escuelaingenieríaindustrial
PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO

POO en Java III: sobrecarga de constructores y métodos

Franco Guidi Polanco
Escuela de Ingeniería Industrial
Pontificia Universidad Católica de Valparaíso, Chile
fguidi@ucv.cl

Sobrecarga de constructores

- Una clase puede tener más de un constructor.
- Los constructores se diferencian por cantidad, tipo y orden de parámetros.

  Ejemplo: constructores distintos de la clase `Observación`

```
public Observación(){…
public Observación(int a){…
public Observación(int a, double b){…
public Observación(double a, int b){…
```

- Esto permite instanciar objetos considerando distintos tipos de datos disponibles.

Sobrecarga de constructores (cont.)

```
public class Valor {
    private int x;
    private int y;

    public Valor(){
       x = 0;
       y = 0;
    }
    public Valor(int a, int b){
       x = a;
       y = b;
    }
    ...
}
```

Constructor sin parámetros

Constructor con dos parámetros enteros

Sobrecarga de constructores (cont.)

```
public class Valor {
  private int x;
  private int y;

  public Valor(){
     x = 0;
     y = 0;
  }
  public Valor(int a, int b){
     x = a;
     y = b;
  }
  ...
}
```

Clase Valor

```
public class Ejemplo {
 public static void main...

   Valor ob1, ob2;
   ob1 = new Valor( 3, 5 );
   ob2 = new Valor();

   //Las siguientes fallan:
   Valor ob3, ob4;
   ob3 = new Valor( 2 );
   ob4 = new Valor( 2.0, 3.0);
   ...
 }
}
```

Aplicación

No existe el constructor de un parámetro **int**

No existe el constructor de dos parámetros **double**

Sobrecarga de métodos

- Una clase puede tener más de un método con el mismo nombre.
- Los métodos se diferencian por nombre del método, y cantidad, tipo y orden de sus parámetros. Todo esto constituye la "firma del método" (*method signature*).

Ejemplo: métodos distintos de una clase

```
public double sumaTiempo(){...
public double sumaTiempo(int a){...
public double sumaTiempo(double a){...
public double sumaTiempo(int a, double b){...
public double sumaTiempo(double a, int b){...
```


Sobrecarga de métodos

- IMPORTANTE: el tipo de valor retornado **no forma parte** de la "firma del método" (no es utilizado para distinguir entre métodos).

```
public double sumaTiempo(int a){...
public float  sumaTiempo(int b){...
public int    sumaTiempo(int a){...
```

Java no distingue entre ellos: genera un **error de compilación**.

Sobrecarga de métodos (cont.)

```
public class Persona {
  private String nombre;
  private int edad;

  public void setEdad(int x){
     edad = x;
  }
  public void setEdad(double x){
     edad = (int) x;
  }
  ...
}
```

Método **setEdad(int)**

Método **setEdad(double)**

Sobrecarga de métodos (cont.)

```
public class Persona {

  private String nombre;
  private int edad;

  public void setEdad(int x){
     edad = x;
  }

  public void setEdad(double x){
     edad = (int) x;
  }
  ...
}
```

Clase Persona

```
public class Ejemplo {

  public static void main...

    Persona p1;
    p1 = new Persona();
    …

    P1.setEdad( 35 );
    p1.setEdad( 35.0 );

   …
  }
}
```

Aplicación

Sobrecarga de métodos y promoción de argumentos

```
public class Valor {

  private int dato;

  public void setEdad(short x){
     dato = x;
  }

  public void setEdad(int x){
     dato = x;
  }

  public void setEdad(double x){
     edad = (int) x;
  }
  ...
}
```

Clase Valor

```
public class Ejemplo {
 public static void main...

   Valor v = new Valor();
   ...

   v.setEdad( 35 );
   v.setEdad( 35.0 );

   // Aquí hay promoción:

   byte b = 3;
   v.setEdad( b );
   v.setEdad( 35f );
   ...
 }
}
```

Aplicación

Estructura general de una clase

```
public class IdentificadorClase {

    declaración variables de instancia

    declaración constructor 1 {cuerpo método 1 }
    declaración constructor 2 { cuerpo método 2 }
    declaración constructor n { cuerpo método n }

    declaración método 1 {cuerpo método 1 }
    declaración método 2 { cuerpo método 2 }
    declaración método m { cuerpo método m }

}
```


Acerca de clases y métodos

- Para listar miembros de una determinada clase se puede usar el desensamblador del SDK: **javap**

Formato: javap <opciones> <NombreClase>

- **Algunas opciones:**
  - `-public`: **sólo elementos declarados public.**
  - `-protected`: **elementos declarados public o protected.**
  - `-private`: **todos los elementos.**

```
C:\ejemplo>javap CuntaCorriente
```

```
C:\ejemplo>javap -private java.lang.String
```