

Ejercicio: Sistema de Gestión de Dispositivos Electrónicos

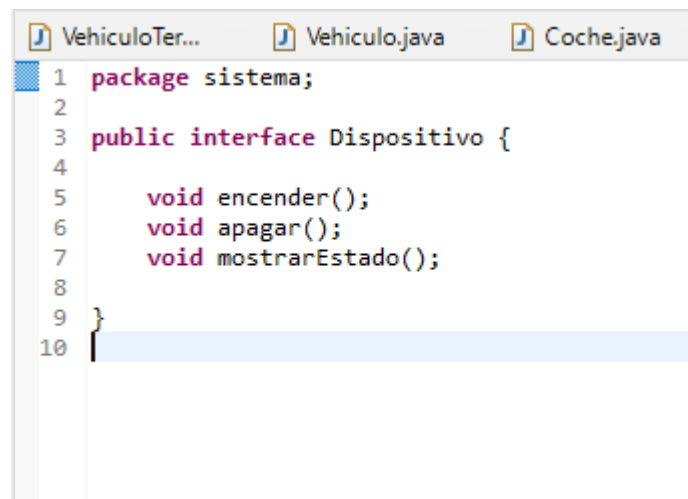
Descripción del ejercicio:

Vas a desarrollar un sistema que gestione distintos tipos de dispositivos electrónicos. Todos los dispositivos deben compartir ciertas características comunes, pero algunos tienen funcionalidades adicionales específicas según su tipo.

Requisitos:

1. Interface **Dispositivo**:

- Define los comportamientos básicos que deben tener todos los dispositivos electrónicos.
- Los métodos de esta interfaz deben incluir:
 - `encender()`: Método para encender el dispositivo.
 - `apagar()`: Método para apagar el dispositivo.
 - `mostrarEstado()`: Método para mostrar si el dispositivo está encendido o apagado.



```

1 package sistema;
2
3 public interface Dispositivo {
4
5     void encender();
6     void apagar();
7     void mostrarEstado();
8
9 }
10
  
```

2. Clase abstracta **DispositivoBase** que implemente la interfaz **Dispositivo**:

- Esta clase abstracta debe implementar la funcionalidad común para todos los dispositivos, como los atributos y la lógica de encender/apagar.
- Debe tener un atributo protegido boolean `estaEncendido` que represente el estado actual del dispositivo.
- Implementa los métodos `encender()`, `apagar()` y `mostrarEstado()`.

```
1 package sistema;
2
3 abstract class DispositivoBase implements Dispositivo {
4
5     protected boolean estaEncendido;
6
7     public DispositivoBase(boolean estaEncendido) {
8         this.estaEncendido = estaEncendido;
9     }
10
11     @Override
12     public void encender() {
13         estaEncendido = true;
14     }
15
16     @Override
17     public void apagar() {
18         estaEncendido = false;
19     }
20
21     @Override
22     public void mostrarEstado() {
23         if (estaEncendido) {
24             System.out.println("El dispositivo está encendido.");
25         } else {
26             System.out.println("El dispositivo está apagado.");
27         }
28     }
29
30 }
```

3. Subclases específicas que extiendan de DispositivoBase:

- Cada subclase debe representar un tipo específico de dispositivo, por ejemplo:
 - **Teléfono:** Implementa un método `realizarLlamada()` para simular la acción de hacer una llamada.

```
VehiculoTer...  Vehiculo.java  Coche.java  Moto.java  Bicicleta.java
1 package sistema;
2
3 public class Telefono extends DispositivoBase {
4
5     private int numero;
6
7     public Telefono(int numero, boolean estaEncendido) {
8         super(estaEncendido);
9         this.numero = numero;
10    }
11
12    public void realizarLlamada(int numero, boolean estaEncendido) {
13        if (estaEncendido) {
14            System.out.println("Realizando llamada al numero: " + numero);
15        } else {
16            System.out.println("No se puede realizar la llamada.");
17        }
18    }
19 }
```

- **Ordenador:** Implementa un método ejecutarPrograma(String programa) para simular la ejecución de un programa.

```
VehiculoTer...  Vehiculo.java  Coche.java  Moto.java  Bicicleta.java  Main.java
1 package sistema;
2
3 public class Ordenador extends DispositivoBase {
4
5     private String programa;
6
7     public Ordenador(boolean estaEncendido, String programa) {
8         super(estaEncendido);
9         this.programa = programa;
10    }
11
12    public void ejecutarPrograma(String programa, boolean estaEncendido) {
13        if (estaEncendido) {
14            System.out.println("Ejecutando el programa: " + programa);
15        } else {
16            System.out.println("El programa no se puede ejecutar");
17        }
18    }
19 }
```

- **Televisor:** Implementa un método cambiarCanal(int canal) para cambiar de canal.

```
VehiculoTer...  Vehiculo.java  Coche.java  Moto.java  Bicicleta.java  Main.java
1 package sistema;
2
3 public class Television extends DispositivoBase {
4
5     private int canal;
6
7     public Television(int canal, boolean estaEncendido) {
8         super(estaEncendido);
9         this.canal = canal;
10    }
11
12    public void cambiarCanal(int canal, boolean estaEncendido) {
13        if (estaEncendido) {
14            System.out.println("El canal ha sido cambiado al número " + canal);
15        } else {
16            System.out.println("No se puede cambiar de canal.");
17        }
18    }
19 }
```

- Cada dispositivo debe tener algún comportamiento único que lo diferencie de los otros tipos.

4. Clase principal Main:

- En esta clase, crea instancias de diferentes tipos de dispositivos y llama a sus métodos para probar su funcionamiento.
- Por ejemplo, enciende los dispositivos, muestra su estado, y realiza las acciones específicas de cada dispositivo.

```
VehiculoTer... Coche.java Moto.java Bicicleta.java Main.java Main.java X Dispositivo.... Di
1 package sistema;
2
3 public class Main {
4     public static void main(String[] args) {
5         // Crear y usar un Telefono
6         int numeroTelefono = 123456789;
7         Telefono telefono = new Telefono(numeroTelefono, false);
8
9         telefono.encender();
10        telefono.mostrarEstado();
11        int numeroLlamada = 987654321;
12        telefono.realizarLlamada(numeroLlamada, true);
13        telefono.apagar();
14        telefono.mostrarEstado();
15
16        System.out.println("-----");
17
18        // Crear y usar un Ordenador
19        String programa = "Microsoft Word";
20        Ordenador ordenador = new Ordenador(false, programa);
21
22        ordenador.encender();
23        ordenador.mostrarEstado();
24        ordenador.ejecutarPrograma(programa, true);
25        ordenador.apagar();
26        ordenador.mostrarEstado();
27
28        System.out.println("-----");
29
30        // Crear y usar una Television
31        int canalInicial = 1;
32        Television televisor = new Television(canalInicial, true);
33
34        televisor.encender();
35        televisor.mostrarEstado();
36        int nuevoCanal = 5;
37        televisor.cambiarCanal(nuevoCanal, true);
38        televisor.apagar();
39        televisor.mostrarEstado();
40    }
41 }
```

Console X

<terminated> Main [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (26 sept 2024, 14:23:22 – 14:23:24) [pid: 4448]

El dispositivo está encendido.
Realizando llamada al numero: 987654321
El dispositivo está apagado.

El dispositivo está encendido.
Ejecutando el programa: Microsoft Word
El dispositivo está apagado.

El dispositivo está encendido.
El canal ha sido cambiado al número 5
El dispositivo está apagado.

Pistas adicionales:

MARP 2024



Repaso JAVA, Herencia e Interfaces

- La **interface Dispositivo** es necesaria para garantizar que todos los dispositivos tengan los mismos métodos básicos (encender(), apagar(), mostrarEstado()).
- La **clase abstracta DispositivoBase** puede ser usada para evitar la repetición de código en las subclases, implementando los métodos comunes que no cambiarán entre dispositivos.

- En las subclases como Teléfono, Ordenador, y Televisor, deben sobrescribir o agregar métodos que reflejen acciones específicas de ese dispositivo.
- Debes asegurarte de que las subclases no repitan código innecesario; la clase abstracta debe encargarse de la mayor parte de la lógica común.