# Uso del modificador *static*

Curso Java Básico

# Modificador static

- Antes de detallar el uso de la palabra reservada static hay que especificar qué es una variable de instancia y qué es un método de instancia.
- Una variable de instancia es un atributo perteneciente definido en la clase, de la misma forma un método de instancia es un método que se define en la clase.

# Modificador static

```
public class UsoDeStatic {
    int variable1;
    float variable2;
    String variable3;

    public void hacerCosas(){
        // código aquí
    }
}
```

Atributos
o "variables de instancia"

Método
de
instancia

# Variables de instancia

- Cuando un número de objetos es creado a partir de la misma clase cada uno tiene sus propias copias de las *variables de instancia* almacenadas en distintas localidades de memoria.

# Variables de instancia

- Supongamos una clase `Carro`, sus variables de instancia o métodos son `marca` y `color`.
- Cada objeto `Carro` tiene sus propios valores de estas variables.

# Variables de clase

- A veces es necesario tener variables que sean comunes a todos los objetos, esto se logra con el modificador `static`.

- Los atributos que tienen la palabra `static` en su declaración son llamados *atributos estáticos* o *variables de clase*.

```
static int numero;
```

# Variables de clase

- Cada instancia (objeto) de una clase **comparte** una variable de clase, la cual está en una localidad fija de memoria.
- **Cualquier objeto** puede modificar el valor de una variable de clase y los demás objetos verán ese cambio.

# Variables de clase

- Las variables de clase PUEDEN ser **modificadas** incluso si no se ha creado **ninguna** instancia (objeto) de la clase.

# Problema

- Queremos crear un número de objetos Carro y asignar a cada uno un número de serie, comenzando por 1 para el primer objeto.
- Este número es único para cada objeto por lo tanto debe ser una *variable de instancia*

# Problema

- También necesitamos un atributo para saber cuantos objetos Carro se han creado para saber que ID asignar al siguiente.
- Dicho atributo no está relacionado con un objeto individual, sino con la **clase entera**.
- Por esa razón necesitamos la *variable de clase* `numeroDeCarros`

# Clase Carro

```
public class Carro {
    //variables de instancia
    String marca;
    String color;
    //variable de clase
    static int numeroDeCarros = 0;
}
```

# Acceso a variables de clase

- Podemos acceder a las variables de clase utilizando el nombre de la clase punto (.) y después el nombre de la variable.

```
Carro.numeroDeCarros = 1;
```

# Programa Pt.1

```
public class Carro {
    //variables de instancia
    String marca;
    String color;
    int idCarro;
    //variable de clase
    static int numeroDeCarros = 0;

    public Carro(String marca, String color) {
        this.marca = marca;
        this.color = color;
        idCarro = ++numeroDeCarros;
    }
```

# Programa Pt.2

```
    public static void main(String[] args) {
        System.out.println("El número de autos es " + Carro.numeroDeCarros );
        Carro carrito = new Carro("Mustang","rojo");
        System.out.println("Se ha creado un mustang rojo");
        System.out.println("El número de autos es " + Carro.numeroDeCarros );
        Carro otroCarro = new Carro("BMW","azul");
        System.out.println("Se ha creado un BMW azul");
        System.out.println("El número de autos es " + Carro.numeroDeCarros );
        Carro convertible = new Carro("Mazda","negro");
        System.out.println("Se ha creado un Mazda negro");
        System.out.println("El número de autos es " + Carro.numeroDeCarros );
    }

}
```

# Ejecución del programa

El número de autos es 0
Se ha creado un mustang rojo
El número de autos es 1
Se ha creado un BMW azul
El número de autos es 2
Se ha creado un Mazda negro
El número de autos es 3

# Comentarios

- Como se puede observar en el programa la *variable de clase* aumenta cada que se crea un nuevo `Carro`, si fuera variable de instancia cada uno tendría su propia copia y no podríamos saber cuántos objetos se han creado.

# Métodos estáticos (static)

- De manera similar a las variables de clase existen los métodos de clase, también conocidos como métodos estáticos (*static methods*).
- Estos métodos pueden ser llamados sin necesidad de crear previamente un objeto de la clase que posee el método.

# La clase *Math*

- La clase Math posee diversos métodos estáticos que son de utilidad para realizar cálculos matemáticos.
- Es un buen ejemplo del uso de los métodos estáticos: cuando no es necesario crear objetos para realizar una tarea. De esta forma, los métodos pertenecen a la clase, no a una instancia en particular.

# Invocar métodos estáticos

- La sintaxis para invocar métodos estáticos es:

  NombreClase.*nombreMetodo*();

Po ejemplo:

```
public class Test {
    public static void main(String[] args) {
        // Valor absoluto
        System.out.println(Math.abs(-100));
        // Función seno
        System.out.println(Math.sin(Math.PI / 2));
        // Raíz cuadrada
        System.out.println(Math.sqrt(9.0));
    }
}
```

# Consideraciones del modificador *static*

- Un método estático NO puede acceder a variables de instancia (no-estáticas)

```
public class Foo {
    int i = 42;
    public static void metodo(){
        i = 43;
    }
}
```

- Un método estático NO puede invocar un método no estático

```
class Bar{
    void metodo1(){}

    static void metodo2(){
        metodo1();
    }
}
```

# Consideraciones del modificador *static*

- Un método estático puede acceder a variables y métodos estáticos

```
class Bar{
    static void metodo1(){}
    static int i = 0;
    static void metodo2(){
        metodo1();
        i = 2;
    }
}
```

- Un método no estático puede acceder a variables y métodos estáticos

```
class Clase{
    static void hacerCosas(){}
    static int num;
    void hacerMasCosas(){
        hacerCosas();
        num++;
    }
}
```