Centro de Servicios y Gestión Empresarial	ACTIVIDAD:	Aplicar Programacion Orientada a Objetos		
	PROGRAMA:		COMPETENCIA:	
	FICHA:		INSTRUCTOR:	
	FECHA:		APRENDIZ:	

EJERCICIO # 1

Crear **una clase llamada Persona** que siga las siguientes condiciones:

Sus atributos son:

- documento (String)
- nombre (String)
- edad (byte)
- sexo (M masculino, F femenino) (char)
- peso(int)
- altura (double)

No queremos que se accedan directamente a ellos. Piensa que modificador de acceso es el más adecuado, también su tipo. Si quieres añadir algún atributo, método puedes hacerlo.
Por defecto, todos los atributos serán valores por defecto según su tipo, ejemplo las variables int = 0, double=0.0, String =null, etc.).
El sexo por defecto será M.
Se implantaran varios constructores:

- Un constructor por defecto que no recibe ni tiene ninguna funcionalidad dentro de él.
- Un constructor con el documento, nombre, edad y sexo
- Un constructor con todos los atributos como parámetro.

Los métodos que se implementaran son:

- **calcularIMC()**: El método debe calcular y devolver la condición de peso de la persona Para calcular el IMC (Índice de masa corporal) tenga en cuenta lo siguiente:

$$IMC=\frac{\text{Peso (kg)}}{\text{Estatura}^2\text{(Mts.)}} \quad \frac{120\ kg}{1.65 \times 1.65} = \frac{120\ kg}{2.7225} = 44$$

  IMC = **peso / ((altura/100)^2)**

  Se ha establecidos las siguientes consideraciones para determinar la condición de peso de la persona, según su IMC, es decir, si la persona tiene IMC menor a 18 está **bajo de peso**, si el IMC está entre 18 y 24 entonces esta **normal de peso**, si el IMC está entre 25 y 27 entonces la persona tiene **sobre peso**, de lo contrario la persona esta **obeso**
- **esMayorDeEdad()**: indica si es mayor de edad, devuelve un booleano.
- **comprobarSexo(char sexo):** comprueba que el sexo introducido es correcto. Si no es correcto, será M.
- **listarInformacion()**: devuelve toda la información del objeto.

Ahora, crea una clase ejecutable (Con método main) y dentro de este metodo haga lo siguiente:

Centro de Servicios y Gestión Empresarial	ACTIVIDAD:	Aplicar Programacion Orientada a Objetos		
	PROGRAMA:		COMPETENCIA:	
	FICHA:		INSTRUCTOR:	
	FECHA:		APRENDIZ:	

1. Pide por teclado el documento, nombre, la edad, sexo, peso y altura.
2. Crea un objetos (obj1) de la clase Persona:
   El objeto debe recibir los datos anteriores por medio del constructor.
3. Muestre toda la informacion del obj1.

4. Pide por teclado el documento, nombre, la edad, sexo, peso y altura.
5. Crea otro objetos (obj2) de la clase Persona:

   El objeto recibirá el documento, nombre, edad y sexo por medio del constructor.
   El segundo objeto debe recibir el peso y la altura por medio de sus atributos (Set).

Muestre toda la informacion del obj2.

EJERCICIO # 2

Crea el proyecto llamado **IntroPOO** y crear una clase llamada **Producto** dentro de un paquete llamado "**productos**" y siga las siguientes condiciones:

Sus atributos son:

- codigo (int)
- nombre (string)
- categoria (String)
- valor (int)
- descuento (double)
- cantidad (byte)
- estado (boolean)

No queremos que se accedan directamente a los atributos (**encapsulamiento**). Piensa que modificador de acceso es el más adecuado y lo que se debe implementar para garantizar lectura y escritura de los atributos.
Por defecto, todos los atributos serán valores por defecto según su tipo (0 números (enteros y double), cadena vacía (String), etc.), excepto el **estado** que será un atributo **public static,** cuyo valor inicial es **true**.
Se implementaran varios constructores:

- Un constructor por defecto que no recibe, ni tiene ninguna funcionalidad dentro de él.
- Un constructor con el código, nombre y categoría.
- Un constructor con todos los atributos como parámetro, **menos** el estado y el descuento.

Los métodos que se definirán son:

Centro de Servicios y Gestión Empresarial	ACTIVIDAD:	Aplicar Programacion Orientada a Objetos		
	PROGRAMA:		COMPETENCIA:	
	FICHA:		INSTRUCTOR:	
	FECHA:		APRENDIZ:	

- Métodos **get** y **set** para los atributos código, nombre, categoría, valor y cantidad.
- Métodos **get** para el atributos **descuento.** El atributo **estado** no tendrá get ni set ¿Explique cuál sería la razón por la cual no lo requiere?
- **calcularVenta( )**: devuelve el valor de la venta (cantidad por el valor del producto). Su modificador de acceso no debe permitir que sea accesible desde otra clase.
- **calcularDescuento( )**: es un método instanciaciable y sin retorno, que calculara el descuento del 5% sobre el valor de la venta (***cantidad*** * ***valor***), si la cantidad es menor a 10, si la cantidad esta entre 10 y 30 calculara el descuento del 10%, de lo contrario calculara el descuento del 15% para el producto y asignar el valor obtenido en cada caso al ***atributo descuento*.**
- **calcularDescuento**(***byte*** *valorDescuento***)**: es un método instanciaciable y sin retorno, el cual recibe el parámetro ***valorDescuento***, es decir, un numero entero entre 0 – 60 y calculara el descuento sobre el valor de la venta (***cantidad*** * ***valor***), teniendo en cuenta el porcentaje según el parámetro y establece el valor obtenido al ***atributo descuento*.**
- **calcularTotal( ):**es un método accesible desde cualquier clase, el cual llama al método **calcularVenta( )** y le resta el valor de descuento y **retorna** el resultado de la resta;
- **cambiarCategoria(String** cat**)**: su *modificador de acceso* debe ser ***public static***, y recibe el nombre de una categoría como ***parámetro*** (cat) y se lo asigna al ***atributo*** categoría.

  Al crear este método se va a generar un ***error*** en el código; usted debe identificar la ***causa*** y las posibles ***soluciones*** para corregir el error. Realice los ajustes necesarios e implemente una de las posibles soluciones.

  ¿El parámetro (**cat**) del método podría **nombrarse** igual que el atributo de la clase?

  ¿Cuándo se ejecute este método desde otra clase, es posible cambiar el valor de la categoría a todos los objetos que pertenezca a esta clase, es decir, a todos los objetos que se hayan creado de la clase Producto?
- **listarInformacion( )**: método instanciable que devuelve toda la información del objeto en una cadena de texto, incluyendo el valor de descuento y el total calculado.

  ¿Se podría llamar o invocar el método calcularDescuento en este método?

Ahora, crea una ***clase ejecutable*** *(con método main) llamada* ***AdminProducto*** *dentro de un paquete llamado "****productos****" y dentro del método main, realice de forma secuencial las siguientes sentencias*:

1. Ingresar por teclado el código, nombre, la categoría y guarde estos valores en variables.
2. Crea un objeto (obj1) de la clase Producto, asignando estos valores (poblar) por medio del constructor más indicado.
3. Ingresar por teclado el valor y la cantidad; guarde estos valores en variables y asigne estos valores al objeto anterior (obj1) por medio de los métodos **set**.
4. Invocar o llamar el método **calcularDescuento( )** del objeto (obj1).

Ing. E.M.M

¿Qué sucedería si no se llamara a este método en este punto?

5. Muestre la informacion del objeto (obj1), invocando o llamando el método **listarInformacion( )**
6. Ejecute el programa y analice los resultados obtenidos hasta aquí.
7. Ingresar por teclado un numero entero correspondiente a un nuevo valor de un producto y cambie el valor de producto (obj1)
8. *Ingresar por teclado el código, nombre, categoría, valor y cantidad y guarde estos valores en variables.*
9. Crea una instancia (obj2) de la clase Producto, entregando sus valores (poblar) por medio del constructor más indicado.
10. Calcule el descuento para el objeto (obj2).
11. Mostrar la información de cada objeto. Por ejemplo:

Código: **5845**
Producto: **Arroz**
Categoría: **Granos**
Estado: **true**
Valor: **1500**
Cantidad: **20**
Descuento: **3000**
Total: **27000**

Verifique que la informacion que se muestra si corresponde con los valores ingresados y calculados.

12. Ingresar por teclado el valor entero del porcentaje de descuento (valor entre 0 y 60) y asigne este valor de descuento al producto (obj1, obj2) de mayor valor, mediante el método **calcularDescuento(byte valor)**.
13. Mostrar la información del producto de mayor valor.
14. Ingresar por teclado el valor del estado (**false** o **true**) y asigne el nuevo estado a los objetos (obj1, obj2), mediante la **clase Producto** es decir, cambie el valor del atributo estado a la clase.
15. Mostrar la información de cada objeto. Observe el estado actual y anterior de los objetos.
16. Ingresar por teclado el nombre de una categoría de producto y asigne la nueva categoría al producto (obj1), mediante el método **set**
17. Mostrar la información de cada objeto1
18. Ingresar por teclado el nombre de otra categoría de producto y asigne la nueva categoría, utilizando el método **cambiarCategoria(String Cat)**

¿Cuál es la diferencia entre la instrucción ejecutada en el numeral **16** y **18**?

19. Muestre la categoría de los objetos (obj1, obj2)

Centro de Servicios y Gestión Empresarial	ACTIVIDAD:	Aplicar Programacion Orientada a Objetos		
	PROGRAMA:		COMPETENCIA:	
	FICHA:		INSTRUCTOR:	
	FECHA:		APRENDIZ:	

EJERCICIO # 3

Crear un paquete llamado "**artículos**" y dentro de este paquete crea la clase llamada **Categoría**, la cual debe tener la siguiente estructura:

Categoria		
private	codigo	int
private	nombre	String
private	Descripcion	String
private	estado	boolean
public	Categoria()	
public	Categoria(int, String, String, boolean)	
public	listarInformacion()	String

Ahora, crea una copia de la clase **Producto** llamada **Artículo** y guárdela dentro del paquete "**artículos**". Verifique que la clase Artículo tenga la misma estructura de la clase Producto y quede guarda en el mismo paquete de la clase Categoría.

Realice de forma secuencial las siguientes sentencias en la clase **Artículo**:

1. Agregue un constructor con los parámetros código, nombre, valor, cantidad
2. Cambie el nombre del atributo **categoría** por **tipo**
3. Actualice la variable **categoría** por **tipo** en los constructores y en el método **listarInformación()**.
4. Actualice la variable **categoría** por **tipo** en el get y set
5. Cambie el tipo de datos **String** del atributo tipo (antes llamado categoría) por **Categoría**
6. Actualice el tipo de dato **String** por **Categoría** para el parámetro **tipo** en los constructores
7. Actualice el tipo de dato **String** por **Categoría** para el **get**Tipo y **set**Tipo
8. Para concatenar el nombre de la categoría en el método **listarInformacion**(), debe hacer referencia de la siguiente forma **tipo.getNombre**(). Esto genera error si no existen los getter y setter en la clase Articulo; lo cual permite leer o asignar el valor del atributo nombre de la clase (objeto) Categoría; de igual forma podrían leer el valor de los demás atributos del objeto categoría.
9. La clase Articulo debería quedar con la siguiente estructura:

Centro de Servicios y Gestión Empresarial	ACTIVIDAD:	Aplicar Programacion Orientada a Objetos		
	PROGRAMA:		COMPETENCIA:	
	FICHA:		INSTRUCTOR:	
	FECHA:		APRENDIZ:	

Articulo		
private	codigo	int
private	nombre	String
private static	tipo	Categoria
private	valor	int
private	descuento	double
private	cantidad	byte
public static	estado	boolean
public	Articulo()	
public	Articulo(int, String, Categoria)	
public	Articulo(int, String, Categoria, int, byte)	
public	**Articulo(int, String, int, byte)**	
private	calcularVenta()	double
public	calcularDescuento()	void
public	calcularDescuento(byte)	void
public	calcularTotal()	double
public static	cambiarCategoria(String)	void
public	listarInformacion()	String

Ahora, crea una clase ejecutable (con método main) llamada **AdminArticulo** dentro de un paquete llamado "**inicio**" y dentro del método main, realice de forma secuencial las siguientes sentencias

1. Ingresar por teclado el código, nombre, valor, cantidad de un artículo y guarde estos valores en variables.
2. Crea un ***objeto*** (objArt) de la clase ***Articulo***, asignando estos valores (poblar) por medio del constructor más indicado.
   Ejemplo:

```
Articulo objArt;
objArt=new Articulo(7052,"Platano",800,(byte)10);
```

3. Ingresar por teclado el código, nombre, descripción de una categoria y guarde estos valores en variables.
4. Crea un ***objeto*** (objCat) de la clase ***Categoría*** utilizando el constructor por defecto.

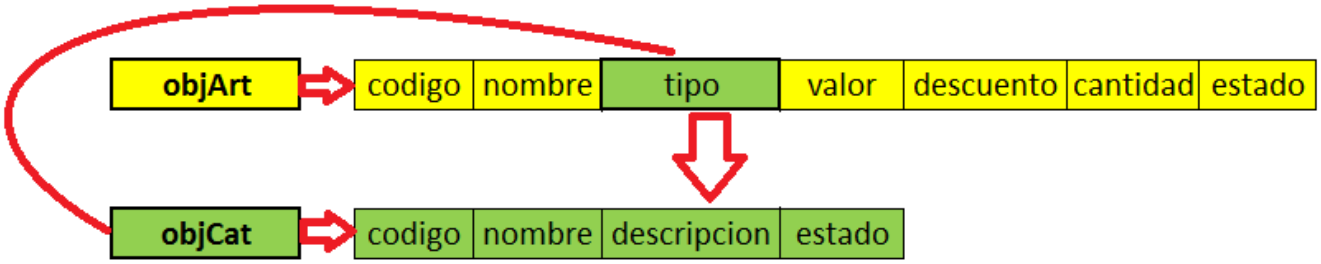
5. Asigna los valores a los atributos del objeto categoría (objCat) por medio de los métodos set. Ejemplo:

```
Categoria objCat=new Categoria();
objCat.setCodigo(2010);
objCat.setNombre("Verduras");
objCat.setDescripcion("Es perecedero");
objCat.setEstado(true);
```

6. Asigne el valor del atributo tipo al Articulo (objArt) por medio del métodos set.

```
objArt.setTipo(objCat);
```

La siguiente imagen ilustra analogicamente la representacion del los objetos creados en memoria RAM



La imagen muestra que un atributo (tipo) del Objeto *Articulo* (objArt) hace referencia otro Objeto de la clase *Categoría* (objCat), identificando la categoría a la cual pertenece un articulo.

7. Mostrar la información del Articulo
8. Cambie la categoría del Articulo (objArt), utilizando el método **cambiarCategoria**(). Muestre dos formas de hacer el cambio de la categoría del **objArt**.
9. Mostrar la información del Artículo. Observe que la descripción de la categoría no se visualiza.
10. Realice los cambios necesarios para que se pueda mostrar la descripción de la categoría a la cual pertenece el objeto **objArt**. Visualice nuevamente la informacion del objeto para confirmar el cambio.
11. Crea un ***objeto*** (objCat2) de la clase ***Categoría*** y pase toda la informacion (poblar) del objeto por medio del constructor.

Centro de Servicios y Gestión Empresarial	ACTIVIDAD:	Aplicar Programacion Orientada a Objetos		
	PROGRAMA:		COMPETENCIA:	
	FICHA:		INSTRUCTOR:	
	FECHA:		APRENDIZ:	

EJERCICIO # 4

Crear una clase llamada **Cuenta** dentro de un paquete llamado "**cuentas**", la cual tendrá los siguientes atributos: *código*, *titular* (documento del titular), *retiros* (es un vector de enteros), *consignaciones* (es una variable de tipo String, cuyo valor por defecto es vacía **" "**) y *saldo* (puede tener decimales).

La clase cuenta debe cumplir con lo siguiente:

1. El titular de una cuenta, podrá realizar 0, 1 ó la cantidad de ***retiros*** autorizada en la cuenta durante el mes, los cuales deben guardarse en el vector "*retiros*" y no tiene límite para el número de consignaciones en la cuenta.
2. Un **constructor** por defecto: el cual inicializa el vector *retiros* con un tamaño de 4, puesto que la cantidad de retiros autorizados mensualmente en una cuenta, por defecto es 4.
3. Un **constructor** con los parámetros: *código*, *titular*, *saldo*, *cantidadRetiros*; el cual inicializa el vector *retiros* con un tamaño cuyo valor está dado por cantidadRetiros.
4. Un **constructor** con todos los atributos como parámetro (*código*, *titular*, *retiros[]*, *saldo*); el parámetro retiros es un vector de enteros.
5. Métodos **get** y **set** para cada atributo, excepto el **set** para *retiros y consignaciones.*
6. Un método instanciable **consignar (int valor):** recibe un valor e incrementa el saldo de la cuenta, es decir, suma el valor al saldo y **retorna** el valor **true.** si el valor introducido como parámetro es menor o igual a cero, no se hará nada (no se resta) y se **retorna** el valor **false**. El valor consignado debe concatenarlo en el atributo *consignaciones*, de tal forma que las consignaciones queden separadas por un espacio y una coma.
7. Un método privado **retirosDisponibles():** devuelve el número de retiros disponibles de la cuenta, teniendo en cuenta que el número de retiros que una cuenta puede tener se estable en el constructor.
8. Un método instanciable **retirar (int valor):** recibe un valor y se lo resta del saldo de la cuenta y el valor retirado debe quedar almacenado en el vector *retiros* en la posición disponible y retorna un mensaje con el nuevo saldo ("***Retiro exitoso, saldo disponible:*** "+***xxx***)
   Para realizar esta operación se debe tener en cuenta las siguientes consideraciones:
   Si el valor del saldo es mayor o igual que el valor a retirar se reducirá el saldo de lo contrario debe retornar un *mensaje* con el valor del saldo ("***Su saldo es:*** " + ***xxx***).
   Si el número de retiros disponibles en la cuenta es mayor a cero, se reducirá el saldo de lo contrario debe retornar el *mensaje "**No tiene retiros disponibles**"*
   Si el valor a retirar es mayor a cero**,** se reducirá el saldo de lo contrario debe retornar el mensaje "***Error. Valor incorrecto***".

9. Un método publico **obtenerConsignaciones():** devuelve un vector de enteros con los valores de las consignaciones realizadas; en caso de que no se tengan consignaciones en debe devolver null;
10. Un método **listarInformacion()**: el cual debe mostrar toda la información de la cuenta (atributos del objeto), incluyendo el valor de todos los retiros; ejemplo
    Retiro1: 5000
    Retiro2: 30000
    Retiro3:10000, adicionalmente muestre la cantidad de retiros disponibles y las consignaciones realizadas.

*Crea una clase **AdminCuenta** (con método main) dentro del mismo paquete donde está la clase "**cuenta**". Esta clase debe cumplir con las siguientes funcionalidades:*

1. Crear una variable **global** y **static** de tipo Cuenta (**Cuenta** *objCuenta**;**).
2. Un método que muestre un menú y devuelva el número correspondiente a la opción seleccionada:

   Seleccione opción:
   1: Registrar cuenta
   2: Consignar en la cuenta
   3: Retirar de la cuenta
   4: Mostrar cuenta
   5: Salir

3. Además desde la clase ejecutable ofrece otro comportamiento:
   1. Si el usuario ingresa el número **1** (opción *Registrar cuenta*), debe ejecutar el método ***registraCuenta ():*** el cual permitirá:
      - Ingresar por teclado el código, *titular, saldo* y la cantidad de retiros autorizados de la cuenta.
      - Crear un objeto (**objCuenta**), teniendo en cuenta los siguiente:
        a. Si la cantidad de retiros es cero (0) debe crear el objeto (instancia) utilizando el constructor por defecto y asignar los valores de los atributos al objeto, mediante los métodos **set**
        b. Si la cantidad de retiros esta entre 1 y 4, debe crear el objeto utilizando el constructor que requiere como argumento el *código*, *titular*, *saldo*, *cantidadRetiros*
        c. Si la cantidad de retiros es mayor a 4, debe crear el objeto utilizando el constructor que requiere como argumento *código*, *titular*, ***retiros***, *saldo*. Tiene que crear un vector cuyo tamaño está dado por cantidad de retiros para pasarlo como argumento al constructor.

      - Mostrar el mensaje "***Cuenta registrada exitosamente***".

2. Si ingresa el numero **2** (opción *Consignar a la cuenta*), debe ejecutar el método ***consignar():*** el cual permitirá:
   - ❖ Ingresar por teclado el valor a consignar en la cuenta y este valor se le debe entregar al objeto (**objCuenta**) mediante la invocación del método **consignar(int valor)*.***
   - ❖ Mostrar el mensaje "***Consignación registrada exitosamente***"
3. Si ingresa el numero **3** (opción *Retirar de la cuenta*), debe ejecutar el método ***retirar():*** el cual permitirá:
   - ❖ Ingresar por teclado el valor a retirar de la cuenta y este valor se le debe entregar al objeto (**objCuenta**) mediante la invocación del método **retirar(int valor)*.***
   - ❖ Mostrar el mensaje "***Retiro registrado exitosamente***"
4. Si ingresa el numero **3** (opción *Mostrar cuenta*), debe ejecutar el método ***mostrar():*** el cual permitirá validar lo siguiente:
   Si existe una cuenta, invocara el método **listarInformacion()** del objeto (**objCuenta**), de lo contrario debe mostrar el mensaje "***No existe cuenta para mostrar***"

Nota: los métodos ***consignar(), retirar(), mostrar()*** deben validad que exista el objeto (**objCuenta**), es decir, que este creado (inicializado)