UNIVERSIDAD EL BOSQUE

Fundamentos de programación

Clase 8

Agenda

- Asignación de responsabilidades
  - Técnica del experto
  - Técnica de descomposición de requerimientos
- Constantes
- Valores inmutables
- Asociaciones opcionales

Asignación de Responsabilidades

- Cada Clase debe tener a su cargo una serie de servicios y responsabilidades sobre la información que posee.
  - Técnica del experto
  - Técnica de descomposición de los requerimientos

Técnica del experto

- El dueño de la información es el responsable de ella, debe permitir que otros tengan acceso y puedan pedir que se cambie su valor. (no significa que por cada atributo siempre haya dos métodos, uno de retorno y uno de modificación)
- Define quién es responsable de hacer algo, pero son las reglas del contexto las que dicen cómo cumplir dicha responsabilidad.
- Para utilizar esta técnica se debe recorrer todos los atributos y asociaciones y definir los métodos.

Técnica de descomposición de los requerimientos

- Muchos requerimientos funcionales requieren más de un paso para poder resolverlos. Cada paso corresponde a la llamada a un método de alguna clase.
- Se debe descomponer cada requerimiento funcional en los sub-problemas que se deben resolver.
- Una vez identificados los servicios, se puede usar la técnica del experto para distribuir responsabilidades entre las clases.

Constantes

- Son nombres significativos que se pueden utilizar para asociar valores a opciones.
- Por convención se definen como atributos en mayúsculas y si tienen más de una palabra se separan con “_” i.e PRECIO_MAXIMO

```
//---------------------------------
//Constantes
//---------------------------------
public final static int TIPO_1 = 0;
public final static int TIPO_2 = 1;
public final static int TIPO_3 = 2;

//---------------------------------
//Atributos
//---------------------------------
private int tipo;
```

Valores inmutables

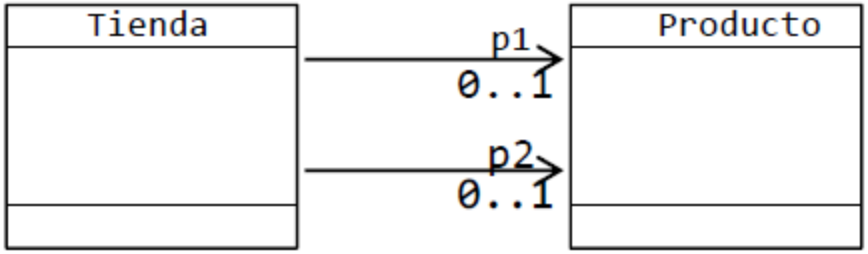
- Son nombres significativos que remplazan valores que no van a cambiar durante la ejecución del programa
- Es una buena práctica usar constantes especialmente para el mantenimiento de las aplicaciones

```
//--------------------------------
//Constantes
//--------------------------------
private final static double VALOR_1 = 0.54;
private final static int VALOR_2 = 10;
private final static String VALOR_3 = "VALOR";
```

Asociaciones opcionales

- Pueden haber asociaciones que dependiendo del estado del programa representan objetos que pueden o no existir.
- **Cardinalidad** representa el número de instancias de una clase que se pueden manejar a través de una asociación.
- Para una asociación que puede o no existir la cardinalidad se expresa 0..1
- Dentro de un método para indicar que el objeto no existe, se utiliza el valor **null**
- Cuando se intenta llamar a un método de un objeto que no existe, el compilador muestra el error de compilación **NullPointerException**

Asociaciones opcionales




```
p1 = null;
p2 = null;
```

¿PREGUNTAS?