DISEÑO ORIENTADO A OBJETOS

Prof. Esp. Ing. Agustín Fernandez

Patrones de Diseño Orientado a Objetos [GoF]

- **Christopher Alexander:** "cada patrón describe un problema que ocurre de manera recurrente en nuestro entorno, así como la solución a ese problema, de tal manera que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces"
- **Wikipedia**: Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Patrones de Diseño Orientado a Objetos [GoF]

- Para que una solución sea considerada un patrón debe poseer ciertas características:
- **Efectivo**: resolviendo problemas similares en ocasiones anteriores.
- **Reutilizable**: aplicable a diferentes problemas de diseño en distintas circunstancias.

Patrones de Diseño Orientado a Objetos [GoF]

- **Pretenden**:
  - Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
  - Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
  - Formalizar un vocabulario común entre diseñadores.
  - Estandarizar el modo en que se realiza el diseño.
  - Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento existente.
  - No reinventar la rueda.

Patrones de Diseño Orientado a Objetos [GoF]

- **No pretenden**:
- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

Patrones de Diseño Orientado a Objetos [GoF]

- **Recuerden**:
- No es obligatorio utilizar los patrones.
- Es aconsejable en el caso de tener el mismo problema o similar que soluciona el patrón.
- Abusar o forzar el uso de los patrones puede ser un error.

Patrones de Diseño Orientado a Objetos [GoF]

- **Antipatrón**: Un antipatrón de diseño es un patrón de diseño que conduce a una mala solución para un problema.
  - Evitar los antipatrones siempre que sea posible, requiere su reconocimiento e identificación dentro del ciclo de vida del software.
  - Patrón de diseño → Buen Camino.
  - Antipatrón de diseño → Mal Camino.

Patrones de Diseño Orientado a Objetos [GoF]

- **Tipos de patrones de diseño:**
  - **De creación**: Resuelven problemas relacionados con la creación de instancias de objetos.
  - **De estructura**: Se centran en problemas relacionados con la forma de estructurar las clases.
  - **De comportamiento**: Permiten resolver problemas relacionados con el comportamiento de la aplicación, normalmente en tiempo de ejecución.

Patrones de Diseño Orientado a Objetos [GoF]

- **Tipos de patrones de diseño:**
  - **De creación**:
    - Abstract factory
    - Factory method
    - Singleton
    - Prototype
    - Builder

Patrones de Diseño Orientado a Objetos [GoF]

- **Tipos de patrones de diseño:**
  - **De estructura**:
    - Adapter
    - Bridge
    - Composite
    - Decorator
    - Facade
    - Flyweight
    - Proxy

Patrones de Diseño Orientado a Objetos [GoF]

- **Tipos de patrones de diseño:**
  - **De comportamiento**:
    - Chain of responsibility
    - Command
    - Interpreter
    - Iterator
    - Mediator
    - Memento
    - Observer
    - State
    - Strategy
    - Template method
    - Visitor

Idioms

- Idioms (Idiomas o modismos): Los modismos o expresiones idiomáticas son el nivel más bajo de abstracción en un sistema de patrones. La mayoría de los modismos son específicos de un lenguaje. Describen como implementar una parte particular de diseño al utilizar un lenguaje de programación determinado.

Idioms

- **Ejemplos:**
  - **Paquetes por layers:**
    - com.blah.action
    - com.blah.dao
    - com.blah.model
    - com.blah.util
  - **Paquetes por características:**
    - com.blah.painting
    - com.blah.seller
    - com.blah.auction
    - com.blah.webmaster
    - com.blah.useraccess
    - com.blah.util

Idioms

- **Ejemplos:**
  - **Implementando equals():**
    - El parámetro debe ser de tipo Object y no del tipo de la clase que sobrescribe el método equals. Se debe comparar el objeto con null y devolver false en caso de que el objeto sea igual a null. No se debe lanzar nunca un NullPointerException. Los campos de tipo nativo en Java (ej. int, float, etc) se deben comparar usando ==, si comparamos campos de tipo objeto debemos usar equals() y si comparamos un campo de tipo array de valores primitivos debemos usar Arrays.equals(). Cuando sobrescribimos equals() recordar sobrescribir hashCode() para ser consistentes con equals().

Idioms

- **Ejemplos:**
  - **Implementando equals():**

```
@Override
public boolean equals(Object obj) {
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Alumno other = (Alumno) obj;
    if (this.legajo != other.legajo) {
        return false;
    }
    if (!Objects.equals(this.nombre, other.nombre)) {
        return false;
    }
    if (!Objects.equals(this.apellido, other.apellido)) {
        return false;
    }
    return true;
}
```

Idioms

- **Ejemplos:**
  - **Rotando un String:**
    - La forma de rotar una cadena sin mucho lío

```
public String girarApellido() {
    return new StringBuilder(this.apellido).reverse().toString();
}
```

Idioms

- **Ejemplos:**
  - **Chequeo defensivo de Objetos:**
    - Nunca asumir que un objeto que es pasado como argumento no es null. Siempre debemos chequear esa condición explicitamente.

```
int encontrarIndice(List<String> list) {
    if (list == null) {
        throw new NullPointerException();
    }

    return list.indexOf("algo");
}
```

Idioms

- **Ejemplos:**
  - **Chequeo defensivo de índices de un arreglo:**
    - Nunca asumir que el índice pasado como parámetro, de un arreglo, se encuentre dentro de los limites de dicho arreglo. Siempre chequearlo explícitamente.

```
void frob(byte[] b, int index) {
    if (b == null) {
        throw new NullPointerException();
    }
    if (index < 0 || index >= b.length) {
        throw new IndexOutOfBoundsException();
    }

}
```

Idioms

- **Más Ejemplos:**
  - http://nayuki.eigenstate.org/page/good-java-idioms
  - http://c2.com/cgi/wiki?JavaIdioms