ESCOLA
POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA

AMPLIACIÓ DE BASES DE DADES I ENGINYERIA DEL PROGRAMARI

Pràctica 3: Primera activitat de patrons de disseny

Aaron Arenas Tomas - 78098697N
Ramon Escoda Semís - 49751728Z
Marc Cervera Rosell - 47980320C

18 de maig 2022

Grau en Enginyeria Informàtica

Índex

Índex de figures

1 Disseny

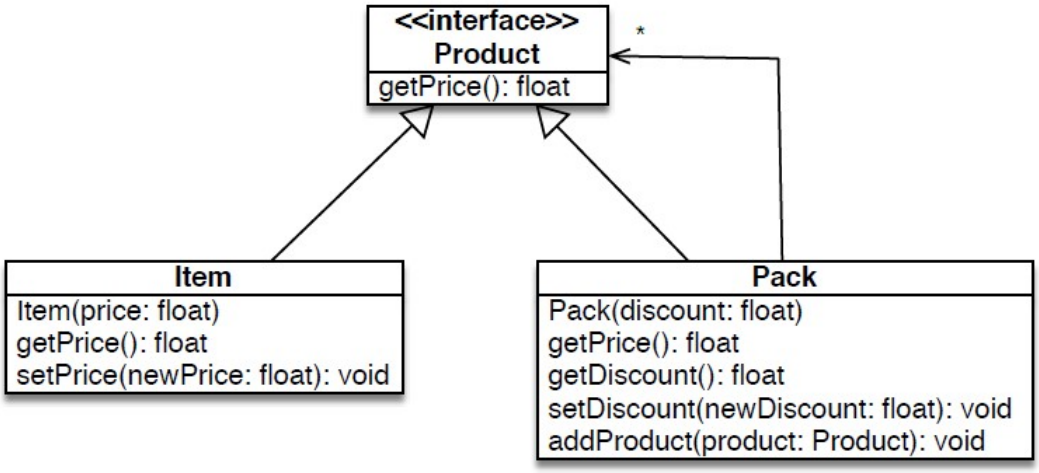

Figura 1: Disseny UML

Especificacions dels mètodes de les classes:

On la classe `Item`:

- representa un producte individual que, al seu constructor, rep el valor del preu el qual, per simplificar, serà un `float`.
- té el mètode `setPrice` per canviar el preu del producte
- té el mètode `getPrice` que retorna el preu del producte

Figura 2: Especificacions dels mètodes de la classe *Item*

I la classe `Pack`

- representa un grup de productes (que poden ser tant `Items` com `Packs`) que, al seu constructor rep el valor del tant per ú de descompte a aplicar
- té el mètode `setDiscount` per canviar el tant per ú de descompte
- té el mètode `addProduct` per afegir un nou producte al pack
- té el mètode `getDiscount` que retorna el tant per ú de descompte
- té el mètode `getPrice` que retorna el preu del pack: suma dels preus dels productes descomptada segons el tant per ú de descompte. Per exemple: si els productes tenen valors 10.0 i 20.0 i el tant per ú és 0.2, el preu és (1.0 – 0.2) * (10.0 + 20.0) = 24.0

Figura 3: Especificacions dels mètodes de la classe *Pack*

```
1 public final class PriceChanged {
2   private final float oldPrice;
3   private final float newPrice;
4   public PriceChanged(float oldPrice, float newPrice) {
5     this.oldPrice = oldPrice; this.newPrice = newPrice;
6   }
7   public float getOldPrice() { return oldPrice; }
8   public float getNewPrice() { return newPrice; }
9 }
```

Figura 4: Especificacions dels mètodes de la classe *PriceChanged*

2 Solució consensuada

Un cop comparades totes les resolucions individuals, les quals s'adjunten juntament amb aquest document, s'ha decidit, per unanimitat, entregar com a solució vàlida (segons el criteri dels membres del grup) i puntuable, la solució del membre /*NOM AUTOR*/. Com es podrà observar, en el codi de la solució final hi ha mètodes que no formen part de la solució al problema demanat, però s'han inclòs atès que s'ha usat l'IDE per programar la totalitat del codi i s'ha fet ús d'una classe anomenada *Main* que realitza diversos jocs de proves per comprovar la correctesa de la solució. Per poder veure la classe *Main*, hi ha dues opcions; veure l'annex d'aquest document (Secció *4 a la pàgina 5*) o seguir l'enllaç al directori de GitHub que hi ha més andavant (Secció *3 a la pàgina 4*).

2.1 Interfície *Product*

```
/*
   Marc Cervera Rossell
   Aaron Arenas Tomas
   Ramon Escoda Semis
*/

public interface Product {
    float getPrice();
}
```

2.2 Classe *Item*

```java
/*
   Marc Cervera Rossell
   Aaron Arenas Tomas
   Ramon Escoda Semis
*/

public class Item implements Product {
    private float price;

    public Item(float price) {
        this.price = price;
    }

    @Override
    public float getPrice() {
        return this.price;
    }

    public void setPrice(float price) {
        if (price < 0) {
            throw new IllegalArgumentException("Negative price");
        } else {
            this.price = price;
        }
    }

    @Override
    public String toString() {
        return "Item {" +
                "price = " + price +
                '}';
    }
}
```

2.3 Classe *Pack*

```java
/*
   Marc Cervera Rossell
   Aaron Arenas Tomas
```

```
   Ramon Escoda Semis
*/

public class Pack implements Product {

    private float discount;

    private List<Product> products;

    public Pack(float discount) {
        this.discount = discount;
        this.products = new ArrayList<Product>();
    }

    public void removeProd(Product p) {
        this.products.remove(p);
    }

    @Override
    public float getPrice() {
        float price = 0f;
        for (Product p : products) {
            price += p.getPrice();
        }
        price = (1f - getDiscount()) * price;
        return price;
    }

    public float getDiscount() {
        return discount;
    }

    public void setDiscount(float discount) {
        if (discount < 0f || discount > 1f) {
            throw new IllegalArgumentException("Not valid discount");
        } else {
            this.discount = discount;
        }
    }

    public void addProduct(Product product) {
```

```
        this.products.add(product);
    }

    @Override
    public String toString() {
        return "Pack {" +
                "discount = " + discount +
                ", products = " + products +
                '}';
    }
```

2.4 Classe *PriceChanged*

```
/*
   Marc Cervera Rossell
   Aaron Arenas Tomas
   Ramon Escoda Semis
*/

public final class PriceChanged {
    private final float oldPrice;
    private final float newPrice;

    public PriceChanged(float oldPrice, float newPrice) {
        this.oldPrice = oldPrice;
        this.newPrice = newPrice;
    }

    public float getOldPrice() {
        return oldPrice;
    }

    public float getNewPrice() {
        return newPrice;
    }
```

3 Enllaç al directori GitHub

En aquesta secció és troba l'enllaç al directori de GitHub amb el codi de la solució final.

Com es podrà observar, el projecte d'IntellIJ disposa d'una classe *Main*, on hi ha diferets exemples d'execució on es comprova la correctesa dels diferents patrons aplicats.
Enllaç al directori GitHub

4 Annex

4.1 Classe *Main*

Com s'ha esmentat en anterioritat en aquest document, aquesta classe ha servit a l'equip de desenvolupament per ratificar la correctesa de la solució que s'ha donat com a vàlida per ésser puntuable.

4.1.1 Codi de la classe *Main*

```
public class Main {
/*
   Marc Cervera Rossell
   Aaron Arenas Tomas
   Ramon Escoda Semis
*/

    public static void main(String[] args) {
        /* Pack */
        Pack p1 = new Pack(0.5f);

        /* Items */
        Item i1 = new Item(12.7f);
        Item i2 = new Item(10.13f);
        Item i3 = new Item(400.99f);

        /* Including items inside the pack */
        p1.addProduct(i1);
        p1.addProduct(i2);
        p1.addProduct(i3);

        System.out.println("---------- Composite pattern test part
            ----------");
        System.out.println("\n");
        System.out.println("---------- Pack with items ----------");
        System.out.println(p1);
        System.out.println("Final price after discount: " + (p1.getPrice()));
```

```
        /* Adding a pack inside a pack */
        Pack p2 = new Pack(0.1f);
        Item i4 = new Item(1.7f);
        Item i5 = new Item(123.45f);
        p2.addProduct(i4);
        p2.addProduct(i5);
        p1.addProduct(p2);

        System.out.println("\n");

        System.out.println("---------- Adding a pack inside a pack
            ----------");
        System.out.println(p1);
        System.out.println("Final price after discount: " + (p1.getPrice()));
    }
```