Programación Orientada a Objetos en Java

Curso 2006 - 2007

Tema 3 – Clases y Objetos

Gonzalo Méndez Pozo
Dpto. de Ingeniería de Software e Inteligencia Artificial
Universidad Complutense de Madrid

Clases y Objetos

- Programación Estructurada:
  - Tipos Abstractos de Datos (TAD)
- Programación Orientada a Objetos:
  - Clases y objetos
- Las clases y los TAD no son equivalentes, aunque se puede establecer cierta analogía


Objetos

- Son la parte ejecutable de la programación orientada a objetos
- Se manejan a través de variables
- Pertenecen a una clase


Clases

- Son la plantilla a partir de la cual se crean los objetos
- Formadas por:
  - Nombre
  - Atributos
  - Métodos


Clases

- En general, interesa que los atributos no se puedan tocar directamente desde fuera de la clase → privados
- Los métodos son la forma de comunicarse con la clase para pedirle que haga cosas (servicios) → públicos o privados
  - Los métodos get y set “hacen trampa”


Errores de la POO

- Clase ≠ Tipo
- Atributos ≠ Estado
- Métodos ≠ Comportamiento
- Los objetos no imitan la realidad
- Los objetos no son funciones + datos
  - Objeto = cosa

Declaración de Clases en Java

```
public class Circulo
{
    private int centro_x, centro_y, radio;

    public void paint (Graphics g)
    {
        g.drawCircle (centro_x, centro_y, radio,
                      Color.GREEN);
    }
}
```


Declaración de Clases en Java

- Hay que declarar la visibilidad de la clase, sus atributos y sus métodos
- En cada fichero
  - una clase pública con el mismo nombre del fichero
  - 0..n clases privadas


Uso de Objetos

- Primero es necesario declarar una variable perteneciente a la clase:
  - Circulo mi_circulo;
- Después hay que crear el objeto:
  - mi_circulo = new Circulo();
- Ahora ya se puede llamar a los métodos del objeto:
  - mi_circulo.paint();


Constructor

- Método especial que indica lo que se hace cuando se crea un objeto
- Tiene el mismo nombre que la clase
- Pueden existir varios constructores con distintos parámetros
- Si no declaramos ninguno se usa el constructor por defecto
- Si declaramos alguno es obligatorio declarar también el constructor por defecto

Constructor

```
Public class Circulo
{
    int c_x, c_y, radio;

    Circulo (int x, int y)
    {
        c_x = x;
        c_y = y;
        radio = 1;
    }

    Circulo (int x, int y, int r)
    {
        c_x = x;
        c_y = y;
        radio = r;
    }

    Circulo()
    {
        c_x=c_y=radio=1;
    }
}
```


Un objeto especial: this

- La palabra reservada this sirve para que un objeto haga referencia a sí mismo
- Usos:
  - Un objeto se pasa a sí mismo como parámetro al llamar a un método de otro objeto
  - Especificar que un objeto utiliza sus métodos o sus atributos: no es obligatorio, pero a veces es necesario


Un objeto especial: this

```
public class Circulo
{
    public void paint ()
    {
        Ventana.paint(this);
    }
}
```

```
public class Circulo
{
    int x,y,r;

    Circulo(int x, int y, int r)
    {
        this.x = x;
        this.y = y;
        this.r = r;
    }
}
```



Métodos

- De manera general, la declaración de un método en java es de la forma

```
modo_acceso modificadores tipo_retorno nombre_metodo (argumentos)
{
    Cuerpo del método
}
```

- Donde
  - modo_acceso: public, private, protected
    - No es obligatorio especificarlo, pero es conveniente
  - Modificadores: static, abstract, final, native, synchronized
    - No es obligatorio usarlos
  - El tipo de retorno sí es obligatorio. Si no se devuelve nada se usa la palabra reservada void
  - Si no hay augumentos no se pone nada entre los paréntesis

Paso de Parámetros

- En java los parámetros siempre se pasan por copia
- Los argumentos de los tipos básicos no quedan modificados fuera del método aunque se modifiquen dentro
- En el caso de pasar objetos como parámetro, lo que se copia es una referencia al objeto
  - Si modificamos el objeto dentro del método también se modifica fuera

Valor de Salida de un Método

- Se indica con la palabra clave return seguida de lo que se quiere devolver

```
public int cuadrado (int x)
{
    return (x*x);
}
```

- Si se devuelve un void no es necesario utilizar return
- Es conveniente tener un solo return por función
- Hay que asegurarse de que, si se devuelve un valor, siempre se puede hacer el return (si no, da un error de compilación)

Valor de Salida de un Método

- Si lo que devolvemos es un objeto:
  - Se actúa igual que en el caso de devolver un valor de un tipo simple

Algunos Métodos Especiales

- toString:
  - Por defecto, Java convierte cualquier cosa en un String
  - Con objetos, si no le decimos cómo hacerlo, el String contiene la dirección del objeto
  - La implementación de toString indica cómo hacer la conversión

```
public String toString ()
{
    return ("("+c_x+","+c_y+")"+","+r);
}
```

Algunos Métodos Especiales

- equals:
  - No se debe usar == para comparar objetos, pues el resultado indica si los dos valores que se comparan son el mismo objeto, no si dos objetos son iguales

```
public boolean equals (Circulo c)
{
        return (this.c_x==c.c_x && this.c_y==c.c_y &&
                this.c_r==c.c_r);
}
```


Algunos Métodos Especiales

- clone:
  - Si a una variable le asignamos otra que contiene un objeto, ambas variables tienen una referencia al mismo objeto
  - Para que se asigne una copia del objeto hay que hacerlo a través del método clone

```
public Circulo clone()
{
        return new Circulo(this.c_x, this.c_y, this.c_r);
}
```


El Modificador static

- En atributos indica que son atributos de clase, es decir, que tienen el mismo valor para todos los objetos de la clase
  - Si un objeto modifica el valor, se modifica para todos los objetos
- En métodos indica que son métodos de clase, es decir, que se invocan sobre la clase sin necesidad de crear objetos.
  - Sólo pueden manejar atributos static
  - Ejemplos: método main, métodos de la clase Math


Ejercicios

- Implementar la clase Complejo

```
public class Complejo
{
      // atributos, representan el número x+yi
      private double x,y;
      // constructores
      public Complejo() { }
      public Complejo(double x, double y) { }
      // métodos
      public void ponX(double nuevaX) { }
      public void ponY(double nuevaY) { }
      public double valorX() { }
      public double valorY() { }
      public double módulo() { }
 }
```

Ejercicios

- Implementar los siguientes métodos:
  - Calcular el conjugado
    - Modificando el objeto
    - Sin modificar el objeto
  - toString
  - equals
  - clone
- Implementar la clase PruebaComplejos con el método main para probar el funcionamiento de la clase Complejo