PROJETOS PRÁTICOS

Design Patterns com Spring

Carlos Biagolini-Jr.

São Paulo / 2022

Projeto

Descrição do projeto: implementar 3 padrões gerais de códigos/projetos usando Linguagem Java. Os padrões escolhidos foram: Singleton (padrão criacional), Strategy (padrão comportamental) e Facade (padrão estrutural).

Links

Projeto

https://web.dio.me/lab/explorando-padroes-de-projetos-na-pratica-com-java/learning/dbad4e6b-fc8e-4215-b305-435b0ad652c1

Referências externas

Refactoring Guru (2022) Design Patterns. Disponível em https://refactoring.guru/pt-br/design-patterns/s.

Ambiente de desenvolvimento

Linguagem

java version "1.8.0_321"

Java(TM) SE Runtime Environment (build 1.8.0_321-b07)

Java HotSpot(TM) 64-Bit Server VM (build 25.321-b07, mixed mode)

IDE

IntelliJ IDEA 2021.3.1 (Community Edition).

Build #IC-213.6461.79, built on December 28, 2021

Runtime version: 11.0.13+7-b1751.21 amd64

VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.

Plugins

Lombok

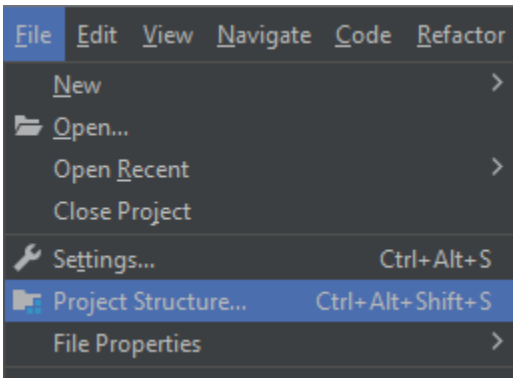
Passo 1: Download do Lombok.

Faça download do arquivo .jar do Lombok, disponível no link https://projectlombok.org/download. Salve num local conhecido.

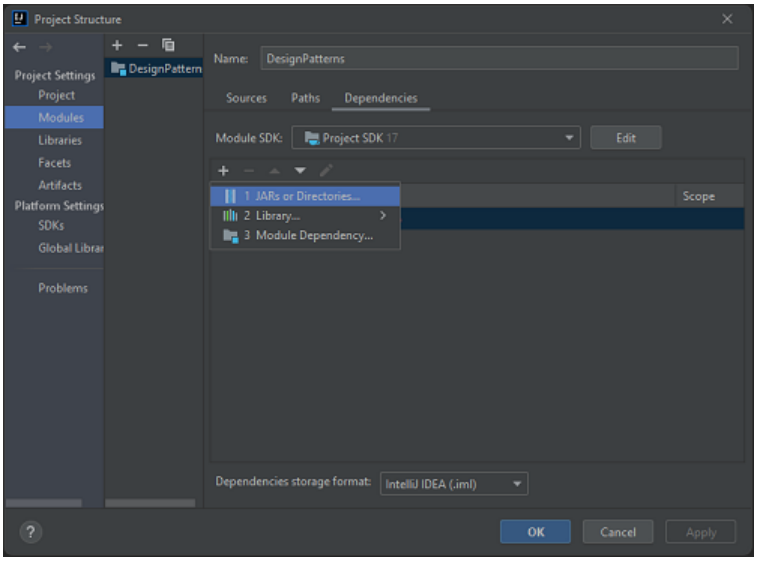
Passo 2: OPCIONAL: como se você apagar o arquivo .jar optido no passo acima o Lombok vai parar de funcionar. Eu optei por colocar esse arquivo dentro de uma pasta do próprio Java. Portanto, nesse passo eu recortei o arquivo da pasta conhecida (passo anterior) e colei em "C:\Program Files\Java\lombok"

Passo 3: Instalar o .jar do Lombk. Siga o passo-a-passo descrito:

File → Project Structure ...

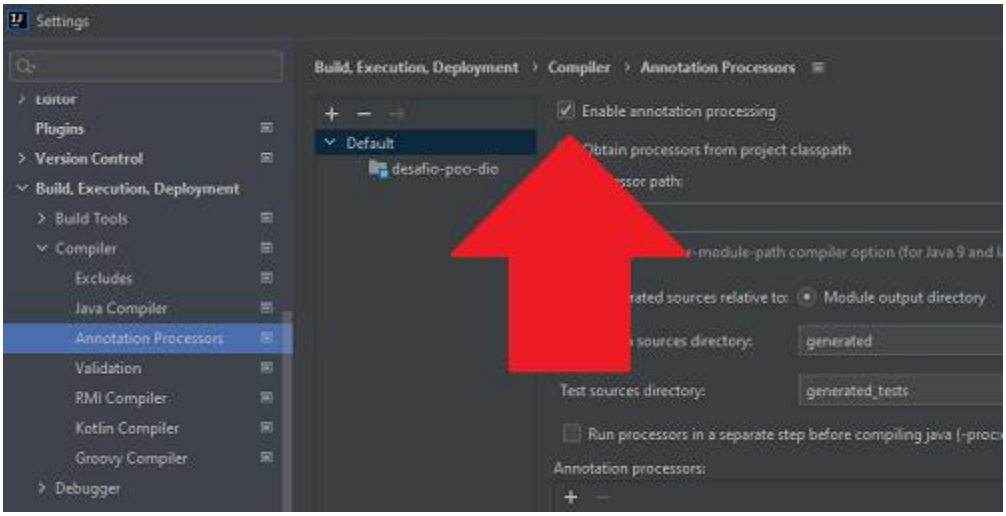


Modules → + JARs or Directories.. → Selecione o Arquivo Jar (se você seguiu o passo 2, estará em C:\Program Files\Java\lombok) → Ok → Ok.

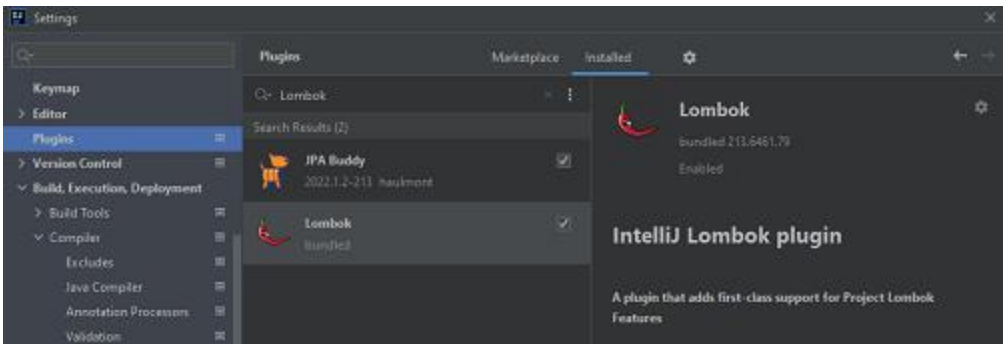


Passo 4: Nesse ponto o Lombok já deve estar funcionado. Caso não esteja funcionando, certifique-se que: i) o IntelliJ está habilitado para o uso das anotações; ii) o plugin do Lombok está ativado;

i) IntelliJ está habilitado para o uso das anotações: https://stackoverflow.com/a/27430992/4678899



ii) Ativar plugin do Lombok: https://projectlombok.org/setup/intellij



Base do projeto

Em caso de dúvidas, veja: https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-application.html#get-started

Passo 1: Criar projeto.

File → New → Project ... → Java 17 → Next → Next → Project name = DesignPatterns; Project location = ~\AmbienteDesenvolvimento\Dio\ → Finish

Passo 2: Criar Main.

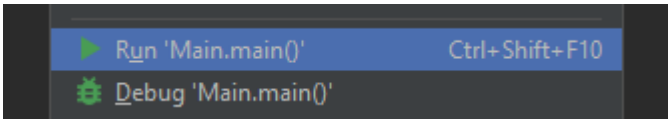
Clicar com botão direito na pasta scrc → New → Java Class. → Name: Main; Tipo: Class

Passo 3: Desenvolver Main.

src/Main.java

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("MAIN WORKS");
    }
}
```

Passo 4: Rodar projeto: Clicar com botão direito na região do código da função main → Run 'Main.main()'.



Controle de versionamento

Passo 1: Acesse o link https://github.com/new e crie o projeto: desafioDioDesignPatterns

Passo 2: Quando você tiver seus primeiros arquivos, já pode iniciar o repositório

```
git init
git config --local user.name "Carlos Biagolini-Jr"
git config --local user.email "c.biagolini@gmail.com"
git remote add origin https://github.com/biagolini/desafioDioDesignPatterns.git
git add .
git commit -m "Base"
git branch -M main
git push -u origin main
```

Menu de opções

Passo 1: Crie a classe “src/br/com/operador/Menu.java”

Passo 2: Desenvolva o menu de opções. O layout da do menu foi inspirado em: https://stackoverflow.com/a/65813230/4678899

src/br/com/operador/Menu.java

```
package br.com.tutorial.operador;

import java.util.Scanner;

public class Menu {
    Scanner scanner = new Scanner(System.in);

    private void singleton() {
        System.out.printf("Exemplo de uso do padrão Singleton...\n");
    }

    private void strategy() {
        System.out.printf("Exemplo de uso do padrão Strategy...\n");
    }

    private void facade() {
        System.out.printf("Exemplo de uso do padrão Facade...\n");
    }

    public void menu(){
        Integer opcao = 0;
        do{
            System.out.println(formatRow("!---------!-------------------------------------
-#"));
            System.out.println(formatRow("|  Opção  | Descrição
|"));
            System.out.println(formatRow("$---------*-------------------------------------
-("));
            System.out.println(formatRow("|    1    | Teste o uso do padrão Singleton
|"));
            System.out.println(formatRow("|    2    | Teste o uso do padrão Strategy
|"));
            System.out.println(formatRow("|    3    | Teste o uso do padrão Facade
|"));
            System.out.println(formatRow("|    0    | Sair
|"));
            System.out.println(formatRow(")---------+-------------------------------------
-^"));
            System.out.println("\033[3mDigite o número correspondente a opção dese-
jada:\033[0m");

            opcao = scanner.nextInt();
            scanner.nextLine();

            switch (opcao) {
                case 1: singleton(); break;
                case 2: strategy(); break;
                case 3: facade(); break;
                case 0:
                    System.out.println(formatRow("-------------------------------------
```

```
----------"));
                    System.out.println(formatRow("\033[3m Obrigado por utilizar nosso
software, até logo \033[0m"));
                    break;
                default:System.out.printf("Opção inválida;\n");
            }
            System.out.printf("\033[0;1m" +"\n\n\t\t\t\t\t\t\033[3mPressione ENTER para
prosseguir\033[0m");
            scanner.nextLine();

        } while (opcao!=0);
    }

    public static String formatRow(String str) {
        return str
                .replace('|', '\u2502')
                .replace('!', '\u250c')
                .replace('@', '\u252c')
                .replace('#', '\u2510')
                .replace('$', '\u251c')
                .replace('*', '\u253c')
                .replace('(', '\u2524')
                .replace(')', '\u2514')
                .replace('+', '\u2534')
                .replace('^', '\u2518')
                .replace('-', '\u2500');
    }

}
```

Passo 3: Aplicar o menu de opções na função Main.

src/Main.java

```
import br.com.tutorial.operador.Menu;

public class Main {
    public static void main(String[] args) {
        Menu menu = new Menu();
        menu.menu();
    }
}
```

Passo 4: Postar parcial no git

```
git add .
git commit -m "Parte 01 Menu de navegação"
git push origin main
```

Encapsular testes

Passo 1: Criar classes para cada um dos padrões, e no menu de opções chamar as funções de teste de cada padrão.

src/br/com/tutorial/padroes/singleton/Singleton.java

```java
package br.com.tutorial.padroes.singleton;

public class Singleton {

    public void testeSingleton(){
        System.out.printf("Exemplo de uso do padrão Singleton...\n");
    }
}
```

src/br/com/tutorial/padroes/strategy/Strategy.java

```java
package br.com.tutorial.padroes.strategy;

public class Strategy {

    public void testeStrategy(){
        System.out.printf("Exemplo de uso do padrão Strategy...\n");
    }
}
```

src/br/com/tutorial/padroes/facade/Facade.java

```java
package br.com.tutorial.padroes.facade;

public class Facade {

    public void testeFacade(){
        System.out.printf("Exemplo de uso do padrão Facade...\n");
    }
}
```

Passo 2: Postar parcial no git

```
git add .
git commit -m "Parte 02 Estabelecer diferentes pacotes para cada padrao"
git push origin main
```

Adaptação do projeto original

Referências recomendadas:

Singleton: https://refactoring.guru/pt-br/design-patterns/singleton

Strategy: https://refactoring.guru/pt-br/design-patterns/strategy

Facade: https://refactoring.guru/pt-br/design-patterns/facade

Passo 1: Desenvolver os métodos seguindo os mesmos procedimentos do professor, adaptando o mesmo para a nossa estrutura de projeto. Código fornecido pelo professor Venilton Falvo-Jr disponibilizado em: https://github.com/digitalinnovationone/lab-padroes-projeto-java

Passo 2: Postar parcial no git

```
git add .
git commit -m "Parte 03 Adaptação do projeto original"
git push origin main
```