PADRÕES DE PROJETO JAVA

Melhorando o desempenho da aplicação

CENTEC
INSTITUTO CENTRO DE ENSINO TECNOLÓGICO

GOVERNO DO
ESTADO DO CEARÁ
*Secretaria da Ciência, Tecnologia
e Educação Superior*

Padrões de projeto

- Padrões de projeto servem para tornar um código reutilizável, padronizado e visam diminuir problemas que possam ocorrer durante a criação do código.

- A primeira vez que essa proposta surgiu foi em 1995 no livro “Design Paterns: Elements of Reusable Object-Oriented Software”.

Padrões de projeto

Esse assunto por si só é bem extenso, então só cobriremos alguns padrões, alguns que usaremos no nosso projeto como Factory Method e MVC, mas de início já aconselho que dê continuidade e conheça os outros, pois é uma habilidade muito útil para o mercado de trabalho

Padrões GoF

Esse é o padrão específico para tratar softwares baseados em orientação a objeto. São divididos em três grupos:

- **Padrões de criação;**
- **Padrões estruturais;**
- **Padrões comportamentais.**

Padrões GoF.

Vimos que os objetos são instâncias da classe, criadas para que possam executar suas funcionalidades. Por ser criado através de um construtor, podemos encontrar algumas limitações em certas situações.

- O código responsável por criar os objetos tem que conhecer seus construtores.

- O construtor não delimita seu número de instâncias.

- O objeto só poderá ser criado se seguir integralmente o seu construtor.

Universidade do Trabalho Digital

Conexão com JDBC

Factory method: Esse método retira das classes de negócio a obrigação de criar objetos, passando essa responsabilidade para suas subclasses ou implementando através de interfaces, abstract factory.

Abstract Factory: As subclasses de uma Factory abstrata são conhecidas como concrete factories. No módulo passado usamos uma factory como exemplo para a criação das conexões com o banco de dados, logo exploraremos um pouco mais esse exemplo.

Universidade do Trabalho Digital

ConnectionFactory.

```java
import java.sql.Connection;
import java.sql.DriverManager;

public class ConnectionFactory {

    private static final String USERNAME = "root";

    private static final String PASSWORD = "";

    private static final String DATABASE_URL =
    "jdbc:mysql://localhost/db_supermercado?useTimezone=true&serverTimezone=UTC";


}
```

Vamos criar a ConnectionFactory. Que vai ser a classe responsável por criar a conexão do banco de dados.Exemplo acima mostra a criação das variáveis de conexão.

ConnectionFactory.

```java
public class ConnectionFactory {

private static final String USERNAME = "root";

    private static final String PASSWORD = "";

    private static final String DATABASE_URL =
    "jdbc:mysql://localhost/db_supermercado?useTimezone=true&serverTimezone=UTC";


    public static Connection createConnectionToMySQL() throws Exception{
        Class.forName("com.mysql.cj.jdbc.Driver");

        Connection connection = DriverManager.getConnection(DATABASE_URL,USERNAME,PASSWORD);

        return connection;
    }
```

Aqui temos uma factory que possuí um método para criar a conexão com o banco de dados quando for chamado

ConnectionFactory.

```
public static void main(String[]args) throws Exception {

    Connection con = createConnectionToMySQL();

    if(con!=null) {
        System.out.println("Conexão obtida com sucesso");
        con.close();
    }

}
```

Esta função serve para testar se a conexão esta funcionando.

Padrões GoF de criação

Singleton: Esse padrão é utilizado quando se deseja restringir o número de instâncias de uma classe a um. Na hora da utilização do singleton devemos atentar a alguns aspectos como:

- O acesso ao construtor da classe é feito através de um método que verifica se já existe uma instancia e só a criará caso não exista;
- O construtor da classe é privado o que impede que seja acessado de fora da mesma;
- Isso torna a classe do tipo final já que dessa maneira não podemos criar subclasses.

Universidade do Trabalho Digital

Padrões GoF de criação

- A própria classe cria uma instância de si, caso não aja nenhuma quando ela for chamada. Note que o modificador de acesso do construtor é private.

```
Public class Singleton(){

    private static Singleton unicaInstancia;

    private Singleton(){

    };

    public static synchronized Singleton getInstance(){
        if(unicaInstancia == null){
            unicaInstancia = new Singleton;
        }

        return unicaInstancia;
    }
}
```

Builder:

Esse padrão é responsável por guardar a lógica de toda a criação de um objeto dentro de uma classe.

Isso faz com que toda a criação do objeto seja abstraída já que está por responsabilidade do builder, tornando assim o código mais enxuto e menos suscetível a erro. Ele é composto por:

- Classe builder
- Concrete builder
- Director
- Product

Universidade do Trabalho Digital