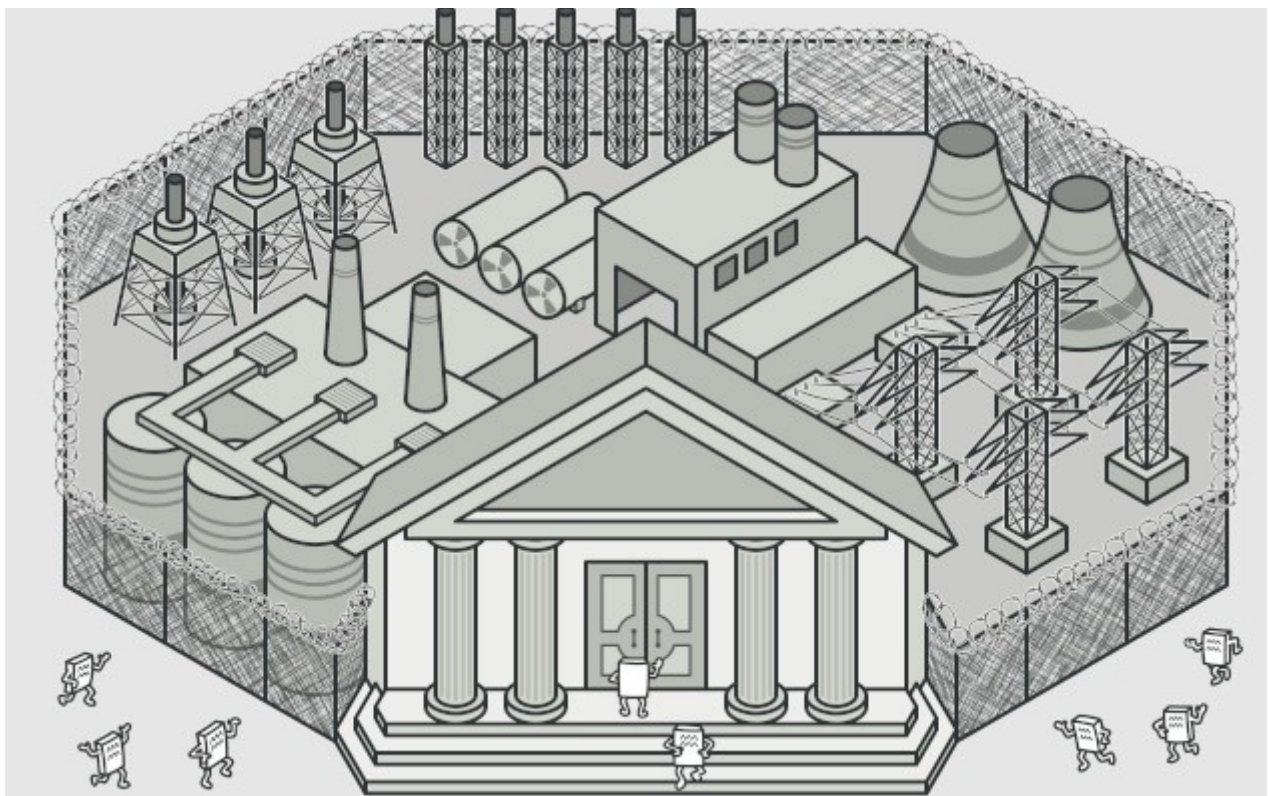


Aula 10 – Padrões de Projeto (Design Patterns)

Padrão de Projeto: Representa uma abordagem bem-sucedida, em nível de código, acerca da resolução de um problema/cenário específico. Vale ressaltar que não é necessário conhecer todos os padrões, mas sim compreendê-los e saber de sua existência. Afinal, não é necessário “reinventar a roda”, mas sim saber onde encontrá-la.

Padrão Facade: É um padrão utilizado para simplificar, via uma interface (fachada), a interação com a parte mais complexa do sistema. Ou seja, é fornecido ao usuário uma “fachada” de uso, onde ele não precisa saber como a parte mais complexa do sistema funciona. Isso me lembra o conceito de “encapsulamento”, onde se deixa apenas funções básicas visíveis, reduzindo a visibilidade da parte mais complexa. Esse padrão traz os benefícios da simplificação do uso do sistema, baixo acoplamento e manutenibilidade.



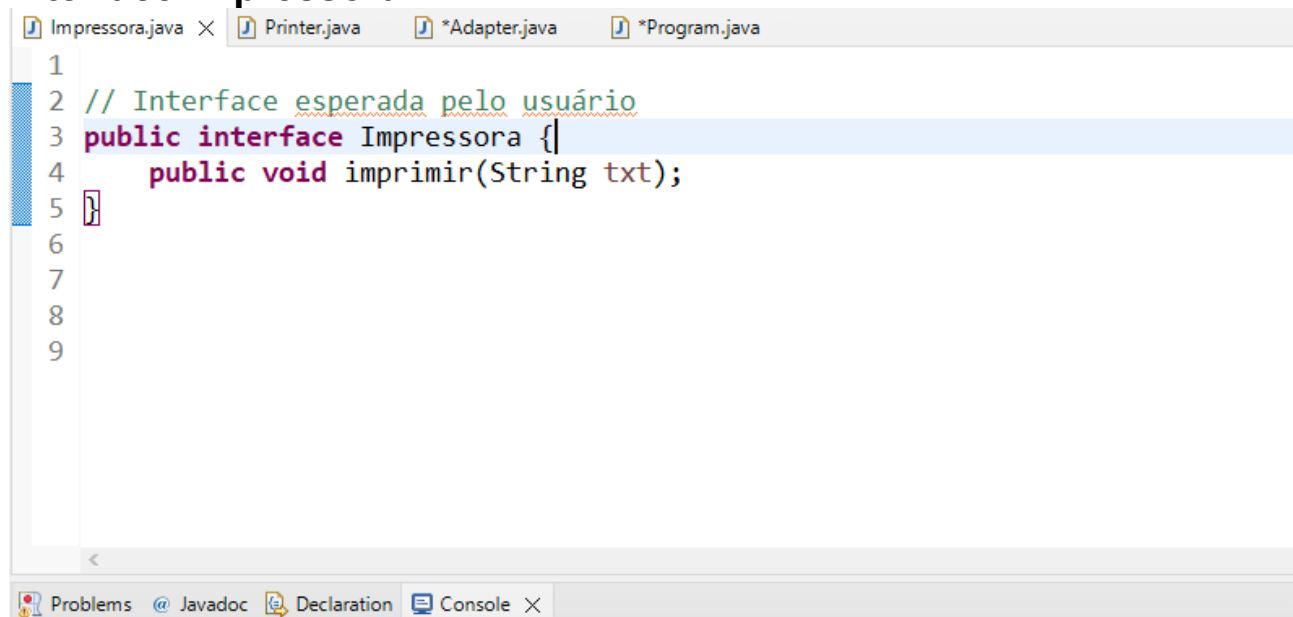
Exemplo: Temos um sistema bancário com uma série de classes e métodos: temos a classe “conta”, que possui os métodos “depositar” e “sacar”; a classe “autenticação”, que possui o método

“autenticar_cliente”; a classe “transação”, que possui o método “transferir”; e, por fim, temos a classe “banco”, que será a nossa fachada. Neste caso, o cliente interagirá exclusivamente com a classe fachada que fornecerá todas as operações necessárias de um sistema bancário.

Padrão Adapter: Esse padrão permite que objetos com interfaces diferentes possam cooperar entre si, convertendo a interface de uma classe em outra interface esperada pelo cliente. Ele atua como uma camada intermediária que traduz as chamadas entre as interfaces, facilitando a comunicação entre objetos com interfaces incompatíveis.

Por exemplo: Neste exemplo, temos uma classe existente chamada “Printer” com um método “print()”, porém, precisamos interagir com ela por meio de uma interface chamada “Impressora” que possui um método “imprimir()”. Para resolver essa incompatibilidade, utilizamos o padrão Adapter. Criamos a classe “Adapter”, que implementa a interface “Impressora” e encapsula uma instância da classe Printer. O método “imprimir()” do “Adapter” traduz a chamada para o método “print()” da classe “Printer”. Assim, podemos interagir com a classe “Printer” por meio do adaptador “AdapterPrinter”, utilizando a interface “Impressora”.

Interface impressora



```
1
2 // Interface esperada pelo usuário
3 public interface Impressora {
4     public void imprimir(String txt);
5 }
6
7
8
9
```

Classe Printer

```
Impressora.java  Printer.java  *Adapter.java  *Program.java

1
2 //classe incompatível ou legada
3 public class Printer{
4     //método de impressão
5     public void print(String txt) {
6         System.out.println("Impressão : "+txt);
7     }
8 }
9
```

Problems Javadoc Declaration Console

Classe Adapter

```
Impressora.java  Printer.java  *Adapter.java  *Program.java

1 //classe adapter atua como uma camada intermediária que traduz as chamadas de um
2 //formato para outro, facilitando a interação entre os objetos.
3 public class Adapter implements Impressora {
4     private Printer printer; // referência p/ o obj printer
5
6     public Adapter(Printer txt) {
7         this.printer = txt; // recebe um objeto do tipo printer
8     }
9     // implementação do método esperado pela interface
10    public void imprimir(String txt) {
11        printer.print(txt);
12        // aqui estamos acessando o objeto legado e passando um valor a ser impresso.
13    }
14
```

Problems Javadoc Declaration Console

Classe principal



```
1
2 public class Program {
3     public static void main(String[] args) {
4         Printer printer = new Printer();// instanciando a classe legada printer
5
6         Impressora adapter = new Adapter(printer);
7         // A classe adapter, que é do tipo "impressora", irá receber uma instância do
8         // objeto legado (printer). Para permitir o acesso ao método dessa classe
9
10        adapter.imprimir("Padrão adapter");
11        //aqui estamos usando o método definido pela interface. E, usando-o como ponto, acessando
12        //o método print da classe printer
13
14    }
```

Problems @ Javadoc Declaration Console X

<terminated> Program (45) [Java Application] C:\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (17 de mai. de 2023 00:03:02 - 00:03:02) [pid: 1117]

Impressão : Padrão adapter

Padrões de projeto Java EE: São uma série de padrões que se concentram em soluções comuns para problemas específicos encontrados no desenvolvimento de aplicativos empresariais usando a plataforma Java Enterprise Edition (Java EE). Esses padrões ajudam a promover boas práticas de design, modularidade, reutilização de código e escalabilidade em aplicativos corporativos.

DAO (Data Access Object): Um padrão que encapsula a lógica de acesso a dados, fornecendo uma interface consistente para interagir com diferentes fontes de dados, como bancos de dados, serviços web ou sistemas de arquivos.

Padrão Transfer Object (TO): É usado para transferir dados entre diferentes camadas ou componentes de um sistema. Ele encapsula um conjunto de atributos relacionados em um objeto simples, proporcionando uma estrutura eficiente para transportar dados de um lugar para outro.

Padrão Compose: Também conhecido como Composite, é um padrão de projeto estrutural que permite tratar objetos individuais e composições de objetos de maneira uniforme. Ele é utilizado para construir estruturas hierárquicas de objetos, onde um objeto pode conter outros objetos, formando uma árvore de composição.

