ADRIAN MAXIMILIANO MUNOZ MARTINEZ

Laboratorio POO  Sabado 7 am – 9am

PRactica 3: Encapsulamiento

**Introducción**

Se desarrollará una aplicación que simulara el funcionamiento de una televisión, esta televisión será un objeto que heredara los atributos y métodos de la clase “Dispositivo electrónico” como los son:

-String Marca

-String Modelo

+void Encender()

+String Apagar ()

E implementara los métodos de la interfaz “Funciones electrónicas”:

+String CambiarCanal()

+String Volumen() .

**Objetivo**

El objetivo de la practica es conocer las formas en las que se maneja la herencia en java por medio de clases concretas e interfaces.

**Practica**

Test.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package practica3;

import java.util.Scanner;

/\*\*

\*

\* @author FCFM

\*/

public class Test {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

Television tv;

tv = new Television();

Scanner sc = new Scanner(System.in);

System.out.print("Ingresa la marca de la television: ");

tv.setMarca(sc.nextLine());

System.out.print("Ingresa el modelo de la television: ");

tv.setModelo(sc.nextLine());

System.out.println("Television");

System.out.println("Marca: " + tv.getMarca());

System.out.println("Modelo: " + tv.getModelo());

tv.encender();

System.out.println(tv.volumen());

System.out.println(tv.cambiarCanal());

System.out.println(tv.apagar());

}

}

Television.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package practica3;

/\*\*

\*

\* @author FCFM

\*/

public class Television extends DispositivoElectronico implements FuncionesElectronicas{

private String color;

public String configMenu(){

return "";

}

@Override

public String cambiarCanal() {

return "Cambiando canal";

}

@Override

public String volumen() {

return "Cambiando volumen";

}

}

DispositivosElectronics.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package practica3;

/\*\*

\*

\* @author FCFM

\*/

public class DispositivoElectronico {

private String modelo;

private String marca;

public void encender(){

System.out.println("Dispositvo Encendido");

}

public String apagar(){

return "Apagando Dispositivo";

}

/\*\*

\* @return the modelo

\*/

public String getModelo() {

return modelo;

}

/\*\*

\* @param modelo the modelo to set

\*/

public void setModelo(String modelo) {

this.modelo = modelo;

}

/\*\*

\* @return the marca

\*/

public String getMarca() {

return marca;

}

/\*\*

\* @param marca the marca to set

\*/

public void setMarca(String marca) {

this.marca = marca;

}

}

FuncionesElectronicas.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package practica3;

/\*\*

\*

\* @author FCFM

\*/

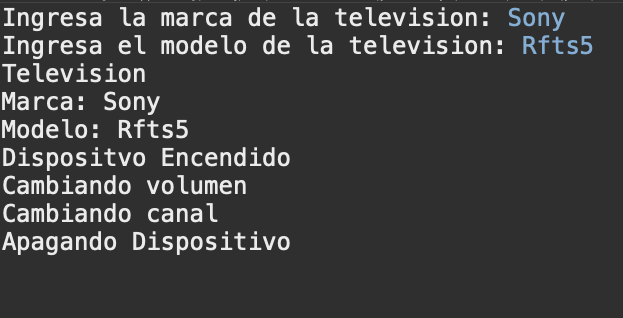
public interface FuncionesElectronicas {

public String cambiarCanal();

public String volumen();

}

**Screenshots**



**Explicación**

La clase que contiene al método main creará una instancia de la clase “Televisión” la cual heredará las características de la clase “Dispositivos Electrónicos” e implementará las funciones de la interfaz “Funciones Electrónicas”.

Después en la clase main se le asignaran valores a los atributos de la clase “Televisión” y se llamaran sus métodos heredados e implementados.

**Conclusión**

Un lenguaje de programación que tiene la capacidad de manejar objetos y que permite que se hereden las características entre ellos ofrece mucha versatilidad ya que un mismo objeto puede heredar funciones esenciales a muchos otros, lo que mantiene la estructura de una familia de clases de una manera mas organizada y de la misma forma aumenta la capacidad del programador de realizar cambios drásticos sin la necesidad de reestructurar por completo el código.