Sobrecarga: Constructores y Métodos

Dr. (c) Noé Alejandro Castro Sánchez

Constructores

- Estructuras similares a los métodos
- Se invocan automáticamente cuando se construye un objeto (new)
- Se usan generalmente para inicializar las variables de instancia (atributos)
- El nombre del constructor debe ser igual al nombre de la clase declarada

Constructores (2)

- No retorna ningún tipo de dato (ni void)

```
public class Cat{
    private int peso;
    private String nombre;

    public Cat (String nombre1, int peso1)
    {
        nombre = nombre1;
        peso = peso1;
    }
    …
}
```

Constructores (3)

- Invocación del constructor:

```
public class TestCat{
    public static void main(String[] a){
        Cat c = new Cat(“Felix”, 10);
        Cat c2 = new Cat(“Garfield”, 20);
        …
    }
}
```

Sobrecarga de métodos

- Varios métodos con el mismo nombre
  - Diferente número de argumentos
  - Diferente tipo de argumentos
- Proporciona varias interfaces bajo el mismo nombre

Ejemplo de sobrecarga de métodos

```
public class Calculadora {
  public int sumar(int x, int y) {
      System.out.println("Método1:");
      return x + y;
  }
  public float sumar(float x, float y) {
      System.out.println("Método2:");
      return x + y;
  }
  public float sumar(int x, float y) {
      System.out.println("Método3:");
      return x + y;
  }
}
```

Ejemplo de sobrecarga de métodos II

```
public class TestCalculadora {
  public static void main (String[] args){
      Calculadora c = new Calculadora();
      int tot1 = c.sumar(2,4);
      System.out.println(tot1);
      float tot2 = c.sumar(2.3f,4.2f);
      System.out.println(tot2);
      float tot3 = c.sumar(2,4.2f);
      System.out.println(tot3);
  }
}
```

Palabra reservada *this*

- Es una referencia al objeto actual, usándose:
  - dentro de métodos de instancia
  - dentro de un constructor
- Se usa para hacer referencia a cualquier miembro del objeto actual
- Se usa para diferenciar variables de instancia de variables locales
- Para invocar un constructor dentro de otro constructor

Ejemplo sobrecarga

```
public class Box {
  int x, y, ancho, alto;
  public Box(int x, int y, int ancho, int alto) {
      this.x = x;
      this.y = y;
      this.ancho = ancho;
      this.alto = alto;
  }
  public Box(int x, int y) { this(x,y,10,10); }
  public Box() {
      this (1,1);
  }
  public void setAncho(int ancho) {
      this.ancho = ancho;
  }
}
```

Ejemplo sobrecarga (II)

```
public class TestBox {
  public static void main (String [] args)
  {
      Box a= new Box(5,3,100,200);
      Box b= new Box(10,5);
      Box c= new Box();
  }
}
```

Ejercicio 1

- Agregar el método setDatos a la clase Cat. Sobrecargar dicho método para que asigne datos de varias formas según los argumentos:
    1. Acepte nombre y peso
    2. Acepte el nombre
    3. Acepte el peso
    4. Acepte peso y nombre

Ejercicio 2

- Modificar la clase *PCuadrado* y agregar los siguientes constructores:
- Constructor vacío que imprima la leyenda "*PCuadrado creado por omisión:*"
  - dimensiones por omisión serán 5 x 5
- Constructor que acepte uno de los lados como argumentos
- En la clase *TestPCuadrado* crear varios objetos vacíos y otros con argumentos para demostrar el uso de los constructores

Ejercicio 3

- Realizar el diagrama de clase en UML para números complejos,
- Implementar el comportamiento abajo indicado para la clase *Complejo* y demostrar su uso con la clase *TestComplejo*.
- Use constructores para inicializar los datos.
- Comportamiento:
  - Sumar
  - Restar
  - Multiplicar

Ejercicio 3 (II)

- $z = a + bi$

  a: representa la parte real

  b: representa la parte imaginaria

- Sean *w, z* números complejos, entonces:

  $w = a + bi$ y **$z = c + di$**

  $t = w + z = (a+c) + (b+d)i$

  $t = w - z = (a-c) + (b-d)i$

  $t = w * z = (ac-bd) + (ad-cb)i$

Ejercicio 3 (III)

- Ejemplo de uso

  Sean w= $2+3i$ y z=$4+2i$

  …

```
t = w.sumar(z);
t.imprime();

// Complejo t = 6 + 5i
```