Orientación a Objetos II 2025

Explicación de práctica
Semana del 31 de marzo

Refactoring

Material Teórico

1 - Refactoring Ejemplo
2 - Catálogo de refactorings y bad smells
3 - Automatización del refactoring
4 - Otros recursos

Videos, información y más

Material de Teoria del 17 de marzo

Acá podrán encontrar diferentes recursos para estudiar el tema de refacto

El material se divide en 4 partes: la primera donde se ve a través de un ej
proceso de refactoring; la segunda donde se ve en detalle el catálogo de
sobre la automatización del refactoring, y la cuarta donde encontrarán lin

1. Refactoring. Ejemplo

- Transparencias (.PDF): proceso de refactoring sobre un ejemplo. Se
  Fowler.
- El proceso de refactoring se da en una secuencia, empezando por c
  descubrir otros refactorings que pueden aplicarse, por ejemplo, par
- A medida que vamos viendo cada refactoring revisaremos su mecá
  refactorings de Fowler.

-

2. Catálogo de refactorings y bad smells

- Transparencias utilizadas (.PDF): se describen muchos de los refacto
  es solo un acompañamiento del libro.

Refactoring

- Refactorización (sustantivo): cambio realizado en la estructura interna de un programa informático para que sea más fácil de entender y más “barato” de modificar sin cambiar su comportamiento observable.

- Refactorizar (verbo): reestructurar software aplicando una serie de refactorizaciones sin cambiar su comportamiento observable.

Refactoring v1 - Fowler

Refactoring

- Refactoring es una transformación que preserva el comportamiento, pero mejora el diseño

Bill Opdyke, PhD Thesis "Refactoring Object-Oriented Frameworks". Univ. of Illinois at Urbana-Champaign (UIUC). 1992. Director: Ralph Johnson.

Refactoring

- Refactorizar NO es tirar el código y escribirlo de nuevo
- Se debe preservar el comportamiento
- Refactorizar es seguir un método para mejorarlo:

1. Identificar code smells
2. Determinar cómo mejorarlo
3. Aplicar la mejora

Ejercicio 2

Para cada una de las siguientes situaciones, realice en forma iterativa los siguientes pasos:
(i) indique el mal olor,
(ii) indique el refactoring que lo corrige,
(iii) aplique el refactoring, mostrando el resultado final (código y/o diseño según corresponda).
Si vuelve a encontrar un mal olor, retorne al paso (i).

Ejercicio 2

2.1 Empleados

```
public class EmpleadoTemporario {
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    public double horasTrabajadas = 0;
    public int cantidadHijos = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico
                + (this.horasTrabajadas * 500)
                + (this.cantidadHijos * 1000)
                - (this.sueldoBasico * 0.13);
    }
}
```

```
public class EmpleadoPlanta {
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    public int cantidadHijos = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico
                + (this.cantidadHijos * 2000)
                - (this.sueldoBasico * 0.13);
    }
}
```

```
public class EmpleadoPasante {
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico - (this.sueldoBasico * 0.13);
    }
}
```

Ejercicio 2

2.1 Empleados

```java
public class EmpleadoTemporario {
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    public double horasTrabajadas = 0;
    public int cantidadHijos = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico
                + (this.horasTrabajadas * 500)
                + (this.cantidadHijos * 1000)
                - (this.sueldoBasico * 0.13);
    }
}
```

```java
public class EmpleadoPlanta {
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    public int cantidadHijos = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico
                + (this.cantidadHijos * 2000)
                - (this.sueldoBasico * 0.13);
    }
}
```

```java
public class EmpleadoPasante {
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico - (this.sueldoBasico * 0.13);
    }
}
```

Ejercicio 2

2.1 Empleados

```
public class EmpleadoTemporario {
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    public double horasTrabajadas = 0;
    public int cantidadHijos = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico
             + (this.horasTrabajadas * 500)
             + (this.cantidadHijos * 1000)
             - (this.sueldoBasico * 0.13);
    }
}
```

```
public class EmpleadoPlanta {
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    public int cantidadHijos = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico
             + (this.cantidadHijos * 2000)
             - (this.sueldoBasico * 0.13);
    }
}
```

```
public class EmpleadoPasante {
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico - (this.sueldoBasico * 0.13);
    }
}
```

Algunos bad smells

- Código duplicado
  - Extract Method
  - Pull Up Method
  - Form Template Method
- Métodos largos
  - Extract Method
  - Decompose Conditional
  - Replace Temp with Query
- Clases grandes
  - Extract Class
  - Extract Subclass
- Muchos parámetros
  - Replace Parameter with Method
  - Preserve Whole Object
  - Introduce Parameter Object
- Cambios divergentes (Divergent Change)
  - Extract Class
- “Shotgun surgery”
  - Move Method/Field
- Envidia de atributo (Feature Envy)
  - Move Method
- Data Class
  - Move Method
- Sentencias Switch
  - Replace Conditional with Polymorphism
- Generalidad especulativa
  - Collapse Hierarchy
  - Inline Class
  - Remove Parameter

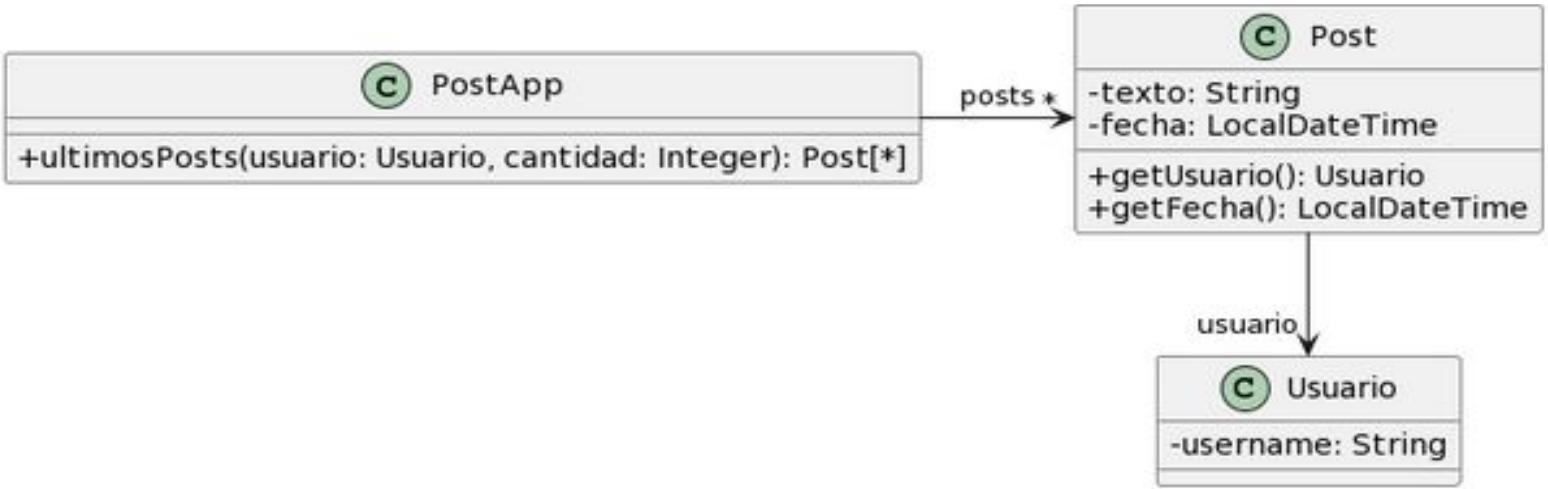
Algunos bad smells

- Cadena de mensajes

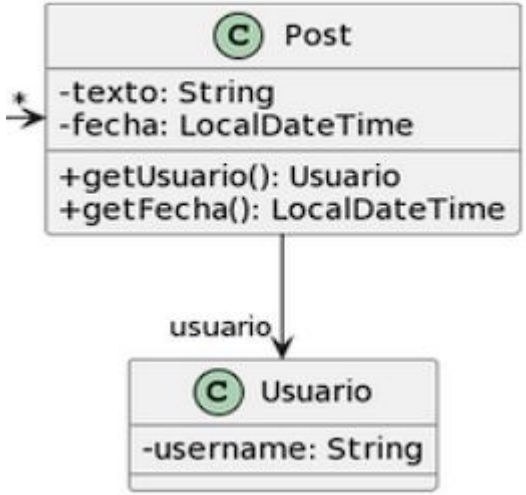
  (banco cuentaNro: unNro) movimientos first fecha

  - Hide Delegate
  - Extract Method & Move Method
- Middle man
  - Remove Middle man
- Inappropriate Intimacy
  - Move Method/Field
- Legado rechazado (Refused bequest)
  - Push Down Method/Field
- Comentarios
  - Extract Method
  - Rename Method

Ejercicio 2.3



Ejercicio 2.3




```java
/**
* Retorna los últimos N posts que no pertenecen al usuario user
*/
public List<Post> ultimosPosts(Usuario user, int cantidad) {

    List<Post> postsOtrosUsuarios = new ArrayList<Post>();
    for (Post post : this.posts) {
        if (!post.getUsuario().equals(user)) {
            postsOtrosUsuarios.add(post);
        }
    }

   // ordena los posts por fecha
   for (int i = 0; i < postsOtrosUsuarios.size(); i++) {
       int masNuevo = i;
       for(int j= i +1; j < postsOtrosUsuarios.size(); j++) {
           if (postsOtrosUsuarios.get(j).getFecha().isAfter(
     postsOtrosUsuarios.get(masNuevo).getFecha())) {
               masNuevo = j;
           }
       }
      Post unPost = postsOtrosUsuarios.set(i,postsOtrosUsuarios.get(masNuevo));
      postsOtrosUsuarios.set(masNuevo, unPost);
   }

    List<Post> ultimosPosts = new ArrayList<Post>();
    int index = 0;
    Iterator<Post> postIterator = postsOtrosUsuarios.iterator();
    while (postIterator.hasNext() &&  index < cantidad) {
        ultimosPosts.add(postIterator.next());
    }
    return ultimosPosts;
}
```