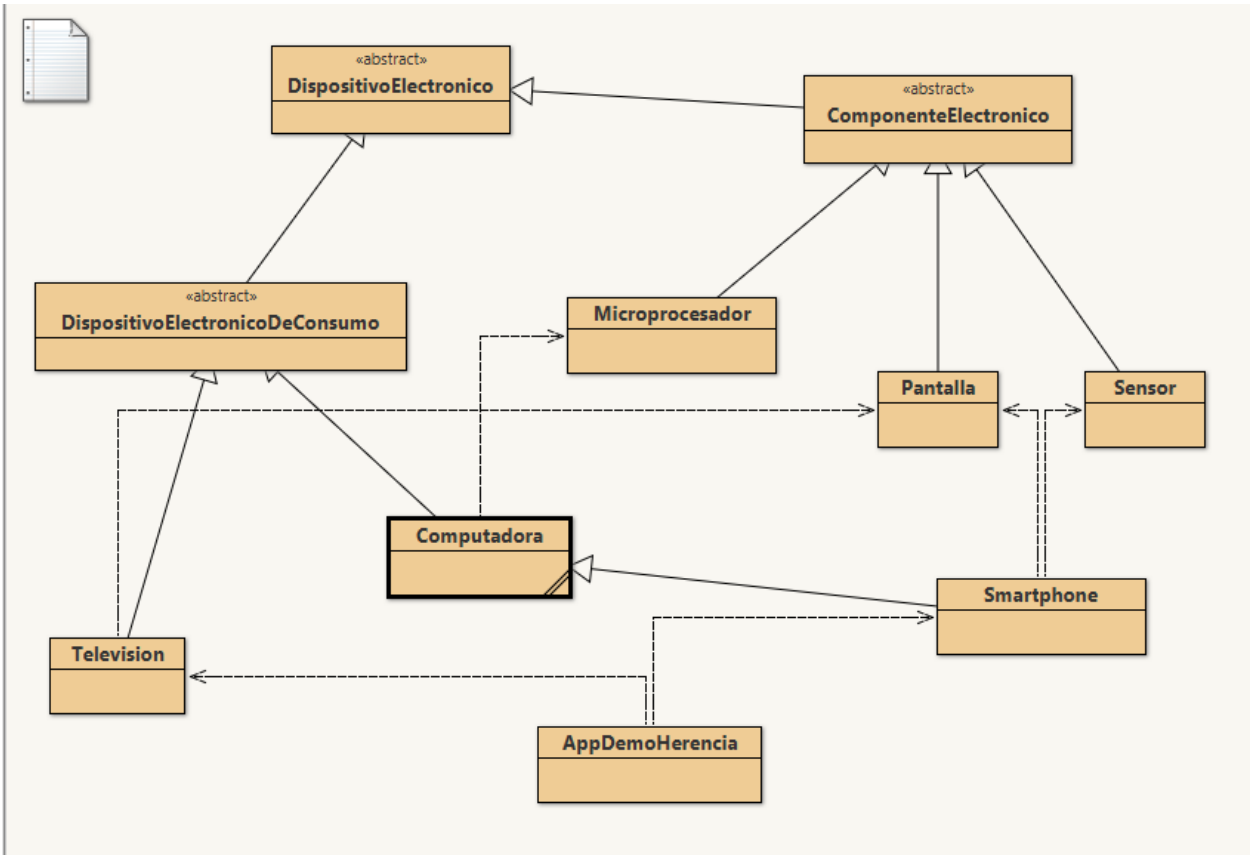
Universidad Autónoma de Baja California
Facultad de Ingeniería campus Mexicali
Ingeniero en Computación
Unidad de Aprendizaje Programación Orientada a Objetos Laboratorio

Práctica 7 - Herencia

Descripción de la actividad

1. Implementar las clases que se muestran en el siguiente diagrama:



DispositivoElectronico es la clase padre.
DispositivoElectronicoDeConsumo y ComponenteElectronico son un tipo de DispositivoElectronico.
Microprocesador, Pantalla y Sensor son un tipo de ComponenteElectronico.
Television y Computadora son un tipo de DispositivoElectronico.
Smartphone es un tipo de Computadora.

Los atributos por clase se muestran en la siguiente tabla:

Clase	Atributos
DispositivoElectronico	fabricante: String //inmutable numeroSerie: String //inmutable marca: String //inmutable nombre: String //inmutable costo: float //mutable componentes: ArrayList<ComponenteElectronico> //mutable usar agregarComponente() y quitarComponente()
ComponenteElectronico	esParteDe: DispositivoElectronico //mutable
DispositivoElectronicoDeConsumo	encendido: boolean // mutable por medio de los //métodos enceder() apagar() //El getter será isEncendido()
Microprocesador	cacheRAM: int //inmutable velocidadHz: long //inmutable
Pantalla	resolucionX: int //inmutable resolucionY: int //inmutable
Sensor	tipo: String //inmutable unidadDeMedida: String //inmutable valor: int //inmutable
Television	pantalla: Pantalla //inmutable
Computadora	cpu: Microprocesador //mutable ramMB: long //mutable
Smartphone	sensorDeHuella: Sensor //inmutable pantalla: Pantalla // inmutable
AppDemoHerencia	ninguno

Los atributos inmutables deben ser inicializados en el constructor y tener getter.
Los atributos mutables deben tener setter y getter y podrán o no ser inicializados en el constructor, es decir se requiere tener un constructor para cada caso.
Todas las clases, con excepción de *AppDemoHerencia*, deberán sobreescribir el método *toString( )*, de tal forma que la representación en cadena de cada objeto contenga el valor de todos sus atributos.

La clase AppDemoHerencia servirá para mostrar el funcionamiento de las demás clases. Deberá implementar sólo el método main(). Dentro de este método se deberá implementar la siguiente funcionalidad con variables locales:

1. Un ArrayList llamado *listaDeSmartphones,* conteniendo dos objetos del tipo Smartphone ya inicializados con valores en todos sus atributos.
2. Un ArrayList llamado *listaDeTVs,* conteniendo dos objetos del tipo Television ya inicializados con valores en todos sus atributos.
3. Una vez creados los objetos, desplegar con un ciclo:
    a. El valor de todos los atributos (incluyendo los heredados) de cada objeto en la *listaDeSmartphones.*
    b. El valor de todos los atributos (incluyendo los heredados) de cada objeto en la *listaDeTVs*.

Entregar por classroom el URL del repositorio de código en GitHub y reporte PDF con código fuente, capturas de pantalla, comentarios y conclusiones.

Recomendación para la construcción

Desarrollar las clases de lo general a lo particular e ir probando cada una conforme se van terminando. Utilizar AppDemoHerencia para probar cada clase.

El orden de construcción propuesto para facilitar el desarrollo es el siguiente:

A. AppDemoHerencia (iniciar sin código en el método main() e ir agregando código de cada clase que se vaya desarrollando)
B. DispositivoElectronico
C. ComponenteElectronico
D. Microprocesador
E. Pantalla
F. Sensor
G. DispositivoElectronicoDeConsumo
H. Television
I. Computadora
J. Smartphone

2. Programar las clases para tener la representación gráfica de todos los objetos tipo DispositivoElectronico, utilizar el prefijo IU para las nuevas clases.Ver en la Tabla 1 el código ejemplo anexo para desplegar un archivo .PNG en un Canvas. En la figura 1 se muestra la salida del código ejemplo.

Cada objeto gráfico deberá de conocer su ubicación y dimensiones, de tal manera que responda cuando detecte un click del mouse dentro de su área.
El constructor de las clases deberá recibir el nombre del archivo de imagen y el objeto al cual estará asociado.

Por ejemplo *IUComputadora* deberá desplegar una imagen de una computadora y tendrá un atributo del tipo *Computadora*. A continuación se muestra un ejemplo de cómo se inicializa el objeto de este tipo*:*

IUComputadora c = new IUComputadora(“mac.png”, new Computadora(....));

3. Realizar un programa de ejemplo que muestre una ventana con al menos un objeto gráfico de cada tipo de dispositivo y despliegue en consola el valor de sus atributos cuando el usuario haga click sobre él.

```
/**
 *
 * @author molguin
 */
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;

public class CanvasImagen extends Canvas {

   private BufferedImage image;

   public CanvasImagen() {
      try {
      // Cargar la imagen desde el archivo PNG
      image = ImageIO.read(new File("chip.png"));
      } catch (Exception e) {
      e.printStackTrace();
      }
   }

   public void paint(Graphics g) {
      // Dibujar la imagen en el objeto Canvas
      g.drawImage(image, 10, 10, 50,50, this);
   }

   public static void main(String[] args) {
      CanvasImagen canvas = new CanvasImagen();
      Frame frame = new Frame();
      frame.setSize(300, 300);
      frame.add(canvas);
      frame.setVisible(true);
   }
}
```

Tabla 1. Código ejemplo para desplegar una imagen dentro de un objeto tipo Canvas.

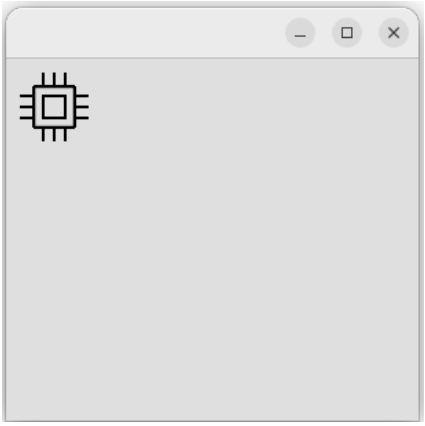


Figura 1. Salida producida por la ejecución del código ejemplo.