

4 Elementos claves para entender POO en Java

Abstracción

El término abstracción consiste en **ver a algo como un todo sin saber cómo está formado internamente.**

Ej. Smartphone, a nosotros nos da igual los componentes que lleve internamente, no necesitamos entenderlo

Encapsulamiento

Cuando se habla de encapsulamiento, hablamos de **ocultar la información**, este concepto es un poco más claro que la abstracción. atributos y métodos en privado

Herencia

El concepto fundamental de la herencia es el proceso en el que **un objeto adquiere características de otro objeto.** Reutilización para ahorrar tiempo

Polimorfismo

El polimorfismo es una palabra de origen griego que significa “**muchas formas**”, es un término un poco complejo de entender, que a veces sólo se lo entiende con la práctica.

El concepto dice que es la capacidad de un objeto para **comportarse de diferentes formas de acuerdo al mensaje enviado.**

Según el objeto recibido, el método se puede comportar de una forma u otra

Qué es una Clase?

Las clases son la base de la programación orientada a objetos, una clase es una **plantilla, molde** o modelo **para crear objetos.**

Sintaxis de una clase en Java.

```
class MiClase{  
  
    //Constructor de la Clase  
  
    //atributos de la clase  
  
    //métodos de la clase  
}
```

4 Elementos claves para entender POO en Java

Vamos a ver un ejemplo:

Creando la Clase Celular

```
package www.ecodeup.ejemploclase;
public class Celular {
    //atributos de la clase
    private String marca;
    private String modelo;
    private String color;

    // constructor con parámetros
    public Celular(String marca, String modelo, String color) {
        this.marca = marca;
        this.modelo = modelo;
        this.color = color;
    }

    //constructor vacio
    public Celular(){

    }

    // método hacer llamada
    public void llamar(String nombre){
        System.out.println("LLamando a "+nombre);
    }

    //método finalizar llamada
    public void cortarLlamada(){
        System.out.println("Llamada finalizada");
    }

    //método para informar de la características del celular
    public void informarCaracteristicas(){
        System.out.println(String.format("Celular Marca: %s",
marca));
        System.out.println(String.format("Celular Modelo: %s",
modelo));
        System.out.println(String.format("Celular Color: %s",
color));
    }

    //getters y Setters
    public String getMarca() {
        return marca;
    }
}
```

4 Elementos claves para entender POO en Java

```
    }  
    public void setMarca(String marca) {  
        this.marca = marca;  
    }  
    public String getModelo() {  
        return modelo;  
    }  
    public void setModelo(String modelo) {  
        this.modelo = modelo;  
    }  
    public String getColor() {  
        return color;  
    }  
    public void setColor(String color) {  
        this.color = color;  
    }  
}
```

Con esta clase tenemos **dos** de los **cuatro** conceptos vistos anteriormente: **Abstracción, Encapsulamiento.**

Qué es un objeto?

Un objeto es una instancia de una clase, una instancia es un ejemplar, un caso específico de una clase, por ejemplo yo puedo crear varios objetos de tipo clase y cada uno es diferente, por ejemplo el primero objeto es de color negro, el segundo objeto es de color blanco.

4 Elementos claves para entender POO en Java

Cómo crear utilizar un objeto de tipo Celular?

```
package www.ecodeup.test;

import www.ecodeup.ejemploclase.Celular;

public class Test {

    public static void main(String[] args) {

        //creo un nuevo objeto de tipo celular, con el
        constructor vacío

        Celular celular = new Celular();
        // le asigno la marca, modelo, y color
        celular.setMarca("Nokia");
        celular.setModelo("1100");
        celular.setColor("Gris");
        //utilizo lo métodos, llamar, cortarLlamada e informar
        características

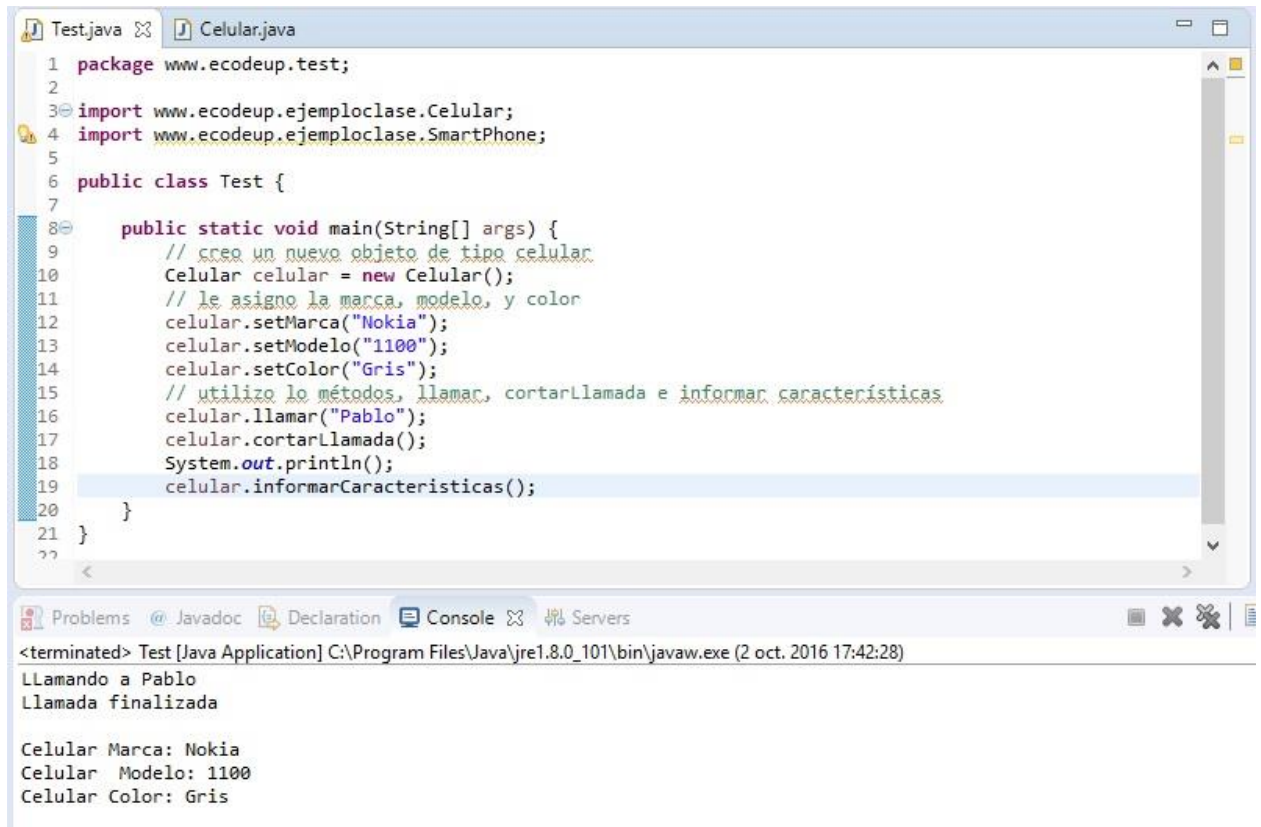
        celular.llamar("Pablo");
        celular.cortarLlamada();
        System.out.println();
        celular.informarCaracteristicas();

    }

}
```

4 Elementos claves para entender POO en Java

Salida por Consola



The screenshot shows an IDE with two tabs: 'Test.java' and 'Celular.java'. The 'Test.java' tab is active, displaying the following code:

```
1 package www.ecodeup.test;
2
3 import www.ecodeup.ejemploclase.Celular;
4 import www.ecodeup.ejemploclase.SmartPhone;
5
6 public class Test {
7
8     public static void main(String[] args) {
9         // creo un nuevo objeto de tipo celular
10        Celular celular = new Celular();
11        // le asigno la marca, modelo, y color
12        celular.setMarca("Nokia");
13        celular.setModelo("1100");
14        celular.setColor("Gris");
15        // utilizo los métodos, llamar, cortar llamada e informar características
16        celular.llamar("Pablo");
17        celular.cortarLlamada();
18        System.out.println();
19        celular.informarCaracteristicas();
20    }
21 }
```

The console output at the bottom shows the execution results:

```
<terminated> Test [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (2 oct. 2016 17:42:28)
Llamando a Pablo
Llamada finalizada

Celular Marca: Nokia
Celular Modelo: 1100
Celular Color: Gris
```

Cómo hacer herencia?

La sintaxis para una clase con herencia viene dado de la siguiente manera:

```
class MiClase extends ClaseHeredar{

    //Constructor de la Clase

    //atributos propios de la clase

    //métodos propio o sobrescritos

}
```

4 Elementos claves para entender POO en Java

Clase Celular

```
package www.ecodeup.ejemploclase;

public class Celular {
    //atributos de la clase
    private String marca;
    private String modelo;
    private String color;

    // constructor con parámetros
    public Celular(String marca, String modelo, String color) {
        this.marca = marca;
        this.modelo = modelo;
        this.color = color;
    }

    //constructor vacio
    public Celular(){

    }

    // método hacer llamada
    public void llamar(String nombre){
        System.out.println("LLamando a "+nombre);
    }

    //método finalizar llamada
```

4 Elementos claves para entender POO en Java

```
public void cortarLlamada(){
    System.out.println("Llamada finalizada");
}

//método para informar de la características del celular
public void informarCaracteristicas(){
    System.out.println(String.format("Celular Marca: %s",
marca));
    System.out.println(String.format("Celular Modelo: %s",
modelo));
    System.out.println(String.format("Celular Color: %s",
color));
}

//getters y Setters
public String getMarca() {
    return marca;
}
public void setMarca(String marca) {
    this.marca = marca;
}
public String getModelo() {
    return modelo;
}
public void setModelo(String modelo) {
    this.modelo = modelo;
}
public String getColor() {
    return color;
```

4 Elementos claves para entender POO en Java

```
    }  
    public void setColor(String color) {  
        this.color = color;  
    }  
}
```

Clase Smartphone heredada de la clase Celular

```
package www.ecodeup.ejemploclase;  
  
public class SmartPhone extends Celular{  
    private float pixelesCamara;  
    private float tamanoMemoriaInterna;  
    private float tamanoMemoriaExterna;  
  
    //constructor por defecto  
    public SmartPhone() {  
    }  
  
    //constructor con los atributos de la clase incluso los heredados  
    public SmartPhone(String marca, String modelo, String color,  
float pixelesCamara, float tamanoMemoriaRam,  
float tamaoDisco) {  
        super(marca, modelo, color);  
        this.pixelesCamara = pixelesCamara;  
        this.tamanoMemoriaInterna = tamanoMemoriaRam;  
        this.tamanoMemoriaExterna = tamaoDisco;  
    }  
}
```


4 Elementos claves para entender POO en Java

```
// método sobrescrito (override), utilizo código de la clase
Celular y añado código que necesito

@Override
public void informarCaracteristicas() {
    // TODO Auto-generated method stub
    super.informarCaracteristicas();
    System.out.println(String.format("SmartPhone calidad
cámara: %s pixeles", pixelesCamara));
    System.out.println(String.format("SmartPhone tamaño
memoria interna: %s GB", tamanoMemorialInterna));
    System.out.println(String.format("SmartPhone tamaño
memoria externa: %s GB", tamanoMemoriaExterna));
}

//getters y setters
public float getPixelesCamara() {
    return pixelesCamara;
}

public void setPixelesCamara(float pixelesCamara) {
    this.pixelesCamara = pixelesCamara;
}

public float getTamanoMemorialInterna() {
    return tamanoMemorialInterna;
}
```

4 Elementos claves para entender POO en Java

```
        public void setTamanioMemoriaInterna(float  
tamanioMemoriaInterna) {  
            this.tamanioMemoriaInterna =  
tamanioMemoriaInterna;  
        }  
  
        public float getTamanioMemoriaExterna() {  
            return tamanioMemoriaExterna;  
        }  
  
        public void setTamanioMemoriaExterna(float  
tamanioMemoriaExterna) {  
            this.tamanioMemoriaExterna =  
tamanioMemoriaExterna;  
        }  
    }
```

4 Elementos claves para entender POO en Java

Crear un objeto de tipo Smartphone

```
package www.ecodeup.test;

import www.ecodeup.ejemploclase.SmartPhone;

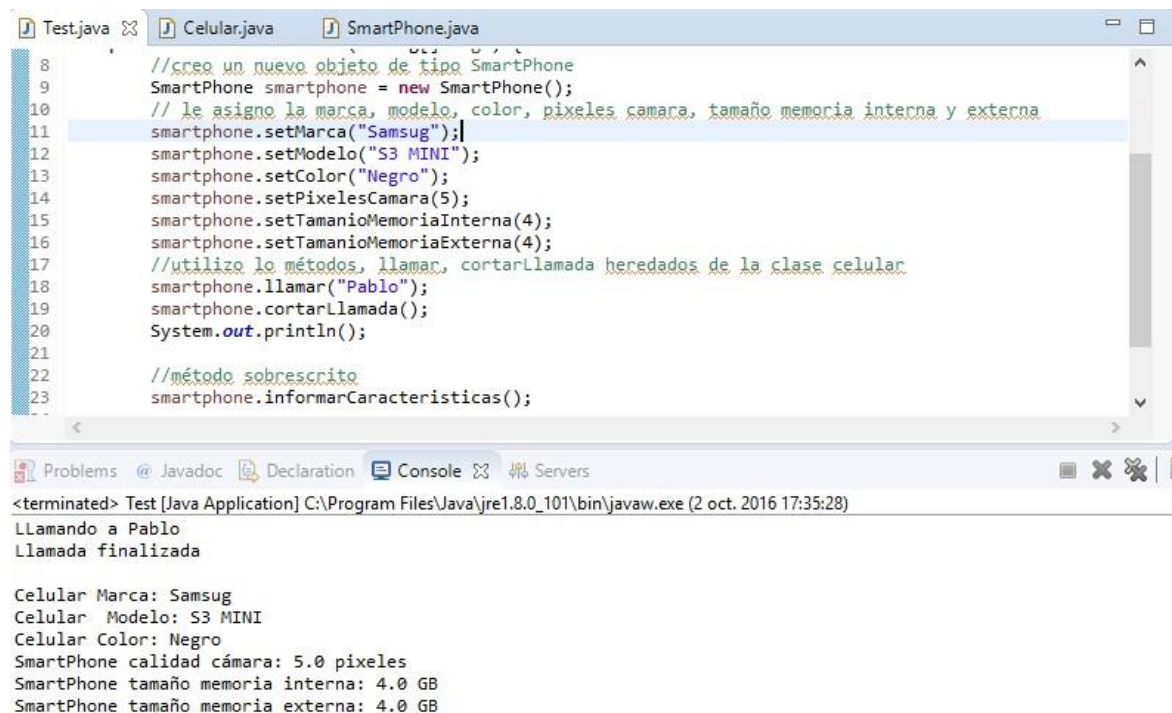
public class Test {

    public static void main(String[] args) {
        //creo un nuevo objeto de tipo SmartPhone
        SmartPhone smartphone = new SmartPhone();
        // le asigno la marca, modelo, color, pixeles camara,
        tamaño memoria interna y externa
        smartphone.setMarca("Samsug");
        smartphone.setModelo("S3 MINI");
        smartphone.setColor("Negro");
        smartphone.setPixelesCamara(5);
        smartphone.setTamanioMemoriaInterna(4);
        smartphone.setTamanioMemoriaExterna(4);
        //utilizo lo métodos, llamar, cortarLlamada heredados
        de la clase celular
        smartphone.llamar("Pablo");
        smartphone.cortarLlamada();
        System.out.println();

        //método sobrescrito
        smartphone.informarCaracteristicas();
    }
}
```

4 Elementos claves para entender POO en Java

Salida por Consola



The screenshot shows an IDE with three tabs: Test.java, Celular.java, and SmartPhone.java. The SmartPhone.java tab is active, displaying the following code:

```
8 //creo un nuevo objeto de tipo SmartPhone
9 SmartPhone smartphone = new SmartPhone();
10 // le asigno la marca, modelo, color, pixeles camara, tamaño memoria interna y externa
11 smartphone.setMarca("Samsug");
12 smartphone.setModelo("S3 MINI");
13 smartphone.setColor("Negro");
14 smartphone.setPixelesCamara(5);
15 smartphone.setTamañoMemoriaInterna(4);
16 smartphone.setTamañoMemoriaExterna(4);
17 //utilizo los métodos, llamar, cortarLlamada heredados de la clase celular
18 smartphone.llamar("Pablo");
19 smartphone.cortarLlamada();
20 System.out.println();
21
22 //método sobrescrito
23 smartphone.informarCaracteristicas();
```

Below the code editor, the Console tab is active, showing the output of the program:

```
<terminated> Test [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (2 oct. 2016 17:35:28)
Llamando a Pablo
Llamada finalizada

Celular Marca: Samsug
Celular Modelo: S3 MINI
Celular Color: Negro
SmartPhone calidad cámara: 5.0 pixeles
SmartPhone tamaño memoria interna: 4.0 GB
SmartPhone tamaño memoria externa: 4.0 GB
```

Cuando se utiliza el Polimorfismo?

Diseño e implementación sin Polimorfismo

Clase Vehículo

```
package com.ecodeup.polimorfismo;

//clase padre Vehiculo (Clase base)
public class Vehiculo {
    public void mostrarTipoVehiculo(){
        System.out.println("Soy un Vehiculo");
        System.out.println("Cobrar peaje Vheículo");
    }
}
```

4 Elementos claves para entender POO en Java

Clase Auto

```
package com.ecodeup.polimorfismo;

//clase hija Auto (SubClase)
public class Auto extends Vehiculo{
    @Override
    public void mostrarTipoVehiculo() {
        System.out.println("Soy un auto");
        System.out.println("Cobrar peaje auto");
    }
}
```

Clase Moto

```
package com.ecodeup.polimorfismo;

//clase hija Moto (SubClase)
public class Moto extends Vehiculo {
    @Override
    public void mostrarTipoVehiculo() {
        System.out.println("Soy una Moto");
        System.out.println("Cobrar peaje moto");
    }
}
```

4 Elementos claves para entender POO en Java

Clase Camion

```
package com.ecodeup.polimorfismo;

//clase hija Camion (SubClase)
public class Camion extends Vehiculo{
    @Override
    public void mostrarTipoVehiculo() {
        System.out.println("Soy un camión");
        System.out.println("Cobrar peaje camión");
    }
}
```

Clase Bus

```
package com.ecodeup.polimorfismo;

public class Bus extends Vehiculo{
    @Override
    public void mostrarTipoVehiculo() {
        System.out.println("Soy un bus");
        System.out.println("Cobrar peaje bus");
    }
}
```

4 Elementos claves para entender POO en Java

Clase Peaje

```
package com.ecodeup.polimorfismo;

public class Peaje {

    //método para cobrar peaje a la moto
    public void cobrarPeajeMoto(Moto moto){
        moto.mostrarTipoVehiculo();
    }

    //método para cobrar peaje al bus
    public void cobrarPeajeBus(Bus bus){
        bus.mostrarTipoVehiculo();
    }

    //método para cobrar peaje al auto
    public void cobrarPeajeAuto(Auto auto){
        auto.mostrarTipoVehiculo();
    }

    //método para cobrar peaje al camión
    public void cobrarPeajeCamion(Camion camion){
        camion.mostrarTipoVehiculo();
    }

}
```

4 Elementos claves para entender POO en Java

Test de la Clase Peaje

```
package com.ecodeup.test;

import com.ecodeup.polimorfismo.Auto;
import com.ecodeup.polimorfismo.Bus;
import com.ecodeup.polimorfismo.Camion;
import com.ecodeup.polimorfismo.Moto;
import com.ecodeup.polimorfismo.Peaje;

public class Test {

    public static void main(String[] args) {

        //declaro los objetos que heredan de tipo vehículo
        Auto auto = new Auto();
        Moto moto= new Moto();
        Bus bus = new Bus();
        Camion camion = new Camion();

        //declaro el objeto peaje para identificar y cobrar el
        peaje de acuerdo al tipo de vehículo
        Peaje peaje= new Peaje();
        //cobrar peaje auto
        peaje.cobrarPeajeAuto(auto);
        System.out.println();
        //cobrar peaje bus
        peaje.cobrarPeajeBus(bus);
        System.out.println();
        //cobrar peaje camion
```


4 Elementos claves para entender POO en Java

```
        peaje.cobrarPeajeCamion(camion);

        System.out.println();

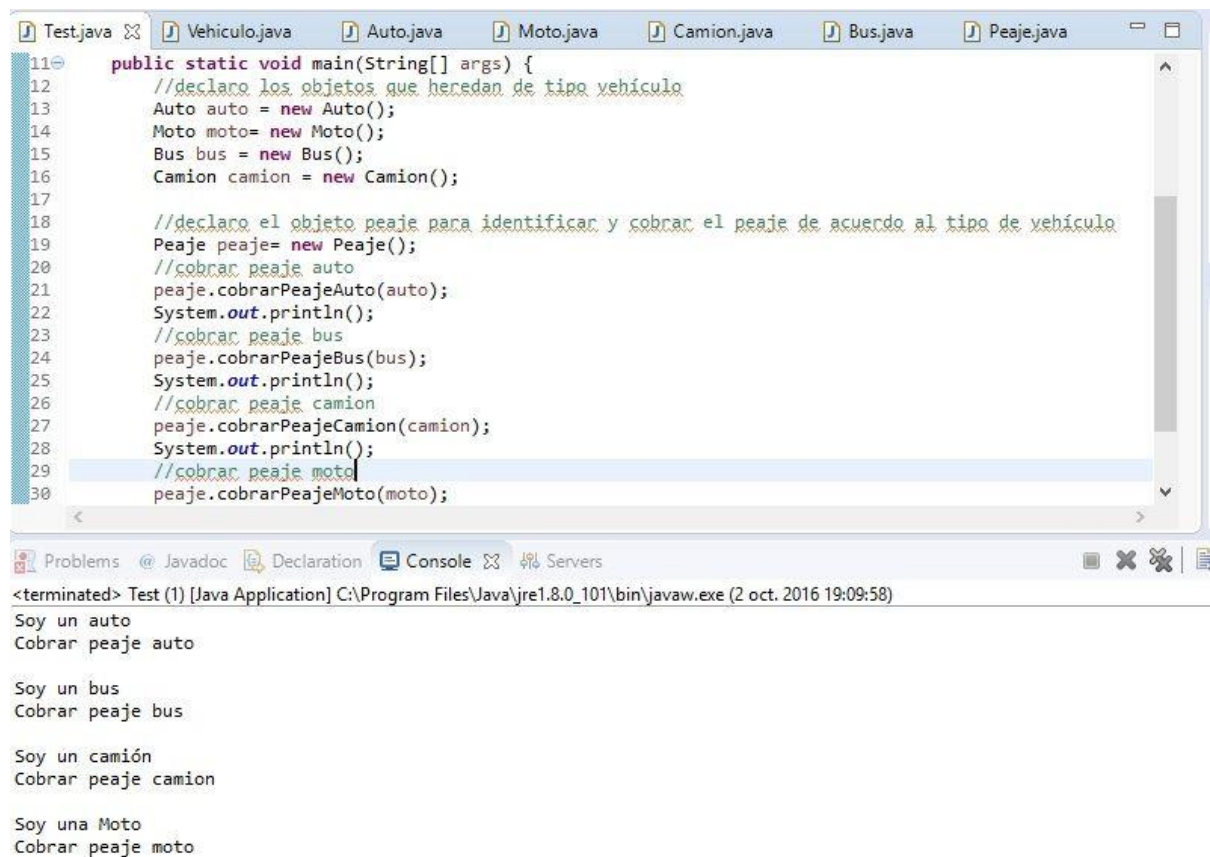
        //cobrar peaje moto

        peaje.cobrarPeajeMoto(moto);

    }

}
```

Salida por Consola



The screenshot shows an IDE with several Java files open: Test.java, Vehiculo.java, Auto.java, Moto.java, Camion.java, Bus.java, and Peaje.java. The Test.java file is active, showing a main method that creates instances of Auto, Moto, Bus, and Camion, and a Peaje object. It then calls the cobrarPeaje method on the Peaje object for each vehicle type. The console output shows the results of these calls, with each vehicle type printing its name and the toll amount.

```
public static void main(String[] args) {
    //declaro los objetos que heredan de tipo vehiculo
    Auto auto = new Auto();
    Moto moto= new Moto();
    Bus bus = new Bus();
    Camion camion = new Camion();

    //declaro el objeto peaje para identificar y cobrar el peaje de acuerdo al tipo de vehiculo
    Peaje peaje= new Peaje();
    //cobrar peaje auto
    peaje.cobrarPeajeAuto(auto);
    System.out.println();
    //cobrar peaje bus
    peaje.cobrarPeajeBus(bus);
    System.out.println();
    //cobrar peaje camion
    peaje.cobrarPeajeCamion(camion);
    System.out.println();
    //cobrar peaje moto
    peaje.cobrarPeajeMoto(moto);
}
```

<terminated> Test (1) [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (2 oct. 2016 19:09:58)

Soy un auto
Cobrar peaje auto

Soy un bus
Cobrar peaje bus

Soy un camión
Cobrar peaje camion

Soy una Moto
Cobrar peaje moto

4 Elementos claves para entender POO en Java

Diseño e implementación con Polimorfismo

```
package com.ecodeup.polimorfismo;

public class Peaje {

    //método para cobrar peaje para cualquier vehículo
    public void cobrarPeaje(Vehiculo vehiculo){
        vehiculo.mostrarTipoVehiculo();
    }
}
```

Test de la clase Peaje

```
package com.ecodeup.test;

import com.ecodeup.polimorfismo.Auto;
import com.ecodeup.polimorfismo.Bus;
import com.ecodeup.polimorfismo.Camion;
import com.ecodeup.polimorfismo.Moto;
import com.ecodeup.polimorfismo.Peaje;

public class Test {

    public static void main(String[] args) {
        // declaro los objetos que heredan de tipo vehículo
        Auto auto = new Auto();
        Moto moto = new Moto();
    }
}
```

4 Elementos claves para entender POO en Java

```
Bus bus = new Bus();

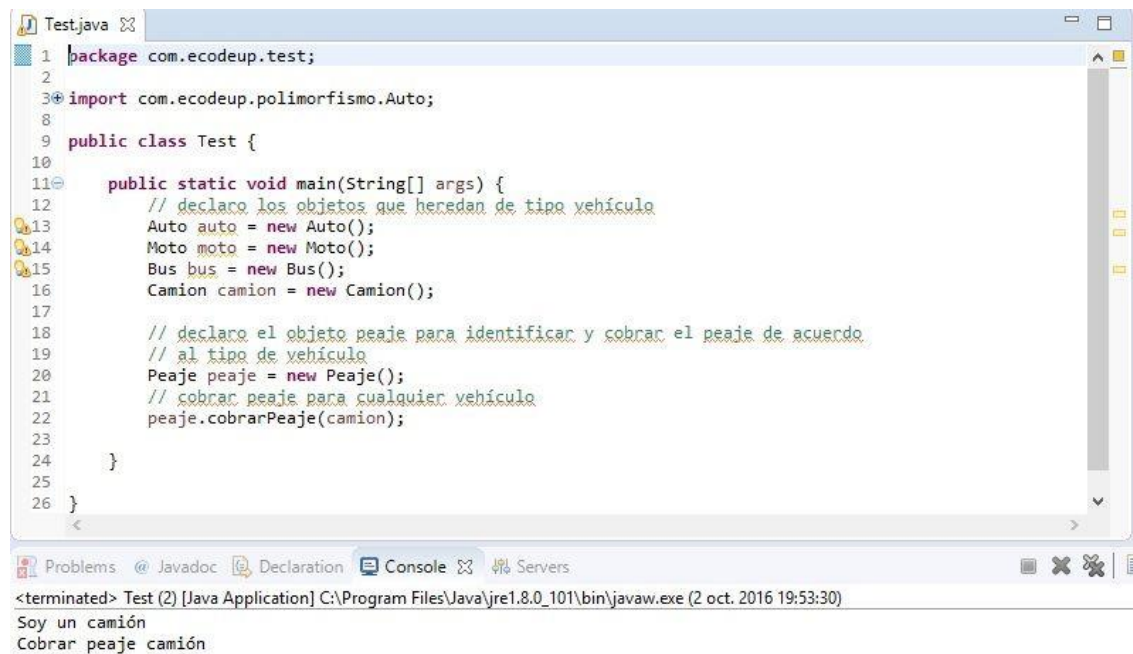
Camion camion = new Camion();

// declaro el objeto peaje para identificar y cobrar el
// al tipo de vehículo
Peaje peaje = new Peaje();
// cobrar peaje para cualquier vehículo
peaje.cobrarPeaje(camion);

}

}
```

Salida por Consola



The screenshot shows an IDE window titled 'Test.java' with the following code:

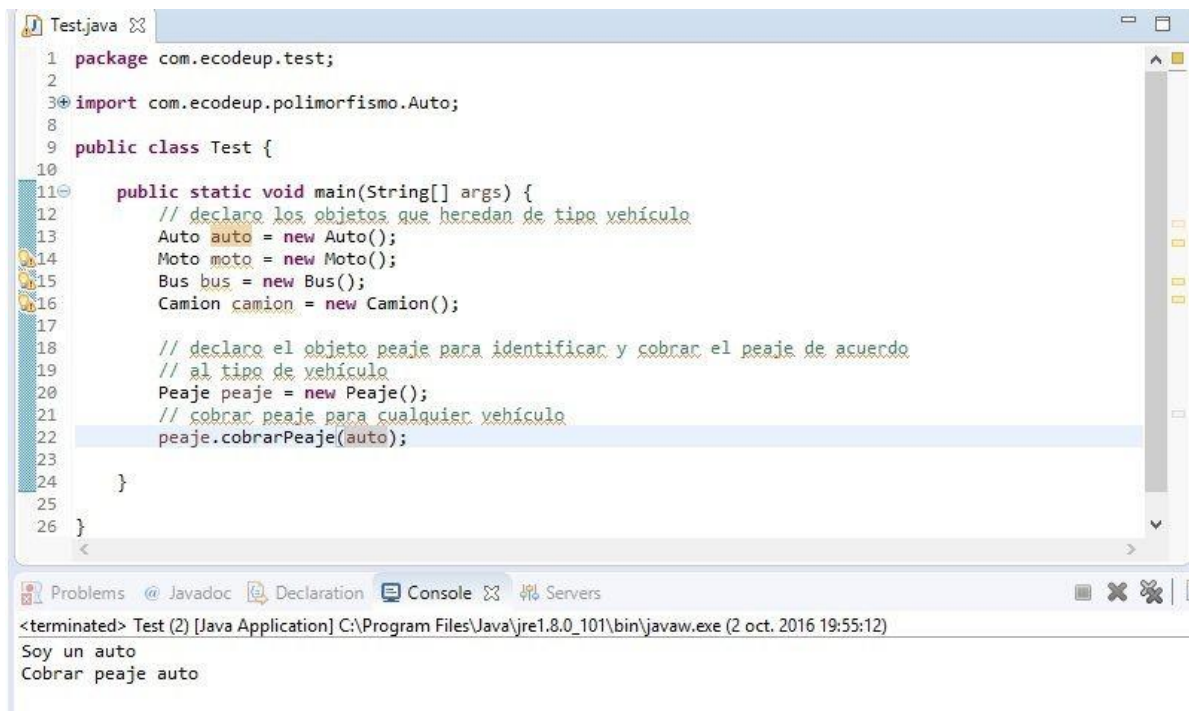
```
1 package com.ecodeup.test;
2
3 import com.ecodeup.polimorfismo.Auto;
4
5 public class Test {
6
7     public static void main(String[] args) {
8         // declaro los objetos que heredan de tipo vehículo
9         Auto auto = new Auto();
10        Moto moto = new Moto();
11        Bus bus = new Bus();
12        Camion camion = new Camion();
13
14        // declaro el objeto peaje para identificar y cobrar el peaje de acuerdo
15        // al tipo de vehículo
16        Peaje peaje = new Peaje();
17        // cobrar peaje para cualquier vehículo
18        peaje.cobrarPeaje(camion);
19    }
20 }
21 }
```

The console output at the bottom shows the following messages:

```
<terminated> Test (2) [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (2 oct. 2016 19:53:30)
Soy un camión
Cobrar peaje camión
```

4 Elementos claves para entender POO en Java

Salida por Consola



The screenshot shows an IDE window with a Java file named 'Test.java'. The code defines a package, imports a class, and creates a 'Test' class with a 'main' method. The 'main' method creates instances of 'Auto', 'Moto', 'Bus', and 'Camion', and then creates a 'Peaje' object to process the 'auto' instance. The console output at the bottom shows the execution results.

```
1 package com.ecodeup.test;
2
3 import com.ecodeup.polimorfismo.Auto;
4
5
6
7
8
9 public class Test {
10
11     public static void main(String[] args) {
12         // declaro los objetos que heredan de tipo vehiculo
13         Auto auto = new Auto();
14         Moto moto = new Moto();
15         Bus bus = new Bus();
16         Camion camion = new Camion();
17
18         // declaro el objeto peaje para identificar y cobrar el peaje de acuerdo
19         // al tipo de vehiculo
20         Peaje peaje = new Peaje();
21         // cobrar peaje para cualquier vehiculo
22         peaje.cobrarPeaje(auto);
23
24     }
25
26 }
```

Problems @ Javadoc Declaration Console Servers

<terminated> Test (2) [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (2 oct. 2016 19:55:12)

Soy un auto
Cobrar peaje auto