Lab VI.

Objetivos

Os objetivos deste trabalho são:

- Identificar e utilizar padrões relacionados com a construção e estrutura de objetos e classes
- Aplicar boas práticas de programação por padrões em casos práticos

Nota: Para além do código no github, apresente o diagrama de classes da solução final.

VI.1 Pastelaria

Pretende-se criar um conjunto de classes que modele a elaboração de bolos numa pastelaria. Para tal, considere que um bolo é representado pela classe *Cake*. Por omissão, os bolos são circulares, mas podem ser quadrados ou retangulares (não é necessário definir a dimensão), e podem ter um diferente número de camadas, com uma camada intermédia de creme.

```
class Cake {
    private Shape shape;
    private String cakeLayer;
    private int numCakeLayers;
    private Cream midLayerCream;
    private Cream topLayerCream;
    private Topping topping;
    private String message;

       //.. restantes métodos
}
```

Considere ainda que todos os bolos são construídos seguindo um padrão *Builder* que usa a interface *CakeBuilder*.

```
interface CakeBuilder {
    public void setCakeShape(Shape shape);
    public void addCakeLayer();
    public void addCreamLayer();
    public void addTopLayer();
    public void addTopping();
    public void addMessage(String m);
    public void createCake();
    public Cake getCake();
}
```

Modele as classes e construa o código necessário para que o cliente possa executar pedidos como os apresentados no método *main* seguinte. Note que o código necessário para construir cada bolo é sempre o mesmo, apenas variando o *CakeBuilder* passado em *CakeMaster*.

```
public static void main(String[] args) {
    CakeMaster cakeMaster = new CakeMaster();

    CakeBuilder chocolate = new ChocolateCakeBuilder();
    cakeMaster.setCakeBuilder(chocolate);
    cakeMaster.createCake("Congratulations");              // 1 cake layer
    Cake cake = cakeMaster.getCake();
    System.out.println("Your cake is ready: " + cake);

    CakeBuilder sponge = new SpongeCakeBuilder();
    cakeMaster.setCakeBuilder(sponge);
    cakeMaster.createCake(Shape.Square, 2, "Well done");   // squared, 2 layers
    cake = cakeMaster.getCake();
    System.out.println("Your cake is ready: " + cake);

    CakeBuilder yogurt = new YogurtCakeBuilder();
    cakeMaster.setCakeBuilder(yogurt);
    cakeMaster.createCake(3, "The best");                  // 3 cake layers
    cake = cakeMaster.getCake();
    System.out.println("Your cake is ready: " + cake);

    // you should add here other example(s) of CakeBuilder

}
```

Output:

```
Your cake is ready: Soft chocolate cake with 1 layers, topped with Whipped_Cream cream and
Fruit. Message says: "Congratulations".

Your cake is ready: Sponge cake with 2 layers and Red_Berries cream, topped with Whipped_Cream
cream and Fruit. Message says: "Well done".

Your cake is ready: Yogurt cake with 3 layers and Vanilla cream, topped with Red_Berries cream
and Chocolate. Message says: "The best".
```

VI.2 Construtor com demasiados parâmetros

Considere a classe seguinte. Reescreva-a usando o padrão *builder*.

```
public class Movie {
   private final String title;
   private final int year;
   private final Person director;
   private final Person writer;
   private final String series;
   private final List<Person> cast;
   private final List<Place> locations;
   private final List<String> languages;
   private final List<String> genres;
   private final boolean isTelevision;
   private final boolean isNetflix;
   private final boolean isIndependent;

   public Movie(
      final String movieTitle,
      final int movieYear,
      final Person movieDirector,
      final Person movieWriter,
      final String movieSeries,
      final List<Person> movieCast,
      final List<Place> movieLocations,
      final List<String> movieLanguages,
      final List<String> movieGenres,
      final boolean television,
      final boolean netflix,
      final boolean independent)    {
             this.title = movieTitle;
             this.year = movieYear;
             this.director = movieDirector;
             this.writer = movieWriter;
             this.series = movieSeries;
             this.cast = movieCast;
             this.locations = movieLocations;
             this.languages = movieLanguages;
             this.genres = movieGenres;
             this.isTelevision = television;
             this.isNetflix = netflix;
             this.isIndependent = independent;
   }
}
```