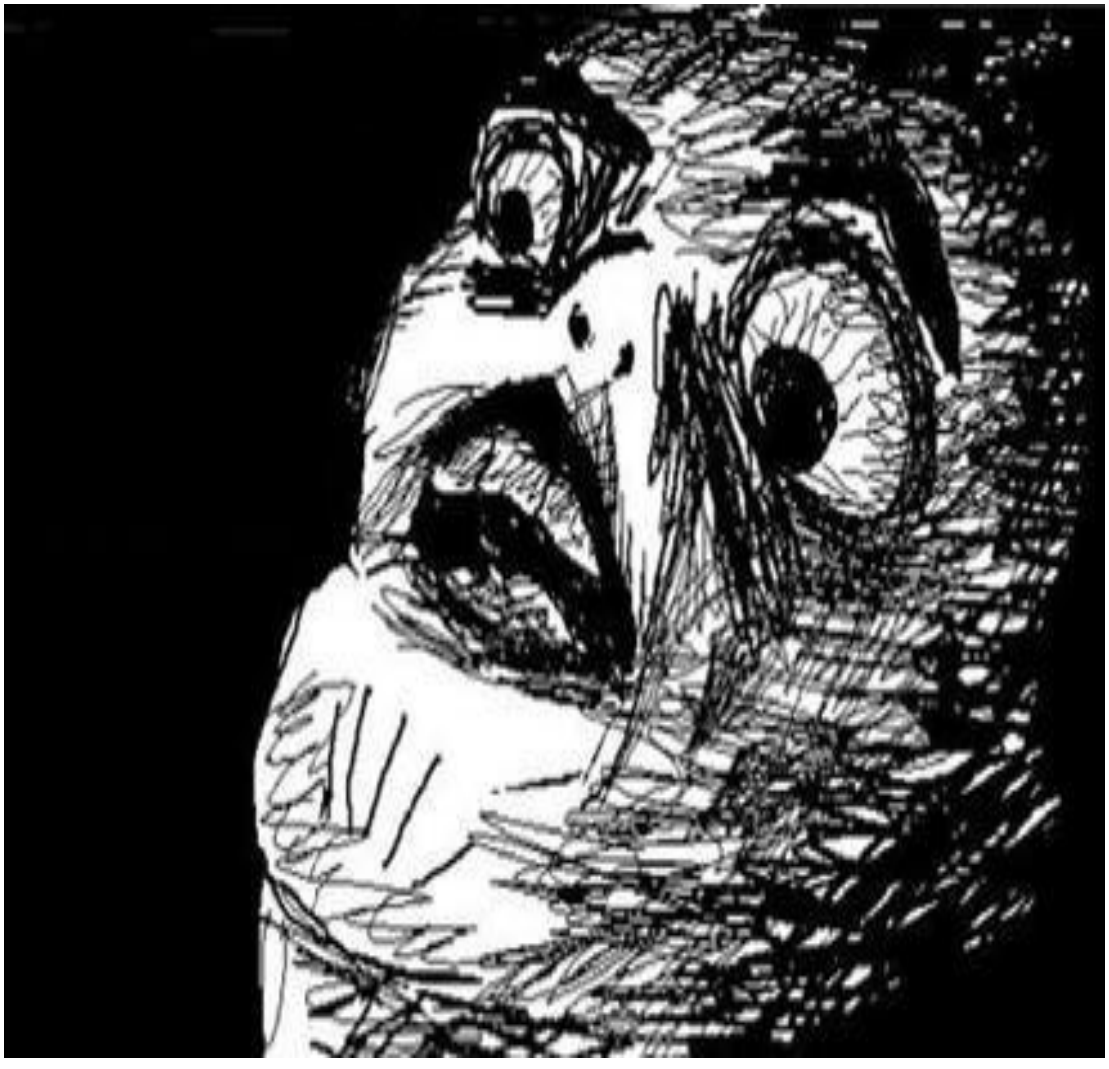
static & non-static

Programación y Laboratorio III

Métodos que no usan valores variables de instancia.

No es necesario crear una instancia de la clase.



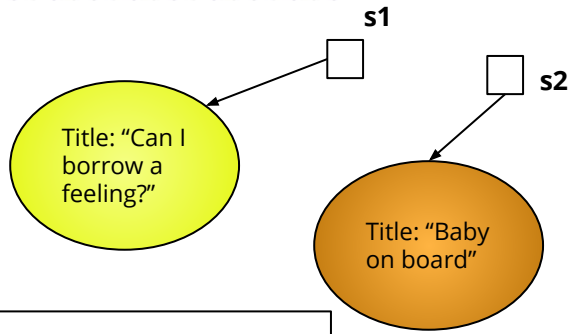
Agenda

- **Método regular (non-static)**
- Método estático
- Métodos regulares vs. estáticos
- Variables estáticas
- Constantes
- `final`

Método regular (non-static)

Variable de instancia

```
public class Song {
      private String title;

      public Song(String t) {
          title = t;
      }

      public void play() {
          SoundPlayer player = new SoundPlayer();
          player.playSound(title);
      }
}
```

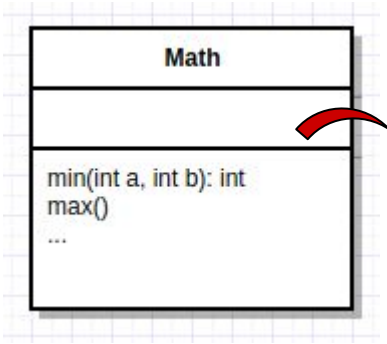



```
Song s1 = new Song();
s1.play();
Song s2 = new Song();
s2.play();
```

Agenda

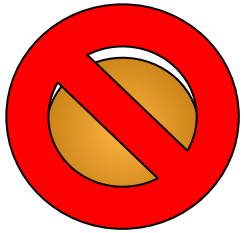
- ~~Método regular (non static)~~
- **Método estático**
- Métodos regulares vs. estáticos
- Variables estáticas
- Constantes
- `final`

Método estático



No hay variables de instancia

```
public static int min(int a, int b) {
    //return a or b
}
```



No hay instancias de objetos

```
int min = Math.min(42, 39);
```

Agenda

- ~~Método regular (non static)~~
- ~~Método estático~~
- **Métodos regulares vs. estáticos**
- Variables estáticas
- Constantes
- `final`

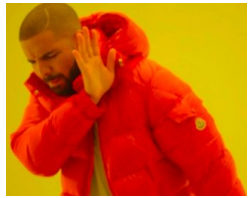
Métodos regulares (non-static) vs. estáticos

- Un método declarado con la palabra reservada `static` nos indica que se puede invocarlo sin necesidad de crear una instancia de la clase.
- Se pueden combinar métodos regulares y estáticos en la misma clase.
- Los métodos estáticos no pueden usar variables de instancia.
- Los métodos estáticos no pueden usar métodos regulares, porque usan variables de instancias.

Ejemplos

```
public class Duck {
      private int size;

      public static void main (String[] args) {
          System.out.println(“Size = ” + size);
      }

      public void setSize(int s) {
          size = s;
      }

      public int getSize() {
          return size;
      }
}
```

```
public class Duck {
      private int size;

      public static void main (String[] args) {
          System.out.println(“Size = ” + getSize());
      }

      public void setSize(int s) {
          size = s;
      }

      public int getSize() {
          return size;
      }
}
```



Agenda

- ~~Método regular (non static)~~
- ~~Método estático~~
- ~~Métodos regulares vs. estáticos~~
- **Variables estáticas**
- Constantes
- `final`

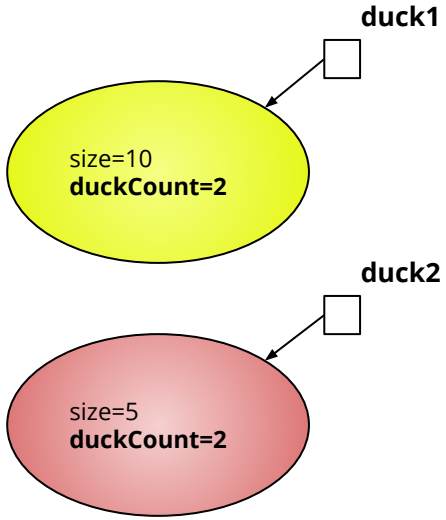
Variables estáticas

"Un valor compartido por todas las instancias una clase"

Variables estáticas

```
public class Duck {
      private int size;
      private static int duckCount = 0;

      Public Duck() {
         duckCount++;
      }

      public void setSize(int s) {
          size = s;
      }

      public int getSize() {
          return size;
      }
}
```

Se inicializa una única vez, cuando la clase de carga por primera vez.



Agenda

- ~~Método regular (non static)~~
- ~~Método estático~~
- ~~Métodos regulares vs. estáticos~~
- ~~Variables estáticas~~
- **Constantes**
- `final`

Constantes

```
public static final double PI = 3.141592653589793
```

- La palabra reservada `final` indica que una vez inicializada, el valor de la variable no puede cambiar.
- Generalmente se establecen como `public` para que puedan ser accedidas desde cualquier lugar de nuestro código.
- Son estáticas para que no sea necesario crear una instancia de la clase para poder usarlas.
- EL NOMBRE DE UNA CONSTANTE DEBE ESTAR EN MAYÚSCULA.

Agenda

- ~~Método regular (non static)~~
- ~~Método estático~~
- ~~Métodos regulares vs. estáticos~~
- ~~Variables estáticas~~
- ~~Constantes~~
- `final`

`final`

- La palabra reservada `final` no es sólo para variables estáticas.
- Se puede usar final para variables de instancias, variables locales, parámetros de métodos y clases.
- Indica que el valor no puede cambiar una vez que fue inicializado.

`final`

- Una variable `final` significa que no puede cambiar su valor.
- Un método `final` significa que no puede sobreescribirse.
- Una clase `final` significa que no puede tener subclases.

Bibliografía oficial

- https://docs.oracle.com/javase/tutorial/java/javaOO/classvars.html
- https://docs.oracle.com/javase/tutorial/java/javaOO/initial.html
- https://docs.oracle.com/javase/8/docs/technotes/guides/language/static-import.html
- https://docs.oracle.com/javase/specs/jvms/se8/html/jvms-2.html#jvms-2.5