Java Herencia

Octavio Robleto

octavio.robleto@gmail.com

https://octaviorobleto.com

Herencia

- La idea de la herencia es permitir la creación de nuevas clases basadas en clases existentes.
- Cuando heredamos de una clase existente, reusamos (o heredamos) métodos y campos, y agregamos nuevos campos y métodos para cumplir con la situación nueva.
- Cada vez que encontremos la relación "es-un" entre dos clases, estamos ante la presencia de herencia.

Esto es una de las bases de la reutilización de código, en lugar de copiar y pegar.

Grafico

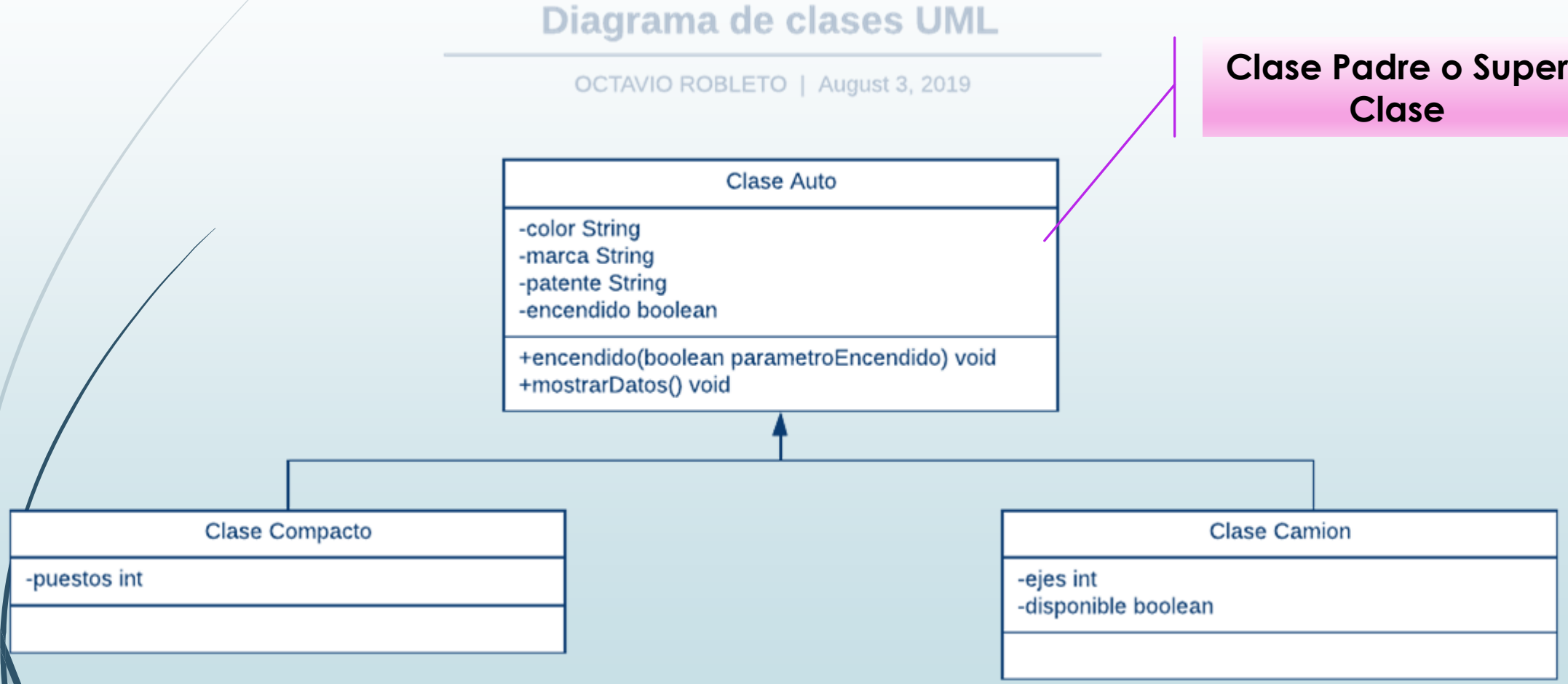
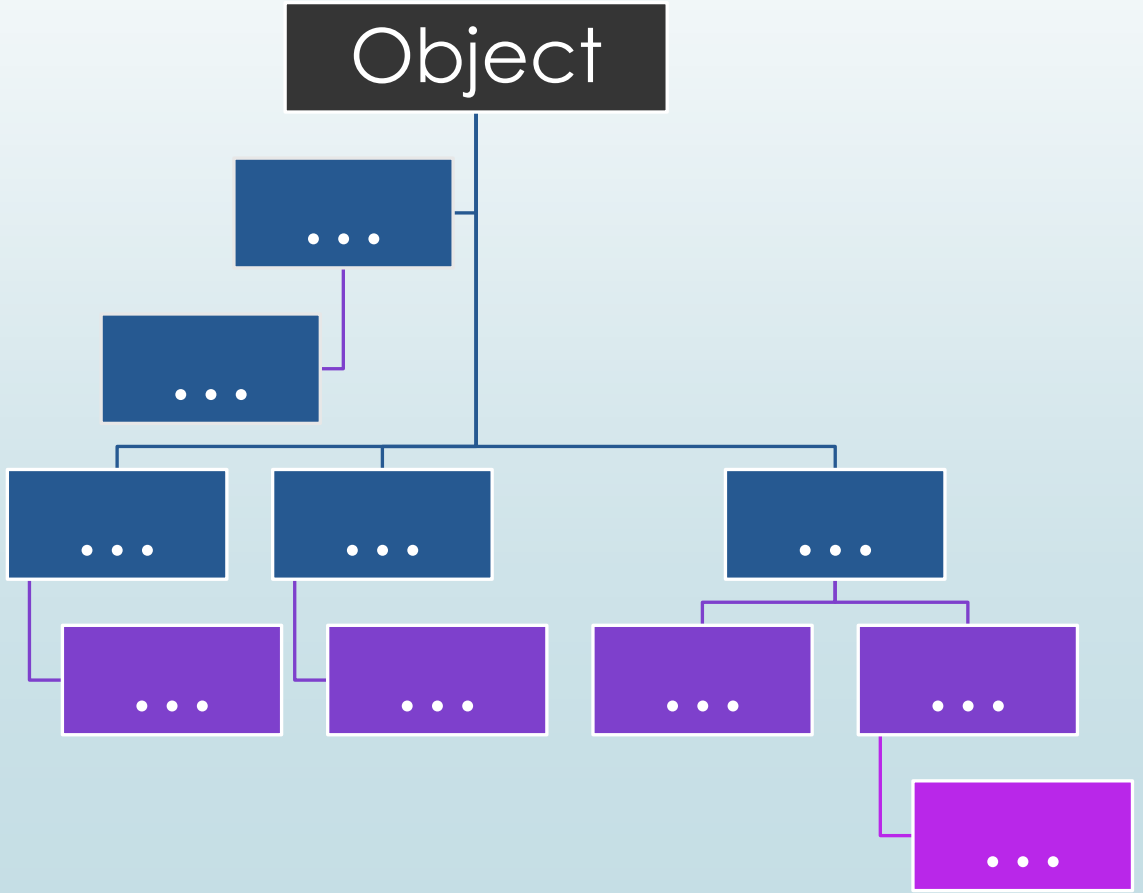


Diagrama de clases UML
OCTAVIO ROBLETO | August 3, 2019
Clase Padre o Super Clase
Clase Auto
-color String
-marca String
-patente String
-encendido boolean
+encendido(boolean parametroEncendido) void
+mostrarDatos() void
Clase Compacto
-puestos int
Clase Camion
-ejes int
-disponible boolean

Jerarquía de Herencia en JAVA

- Java no escapa en si de la herencia, lo que representa una gran ventaja a la hora de utilizar métodos y atributos propios de la API de JAVA ya que todas las clases heredan por defecto de la super clase **Object.**



Herencia en JAVA

- Java permite la herencia simple, que significa que solo podemos heredar de una sola clase, otros lenguajes de POO permiten la herencia múltiple.
- Para poder heredar se debe declarar después del nombre de la clase que creamos la palabra reservada **extends** seguido del nombre de la clase a heredar

```
public class Compacto extends Auto {
    private int puestos;

    public Compacto(String color, String marca, String patente, boolean encendido, int puestos) {
        super(color, marca, patente, encendido);
        this.puestos = puestos;
    }

    public Compacto() {
    }

    public int getPuestos() {
        return puestos;
    }

    public void setPuestos(int puestos) {
        this.puestos = puestos;
    }

}
```

Uso de los métodos y atributos heredados

```
// creamos o instanciamos los objetos
Auto auto1 = new Auto();
Compacto compacto1 = new Compacto(3, "Azul", "Ford", "ANZ-963", true);
Camion camion1 = new Camion(16, true, "Verde", "Mercedes Benz", "CAM-7896", false);
Compacto compacto2 = new Compacto();

// le damos valores a los atributos del auto 1
auto1.setColor("Rojo");
auto1.setMarca("Ferrari");
auto1.setPatente("ABC-188");

// le damos valores a los atributos del compacto 2
compacto2.setColor("Azul");
compacto2.setMarca("Mercedes Benz");
compacto2.setPatente("COM-89655");
compacto2.setPuestos(8);

// Encendemos el Auto en true y apagamos el Auto en false
auto1.encendido(true);

auto1.mostrarDatos();
compacto1.mostrarDatos();
compacto2.mostrarDatos();
camion1.mostrarDatos();
```

Creamos los objetos

Uso de los métodos heredados de la clase auto

Uso métodos propios de la clase

Que pasa con los constructores?

- Al heredar de una clase heredaremos tanto los atributos como los métodos, mientras que los constructores son utilizados, pero no heredados.
- Creamos nuestro constructor con los atributos que necesitamos y para usar el de la clase padre usamos la palabra reservada **super** con los argumentos necesarios.

```
public class Compacto extends Auto {
    private int puestos;

    public Compacto(String color, String marca, String patente, boolean encendido, int puestos) {
        super(color, marca, patente, encendido);
        this.puestos = puestos;
    }

    public Compacto() {
    }

    public int getPuestos() {
        return puestos;
    }

    public void setPuestos(int puestos) {
        this.puestos = puestos;
    }

}
```

Asignamos los valores a los atributos propios de la clase Compacto

Le pasamos a la clase padre por medio del super los valores de los atributos propios de Auto