# Funciones de orden superior

## Forzando Java

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

12 de enero de 2021

# Definiciones

# Temas

1

# Función de orden superior

**Definición (Función de orden superior)**

"Se dice que una función es de *orden superior* cuando recibe **funciones** como parámetros o devuelve otra función como resultado." Maurizio Gabbrielli 2010

# Temas

# ¿Funciones de orden superior en Java?

- Java es un lenguaje orientado a objetos y los mensajes sólo permiten el uso de objetos o tipos primitivos como parámetros o resultados.
- Si queremos recibir o enviar funciones como parámetros, el truco es envolver una función dentro de un objeto.
- Las habilidades de `Java` entonces son limitadas pues no se trata de un diseño natural, pero son suficientes para varias aplicaciones.

# Interfaces funcionales

- Java introduce el concepto de *interfaz funcional* para referirse al mecanismo mediante el cual se define una interfaz con el nombre del único método que queremos pasar como parámetro o devolver como resultado.
- Lo que realmente se transfiere es un objeto instancia de la interfaz funcional indicada.
- Para ejecutar a la función, se manda llamar al método correspondiente del objeto recibido.

# Mecanismos

# Temas

# Implementando la interfaz

Código 1: Interfaz e implementación

```
1  public interface Operación {
2    public int opera(int a);
3  }
4
5  public class Suma implements Operación {
6    private int val;
7    public Suma(int val) { this.val = val; }
8    public int opera(int a) {
9      return this.val + a;
10   }
11 }
12
13 public class Uso {
14   public static void hazOperación(Operación o) {
15     System.out.println("Operando: " + o.opera(5));
16   }
17
18   public static void main(String[] args) {
19     Operación suma = new Suma(0);
20     hazOperación(suma);
21   }
22 }
```

# Clases internas

Código 2: Clase interna

```
1  public interface Operación {
2    public int opera(int a);
3  }
4
5  public class Uso {
6    private static class Suma implements Operación {
7      private int val;
8      public Suma(int val) { this.val = val; }
9      public int opera(int a) {
10         return this.val + a;
11     }
12   }
13
14   public static void hazOperación(Operación o) {
15     System.out.println("Operando: " + o.opera(5));
16   }
17
18   public static void main(String[] args) {
19     Operación suma = new Suma(0);
20     hazOperación(suma);
21   }
22 }
```

# Clases anónimas internas

Código 3: Clase anónima

```
1  public interface Operación {
2    public int opera(int a);
3  }
4
5  public class Uso {
6    public static void hazOperación(Operación o) {
7      System.out.println("Operando: " + o.opera(5));
8    }
9
10   public static void main(String[] args) {
11     Operación potencia = new Operación() {
12       private int val = 1;
13         @Override
14         public int opera(int a) {
15           val = val * a;
16           return val;
17         }
18     };
19     hazOperación(potencia);
20   }
21 }
```

# Lambdas

Código 4: Expresión lambda

```
1  public class Lambdas {
2      private interface Operación {
3          public int opera(int a);
4      }
5
6      public static void hazOperación(Operación o) {
7          System.out.println("Operando: " + o.opera(5));
8      }
9
10     public static void main(String[] args) {
11         Operación o = (a) -> a * a;
12         hazOperación(o);
13     }
14 }
```

# Temas

## Código 5: Regresando una función

```
1  public class RegresaObjetoFunción {
2
3      @FunctionalInterface
4      private interface Función {
5          public String aplica(int valor);
6      }
7
8      private String atributo;
9
10     public Función crea(int valor) {
11         return new Función() {
12             @Override
13             public String aplica(int argumento) {
14                 return "Atributo = " + atributo +
15                         ", valor local al crear = " + valor +
16                         ", argumento = " + argumento;
17             }
18         };
19     }
20
21     public void atributo(String a) {
22         atributo = a;
23     }
24
25     public static void main(String[] args) {
26         RegresaObjetoFunción fábrica = new RegresaObjetoFunción();
```

```
27
28          fábrica.atributo("uno");
29          Función f1 = fábrica.crea(1);
30          System.out.print("Uno:␣␣␣␣␣␣␣␣␣␣");
31          System.out.println(f1.aplica(10));
32
33          fábrica.atributo("dos");
34          Función f2 = fábrica.crea(2);
35          System.out.print("Dos:␣␣␣␣␣␣␣␣␣␣");
36          System.out.println(f2.aplica(20));
37
38          System.out.print("Uno␣otra␣vez:␣");
39          System.out.println(f1.aplica(11));
40      }
41  }
```

# Ejecución

Código 6: Expresión lambda

```
1 Uno:           Atributo = uno, valor local al crear = 1, argumento = 10
2 Dos:           Atributo = dos, valor local al crear = 2, argumento = 20
3 Uno otra vez: Atributo = dos, valor local al crear = 1, argumento = 11
```

# Ejemplo completo

# Funciones de orden superior

Código `FuncionesDeOrdenSuperior.java`

# Bibliografía I


Maurizio Gabbrielli, Simone Martini (2010). *Programming Languages: Principles and Paradigms*. Springer. 440 págs. ISBN: 978-607-707-211-9. DOI: 10.1007/978-1-84882-914-5.

# Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual