Introducción a la programación orientada a objetos

Curso de Programación en Java

Luis Guerra
l.guerra@upm.es

Enero 2012

Contenidos

- Fundamentos de la POO
- Comportamiento y estado
- Clases y objetos en Java

Fundamentos de la POO

Introducción

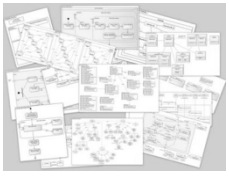
- Un paradigma es una forma de afrontar la construcción de código software
  - No hay paradigmas mejores ni peores
  - Todos tienen sus ventajas e inconvenientes
- Hay distintos paradigmas: POO, Estructurado, Funcional, Lógico, etc

Programación orientada a objetos

- Facilidad de diseño y relación con el mundo real
- Reusabilidad y facilidad de mantenimiento
- Sistemas más complejos
  - Abstracción
  - Trabajo en equipo
- Del lenguaje máquina hacia el mundo real
- Resuelve problemas complicados. No está pensado para tareas sencillas

UML (Unified Modeling Language)

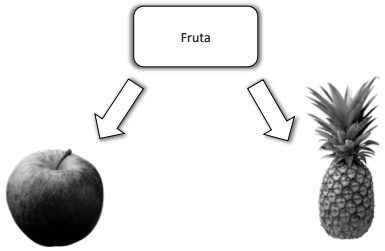
- Lenguaje unificado de modelado
- "Planos" de la aplicación. No sirve para desarrollar, sino para describir (análisis y diseño)
- Se utilizan diferentes diagramas. (13 tipos en UML 2.0)



Programación orientada a objetos

- Objetos: Elementos que interactúan entre sí para conseguir un fin
  - Autónomos
  - Con una funcionalidad concreta definida
  - Instancias de una clase
- Clases
  - Especificación de un conjunto de elementos
- Todo objeto pertenece a una clase

Objetos y clases



Objetos y clases

- Los objetos tiene propiedades que los distinguen
  - Comunes a todos los de su clase
  - Propias de cada uno
- Los objetos interactúan entre sí por medio de mensajes
- Los objetos representan los conceptos fundamentales del programa, y de su interacción surge la funcionalidad

Objetos y clases

- Clases
  - Representan conceptos o entidades significativas de un problema
  - Se pueden ver como plantillas para definir elementos (objetos)
  - Pueden estar directamente relacionadas unas con otras
- Objetos
  - Elementos con comportamiento definido en la clase y estado concreto
  - Instancias de clase
  - Interactúan por medio de mensajes

Otros conceptos importantes

- Encapsulamiento
- Polimorfismo
- Herencia
- …

Se verán mas adelante

Comportamiento y estado

Métodos y atributos

- Comportamiento: Describe los servicios que proporciona una clase (lo que se puede hacer con ella). Son los métodos.
  - El comportamiento distingue a los objetos de una clase de los de otra
- Estado: Describe el la situación interna del objeto. Existen varias formas de definir distintos estados. Son los atributos
  - El estado distingue a los objetos de la misma clase: cada uno tiene su estado

Atributos

- Describen el estado interno de cada objeto concreto
- Pueden ser:
  - Tipos básicos (int, boolean, short, etc)
  - Arrays de elementos
  - Otros objetos

Atributos y variables

- Atributo de instancia: Definido para las instancias de una clase. Una copia por objeto
- Atributo de clase: Definido para la clase. Una copia por clase (común para todos los objetos)

```
static int numero = 5;
```

- Variable local: Definida dentro del cuerpo de un método. De ámbito restringido

Métodos

- Definen el comportamiento de los objetos de una clase
- Devuelven un resultado
- Pueden necesitar parámetros

```
tipo_retorno nombre_método (parámetros) {
      código del método
}
```

Métodos habituales

- Constructor
  - Se ejecuta al crear un objeto
  - Sirve para inicializar un objeto al crearlo
  - Existe sobrecarga (distintos parámetros) (para cualquier método)
  - Coincide con el nombre de la clase y no devuelve nada por definición (no se indica el tipo de retorno)

Métodos habituales

- Destructor
  - Se ejecuta al crear destruir un objeto
  - No se suele usar: Java destruye los objetos automáticamente (*Recolector de Basura*)
  - Se tiene que sobrescribir
  - No devuelve nada por definición

```
void finalize()
```

Métodos habituales

- Punto de entrada
  - Método que se invoca al comienzo del programa
  - Al menos una clase del programa debe tenerlo (clase principal)
  - Solo se ejecuta el de la clase principal
  - Varias clases pueden tener punto de entrada, pero solo se ejecutará el de una de ellas

```
public static void main(String[] args)
```

Uso de objetos e interacción

- Los objetos son instanciaciones de las clases. Es necesario realizar dicha instanciación para poder usarlos
- Se comunican entre si mediante paso de mensajes
- La invocación de los métodos es la manera en la que se realiza el paso de mensajes

Clases y objetos en Java

Declaración de una clase

```
public class NombreClase {

	// Atributos

	tipo atributo1;
	tipo atributo2;
	…

	// Métodos

	tSalida nombreMetodo1 (params) {
		// Contenido método 1
	}
	tSalida nombreMetodo2 (params) {
		// Contenido método 2
	}
	…
}
```

Declaración de una clase

```
public class Perro {

	// Atributos

	String nombre;
	int edad;

	// Métodos

	Perro(String nom) {
		this.nombre = nom;
	}
	void ladrar() {
		System.out.println("¡Guau!");
	}
}
```

Creación de objetos

- Para poder usar un objeto hay que crearlo:

```
clase identificador = new clase();

Perro miPerro = new Perro("Pancho");
```

Operador .

- El operador . (punto) permite acceder a los miembros de una clase.
- Se usa tanto para atributos como para métodos.

```
Perro miPerro = new Perro("Pancho");
miPerro.edad = 5;
miPerro.ladrar();
```

this

- Para referirse a un objeto desde si mismo, se usa la palabra reservada this

```
public class Perro {
      String nombre;
      Perro(String nom) {
            this.nombre = nom;
      }
}
```

Uso de objetos

- Una vez creado (instanciado) un objeto, podemos enviarle mensajes y modificar su estado
  - Invocando métodos
  - Accediendo a sus atributos (no se debe hacer)

```
Perro miPerro = new Perro("Pancho");
miPerro.edad = 5;
miPerro.ladrar();
```

Acceso directo a atributos

- El acceso directo a atributos no es recomendable, porque revela la estructura interna del objeto
  - Los métodos definen como se debe usar un objeto. Permitir el acceso a los atributos es "dar vía libre" a manipular su estado sin control
- En su lugar se deben definir métodos que, de forma genérica, modifiquen el estado de acuerdo a consideraciones semánticas
- Se debe prohibir el acceso a los atributos

Cohesión y acoplamiento

- Cohesión: Grado de relación entre las diferentes características de un objeto
  - Objeto con una función muy bien definida: muy cohesionado
  - Objeto con muchas funciones no relacionadas: poco cohesionado
- Acoplamiento: Grado de dependencia entre diferentes objetos
  - Programa en el que todas las clases tiene funciones distintas y definidas: Poco acoplado
  - Programa en el que todas las clases participan en todas las tareas: muy acoplado
- Se debe **maximizar la cohesión** de cada objeto y **minimizar el acoplamiento** entre clases distintas

Encapsulación

- Puede (y suele) haber distintos niveles de visibilidad:
  - public: se puede acceder desde cualquier lugar
  - private: sólo se puede acceder desde la propia clase
  - protected: sólo se puede acceder desde la propia clase o desde una clase que herede de ella
- De esta forma se controla qué cosas son modificables y cómo se pueden modificar
- El estado (atributos) suele ser privado, y se suele modificar a través del comportamiento (métodos).

Visibilidad

- La visibilidad se puede indicar usando las palabras reservadas public, private y protected al comienzo de una declaración

```
public String nombre;

private int edad = 10;

public void ladar() {

      System.out.println("¡Guau!");

}
```

- Si no se indica visibilidad, por defecto los atributos son privados y los métodos públicos