PROGRAMACIÒN ORIENTADA A OBJETOS

TALLER No. 3 JAVA	PROGRAMACIÓN ORIENTADA A OBJETOS

Profesor: Jaime Alberto Guzmán Luna

Contenido del taller:

1) Encapsulamiento
2) Constructores
3) this

Ejercicio 1

Para resolver el siguiente ejercicio, tenga en cuenta lo siguiente.

ArrayList es una clase que se encuentra en el paquete java.util y funciona como un arreglo de objetos dinámico, permitiendo una mayor versatilidad para el programador frente a los arreglos nativos de java. Dentro de un ArrayList, por defecto, se puede almacenar cualquier tipo de objetos.

El operador diamante <> permite restringir métodos o clases a un tipo de objetos específicos. Por ejemplo, al utilizarlo con la clase ArrayList (ArrayList<Clase>) se limita el arreglo a que reciba solo objetos de la clase que se desea.

Entre los métodos disponibles de la clase ArrayList, se encuentran:
• add(): Agrega un nuevo objeto al final del arreglo.
• remove(): Remueve un objeto del arreglo, todos los que están después de él se corren una posición. Puede recibir tanto el objeto a borrar como el índice del mismo.
• size(): Retorna la cantidad de elementos del arreglo.

La sentencia return en un método hace que se retorne el valor y se detenga la ejecución del método, es decir, el código que se encuentra debajo de la sentencia return, no es ejecutado. En un método void (si bien no retorna nada), se puede forzar la finalización de este utilizando únicamente la palabra return.

La clase String tiene varios métodos, entre ellos:

- equals(): Recibe una cadena como argumento y devuelve true si ambas son iguales
- contains(): Recibe una cadena como argumento y devuelve true si la cadena desde donde se llamó, contiene a la cadena recibida como argumento.

Actividad: Complete los espacios en ** del siguiente código de modo que se imprima el siguiente resultado en pantalla y los atributos y métodos estén lo más encapsulados

posible.

Nota: no puede modificar otras secciones, ni agregar o eliminar código.

```
2
2
3
Orden 101 creada
5
128 va a retirar producto
Pantalon retirado
4
4
```

Figura 1: Resultado esperado en pantalla.

```
1  package objtaller3;
2
3  import compras.**
4  import gestionHumana.Empleado;
5  import **
6
7  public class ObjTaller3 {
8
9      public static void main(** args) {
10         Producto p1 = new Producto(1, "Escoba", "Aseo");
11         Producto p2 = new Producto(2, "Camisa", "Ropa");
12         Producto p3 = new Producto(3, "Trapera", "Aseo");
13         Producto p4 = new Producto(4, "Pantalon", "Ropa");
14         Producto p5 = new Producto(5, "Jabon", "Aseo");
15         Empleado emp1 = new Empleado(405, "Juan", "Ingeniero");
16         ArrayList<Producto> productos1 = new ArrayList<>();
17         productos1.add(p1);
18         productos1.add(p3);
19         OrdenCompra orden1 = new OrdenCompra(**, "Aseo", emp1, productos1);
20         System.out.println(Producto.getTotalProductosPedidos());
21         orden1.agregarProducto(p4);
22         System.out.println(Producto.getTotalProductosPedidos());
23         orden1. **(p5);
24         System.out.println(Producto.getTotalProductosPedidos());
25         System.out.println("Orden " + orden1.codigo + " creada");
26
27         Empleado emp2 = new Empleado(128,"Susana", "Administradora de sucursal");
28         ArrayList<Producto> productos2 = new ArrayList<>();
29         productos2.add(p2);
30         productos2.add(p4);
31         OrdenCompra orden2 = new OrdenCompra(202, "Ropa", emp2, productos2);
32         System.out.println(Producto.getTotalProductosPedidos());
33         System.out.println(emp2.cedula + " va a retirar producto");
34         orden2.retirarProducto(emp2, p4);
35         System.out.println(Producto.getTotalProductosPedidos());
36         orden2.retirarProducto(**, p2);
37         System.out.println(Producto.getTotalProductosPedidos());
38     }
39 }
```

```
1  package **
2
3  public class Empleado {
4
5      **  final long cedula;
6      **  String nombre;
7      **  String cargo;
8
9      public Empleado(long cedula, String nombre, String cargo) {
10         this.cedula = cedula;
11         this.nombre = nombre;
12         this.cargo = cargo;
13     }
14
15     **  boolean tengoPermiso() {
16         return cargo.contains("Administrador");
17     }
18 }
```

```
1  package compras;
2
3  import **
4  import java.util.ArrayList;
5
6  public class ** {
7
8      **  int codigo;
9      **  String tipo;
10     **  Empleado comprador;
11     **  ArrayList<Producto> productos;
12
13     public OrdenCompra(int codigo, String tipo, Empleado comprador,
14             ArrayList<Producto> productos) {
15         this.codigo = codigo;
16         this.tipo = tipo;
17         this.comprador = comprador;
18         this.productos = productos;
19         Producto.totalProductosPedidos += productos.size();
20     }
21
22     **  void agregarProducto(Producto producto) {
23         if (producto.tipo.equals(tipo)) {
24             productos.add(producto);
25             Producto.totalProductosPedidos++;
26         }
27     }
28
29     **  void retirarProducto(Empleado empleado, Producto producto) {
30         if (!empleado.tengoPermiso()) {
31             **
32         }
```

```
33          retirarProducto(producto);
34      }
35
36      private void retirarProducto(Producto producto) {
37          for (int i = 0; i < **; i++) {
38              if (producto.getCodigo() == productos.get(i).getCodigo()) {
39                  productos.remove(i);
40                  producto.totalProductosPedidos--;
41                  producto.imprimirNombre();
42                  System.out.println(" retirado");
43                  break;
44              }
45          }
46      }
47
48      public ** descontar() {
49          Producto.totalProductosPedidos -= productos.size();
50      }
51
52  }
```

```
1   package compras;
2
3   public class Producto {
4
5       **  final int codigo;
6       **  String nombre;
7       **  String tipo;
8       **  static int totalProductosPedidos;
9
10      **  Producto(int codigo, String nombre, String tipo) {
11          this.codigo = codigo;
12          this.nombre = nombre;
13          this.tipo = tipo;
14      }
15
16          **  void imprimirNombre() {
17          System.out.print(nombre);
18      }
19
20
21
22
23
24      public ** getCodigo() {
25          return codigo;
26      }
27
28      **   int getTotalProductosPedidos() {
29          return totalProductosPedidos;
30      }
31  }
```

Pregunta: ¿Como piensa abordar el problema? ¿Ideo alguna estrategia?

Ejercicio 2 - GitHub

Enlace entrega: https://classroom.github.com/a/3KgW6GBR

PROBLEMA Fábrica de Televisores

Suponga que tiene unos ahorros y desea incursionar en el negocio de los televisores, para esto usted se olvida de todos los diseños de electricidad componentes, usted solo se encargara del funcionamiento del software del televisor.

Para la creación del software se le proporciona la siguiente guía y requerimientos a seguir:

Primero, todas las clases a desarrollar se deben implementar en el paquete taller3.televisores.

Segundo. Se le quiere dar una marca a los televisores. Para esto se definirán las marcas que tendrán estos televisores. Para hacer esto posible se crea una clase Marca que tendrá:

- Un atributo *nombre* (String)
- Un constructor que recibirá el nombre que le daremos al objeto Marca
- Por último, incluir los métodos *get* y *set* del atributo.

Tercero. Definir la clase TV la cual se encargará de modelar el principal funcionamiento y fin de este software. Esta clase tiene inicialmente las siguientes características:

- Los atributos: marca (Marca), canal(int), precio(int), estado(boolean), volumen(int) y control (Control).
- Tendrá un constructor que tiene como parámetros la marca y el estado (si el televisor este encendido o no)
- Por defecto el valor de los atributos canal, volumen y precio serán: 1, 1 y 500 respectivamente.
- Tendrá los métodos set y get para los atributos marca, control, precio, volumen y canal.
- Como somos una fabrica y queremos llevar las cuentas de la fabricación de televisores, implementar el conteo de televisores creados (objetos creados de la clase) para lo cual se debe implementar el atributo numTV (de tipo clase y no de instancia) y el método correspondiente para acceder a este valor.
- Definir los métodos para cambiar el estado de encendido a apagado llamados turnOn y turnOff, que respectivamente se encargaran de encender y apagar el televisor.
- Implementar el método getEstado que retornara el valor del atributo estado.

Otro requerimiento en nuestros televisores es que implementemos el cambio de canal y el cambio de volumen, para esto definimos los métodos canalUp y canalDown, que se encargaran de cambiar el canal aumentándolo o disminuyéndolo, y los métodos volumenUp, y volumenDown que se encargan de cambiar el volumen aumentándolo o disminuyéndolo.

Una de las dos limitaciones que se afronta en estos televisores, es que los canales disponibles solo van del canal 1 al canal 120 y para cambiar de canal necesariamente debe estar encendido el televisor, por obvias razones, así que en este diseño que se está construyendo, se debe implementar las condiciones que sean necesarias para representar lo anterior. También ocurre lo mismo con el volumen, esta ira de 0 a 7 y para que este cambie debe estar encendido el televisor. Para ambos casos, cambio de canal o subir volumen, en caso de un valor por encima o

debajo de sus valores límites o que no se cumpla que esté encendido, no se debe hacer nada.

Para finalizar la fabricación de los televisores, se debe incluir un control asociado al televisor. Para esto se define una clase llamada Control, con el fin de controlar a distancia el televisor. Este tendrá las siguientes características:

- Tendrá un único atributo que será **tv**, el cual permite el enlace entre el control y el televisor asociado.
- El control podrá hacer uso de los métodos del televisor de manera remota, para lo cual se definirán los métodos: turnOn, turnOff, canalUp, canalDown, volumenUp, volumenDown y setCanal.
- Definir un método llamado enlazar con el cual se conecta el control con el televisor que se quiere manejar remotamente. Este método recibirá como parámetro un televisor, y se encargará de asignar el valor del atributo **tv** y al igual que de solicitar al televisor que fue pasado cómo parámetro, que se asigne a su atributo control el mismo objeto control dueño del método enlazar.
- Tenga en cuenta agregar los métodos get y set para el atributo tv.

Si todo salió bien al final se tendrá el modelado de los mejores televisores que se habrán podido fabricar (o programar).

Nota 1: Como se quiere que la competencia no se roben los datos o accedan a ellos, se debe modelar los atributos lo menos accesible posible. Igualmente, dejar como público, las clases, constructores y métodos.

Nota 2: Use los nombres de las clases, atributos, métodos y parámetros en donde se indique, pero si no se indican, puede usar cualquier otro valor. Esto con el fin de usar una herramienta con la cual podrá observar si su implementación es correcta.

Nota 3: Se proporciona un archivo principal llamado Testtv, asegurese que lo allí descrito se ejecute sin errores.

Nota 4: Lea el readme de la actividad.

PARA PENSAR…

1. ¿Por qué cree que es necesario el atributo tv de la clase Control?
2. ¿Por qué cree que es necesario que la clase Control defina el atributo tv y la clase TV defina el atributo control, ambas clases tendrían la referencia de los objetos, esto que facilidad agrega a mi implementación?
3. ¿Por qué es necesario usar la cláusula this en el constructor de TV?
4. ¿A quién hace referencia this en un método de objeto o instancia? ¿Cómo pasaste el valor del objeto de tipo Control al televisor que se pasó como parámetro en el método enlazar?
5. ¿Por qué son útiles los constructores al momento de definir una clase y crear un objeto?
6. ¿Qué utilidad tienen los métodos set y get? ¿Considera que este método puede ser me de utilidad en la creación de restricciones o validaciones para una clase?

7. ¿Qué ocurre con el constructor vacío de la clase TV? ¿Aun es posible crear una instancia de TV sin parámetros?