y no será visible.
  + **comprobarColor(String color):** comprueba que el color es correcto, sino lo es usa el color por defecto. Se invocará al crear el objeto y no será visible.
  + **precioFinal():** según el consumo energético, aumentara su precio, y según su tamaño, también. Esta es la lista de precios:

|  |  |
| --- | --- |
| **LETRA** | **PRECIO** |
| A | 100 |
| B | 80 |
| C | 60 |
| D | 50 |
| E | 30 |
| F | 10 |

|  |  |
| --- | --- |
| **TAMAÑO** | **PRECIO** |
| Entre 0 y 19 kg | 10 € |
| Entre 20 y 49 kg | 50 € |
| Entre 50 y 79 kg | 80 € |
| Mayor que 80 kg | 100 € |

Crearemos una subclase llamada Lavadora con las siguientes características:

* Su atributo es carga, además de los atributos heredados.
* Por defecto, la carga es de 5 kg. Usa una constante para ello.
* Los constructores que se implementarán serán:
  + Un constructor por defecto.
  + Un constructor con el precio y peso. El resto por defecto.
  + Un constructor con la carga y el resto de atributos heredados. Recuerda que debes llamar al constructor de la clase padre.
* Los métodos que se implementara serán:
  + Método get de carga.
  + precioFinal():, si tiene una carga mayor de 30 kg, aumentara el precio 50 €, sino es así no se incrementará el precio. Llama al método padre y añade el código necesario. Recuerda que las condiciones que hemos visto en la clase Electrodoméstico también deben afectar al precio.

Crearemos una subclase llamada Televisión con las siguientes características:

* Sus atributos son resolución (en pulgadas) y sintonizador TDT (booleano), además de los atributos heredados.
* Por defecto, la resolución será de 20 pulgadas y el sintonizador será false.
* Los constructores que se implementarán serán:
  + Un constructor por defecto.
  + Un constructor con el precio y peso. El resto por defecto.
  + Un constructor con la resolución, sintonizador TDT y el resto de atributos heredados. Recuerda que debes llamar al constructor de la clase padre.
* Los métodos que se implementara serán:
  + Método get de resolución y sintonizador TDT.
  + precioFinal(): si tiene una resolución mayor de 40 pulgadas, se incrementará el precio un 30% y si tiene un sintonizador TDT incorporado, aumentara 50 €. Recuerda que las condiciones que hemos visto en la clase Electrodoméstico también deben afectar al precio.

Ahora crea una clase ejecutable que realice lo siguiente:

* Crea 10 Electrodomésticos.
* Asigna a cada electrodomestico un objeto de las clases anteriores con los valores que desees.
* Ejecuta el método precioFinal() en los electrodomesticos.
* Deberás mostrar el precio de cada clase, es decir, el precio de todas las televisiones, por un lado, el de las lavadoras por otro y la suma de los Electrodomésticos (puedes crear objetos Electrodoméstico, pero recuerda que Televisión y Lavadora también son electrodomésticos). Recuerda el uso operador instanceof.

Por ejemplo, si tenemos un Electrodoméstico con un precio final de 300, una lavadora de 200 y una televisión de 500, el resultado final será de 1000 (300+200+500) para electrodomésticos, 200 para lavadora y 500 para televisión.

# Ejercicio 5:

A) Nos piden hacer una un programa que gestione empleados.

Los empleados se definen por tener:

* Nombre
* Edad
* Salario

También tendremos una constante llamada PLUS, que tendrá un valor de 300€

Tenemos dos tipos de empleados:

* Repartidor.
* comercial.

El comercial, aparte de los atributos anteriores, tiene uno más llamado comisión (double).

El repartidor, aparte de los atributos de empleado, tiene otro llamado zona (String).

Crea sus constructores, getters and setters y toString (piensa como aprovechar la herencia).

No se podrán crear objetos del tipo Empleado (la clase padre) pero si de sus hijas.

Las clases tendrán un método llamado plus, que según en cada clase tendrá una implementación distinta. Este plus básicamente aumenta el salario del empleado.

* En comercial, si tiene más de 30 años y cobra una comisión de más de 200 euros, se le aplicara el plus.
* En repartidor, si tiene menos de 25 y reparte en la “zona 3”, este recibirá el plus.
* Puedes hacer que devuelva un booleano o que no devuelva nada, lo dejo a tu elección.

B) Crea una clase ejecutable donde crees distintos empleados y le apliques el plus para comprobar que funciona.