Roteiro 01

Os roteiros a serem desenvolvidos visão trazer a percepção da evolução no processo de desenvolvimento de software. Por isso, iremos criar um projeto com vários pacotes, onde cada pacote represente a evolução da implementação deste projeto

Neste roteiro iremos analisar alguns princípios (boas práticas) de Padrões de Projeto

1 – Programe para uma interface e não para uma implementação

2 – Favoreça o uso de Composição sobre Herança

Início do projeto – Pacote : roteiro1.parte1

1 – Criar um projeto no NetBeans chamado **padroesroteiros**

2 – Dentro do projeto criar um pacote chamado **roteiro1.parte1**

3 – Neste roteiro vamos criar a simulação de um sistema capaz de conectar com diferentes bancos de dados. Exemplo : MySql, Oracle ou SQL Server. E o cenário bem simples a ser criado será um sistema de reserva de quarto para hotel.

4 – Inicialmente a primeira classe a ser criada será a de conexão com o Banco MySql. Esta classe deve ser criada no pacote **roteiro1.parte1** e deve se chamar **MysqlConnection**

5 – Dentro da Classe **MysqlConnection** teremos apenas um método chamado **connect()**

```
public void connect() {
      System.out.println("Conectando com o MySQL");
   }
```

6 – Para representar as classes de negócio do nosso sistema criaremos apenas 3 classes : ServicoQuarto, ServicoReserva, RelatorioReserva.

A primeira classe de negócio a ser criada será **ServicoQuarto**. Ela deverá ter um atributo privado chamado **connection** do tipo Mysqlconnection e deverá ser instanciada dentro do construtor padrão da classe e representa a variável de conexão com o banco de dados. Nesta classe deverá ser criado também um método chamado **verificarQuarto** onde será feita a conexão com o banco e será exibida uma mensagem para representar a lógica de negócio para esta classe.

```
package roteiro1.parte1;

public class ServicoQuarto {
   private MysqlConnection connection;

   public ServicoQuarto() {
     this.connection = new MysqlConnection();
   }

   public void verificarQuarto(){
      this.connection.connect();
      System.out.println("Lógica de negócio para o Serviço de Quarto");
   }
}
```

A segunda classe de negócio a ser criada será **ServicoReserva**. De forma semelhante a primeira classe teremos um atributo **connection** para conexão com o banco que será instanciado no construtor padrão da classe. Devemos criar também um método chamado **criarReserva** onde será feita a conexão com o banco e será exibida uma mensagem para representar a lógica de negócio para criação de uma reserva de quarto.

```
public void criarReserva(){
      this.connection.connect();
      System.out.println("Lógica de negócio para Reserva do Quarto");
   }
```

A terceira classe de negócio a ser criada será **relatorioReserva**. Seguindo o mesmo padrão, criaremos um atributo privado para conexão com o banco que será instanciado dentro do construtor da classe. Devemos criar também um método chamado **gerarReserva** onde será feita a conexão com o banco e será exibida uma mensagem para representar a lógica de negócio para geração de um relatório.

```
public void gerarRelatorio(){
      this.connection.connect();
      System.out.println("Lógica de negócio para geração do relatório");
   }
```

7 – Criaremos agora uma classe com o método **main()** para teste da conexão com o banco e dos serviços do nosso sistema chamada **TesteConexao.** Verique se o resultado saiu como esperado.

```
package roteiro1.parte1;

public class TesteConexao {

   public static void main(String[] args) {
      ServicoQuarto quarto = new ServicoQuarto();
      quarto.verificarQuarto();

      ServicoReserva reserva = new ServicoReserva();
      reserva.criarReserva();

      RelatorioReserva relatorio = new RelatorioReserva();
      relatorio.gerarRelatorio();
   }
}
```

8 – Utilize uma ferramenta de software qualquer para geração do diagrama de classes para esta etapa do projeto (Sugestão : Astah Community). Obs.: Adicione aqui o diagrama para que seja disponibilizado no teams

Pacote : roteiro1.parte2

Imagine agora que o nosso sistema precise evoluir e precisaremos conectar com outro banco de dados, o banco de dados Oracle.

1 – No mesmo projeto crie o pacote roteiro1.parte2

2 – Copie todas as classes criadas na parte1 para o novo pacote

3 – Crie agora a classe de conexão com o Oracle semelhante a classe **MysqlConnection.** Esta nova classe deve se chamar **OracleConnection.**

4 – Sem nenhuma intervenção na modelagem feita da parte 1, faça uma refatoração das 3 classes de negócio (ServicoQuarto, ServicoReserva, RelatorioReserva) para que agora realizem a conexão com o Oracle.

5 – Realize os testes com a Classe TesteConexão.

6 – Avalie os impactos desta mudança no projeto caso estivéssemos em um sistema maior. Esta foi a melhor solução ?

Pacote : roteiro1.parte3

Prosseguindo na evolução do projeto iremos aplicar agora o princípio de se programar para uma interface.

1 - No mesmo projeto crie o pacote roteiro1.parte3

2 – Copie todas as classes criadas na parte2 para o novo pacote

3 – As classes MysqlConnection e OracleConnection possuem um método em comum chamado **connect**. Este método possui implementações de código específicas de cada banco de dados. A ideia de se programar para uma interface (Criando uma interface ou Classe Abstrata) é que se possa generalizar este código no método connect de forma que diversas outras classes possam utilizar com relativa facilidade. Para isso, criaremos uma interface. Lembrando que todos os métodos definidos dentro de uma interface java não possuem implementação. Chamaremos esta interface de **Connection**.

```
package roteiro1.parte3;

public interface Connection {

   public void connect();

}
```

4 - As classes MysqlConnection e OracleConnection deverão agora ser refatoradas para implementar a interface Connection. Lembrando que quando uma classe implementa uma Interface, todos os métodos definidos na interface deverão obrigatoriamente ser implementados nesta classe que a implementa. Neste caso, o método connect presente na interface deve ser obrigatoriamente implementado nas classes MysqlConnection e OracleConnection, que por sua vez já foi implementado nas etapas anteriores.

```
package roteiro1.parte3;

public class MysqlConnection implements Connection {

   public void connect() {
      System.out.println("Conectando com o MySQL");
   }

}
```

```
package roteiro1.parte3;

public class OracleConnection implements Connection{

   public void connect() {
      System.out.println("Conectando com o Oracle");
   }

}
```

5 – As classes de negócio (ServicoQuarto, ServicoReserva, RelatorioReserva) devem agora ser refatoradas. Elas não devem fazer referencia as classes concretas de conexão com o banco de dados, mas sim a interface. Esta ação reduz o forte acoplamento entre as classes de negócio e as classes de conexão com o banco. Segue o exemplo de uma das classes.

```
package roteiro1.parte3;

public class ServicoQuarto {
   private Connection  connection;

   public ServicoQuarto() {
     this.connection = new MysqlConnection();
   }

   public void verificarQuarto(){
      this.connection.connect();
      System.out.println("Lógica de negócio para o Serviço de Quarto");
   }
}
```

6 – Uma outra ação para reduzir o forte acoplamento entre as classes á fazer com que as classes de negócio recebam como parâmetro a conexão desejada. Mas, observe que o parâmetro não deve ser uma referência de classe concreta, mas sim a interface (ou abstrada). Segue o exemplo de uma das classes.

```
package roteiro1.parte3;

public class ServicoQuarto {
   private Connection  connection;

   public ServicoQuarto(Connection  c) {
     this.connection = c;
   }

   public void verificarQuarto(){
      this.connection.connect();
      System.out.println("Lógica de negócio para o Serviço de Quarto");
   }
}
```

7 – Faça as adaptações na classe TesteConexão e verifique o resultado final.

8 – Como seria a intervenção de código caso surgisse a necessidade de conectar com um banco de dados SQL Server por exemplo ? Faça as intervenções necessárias

9 – Utilize uma ferramenta de software qualquer para geração do diagrama de classes para esta etapa do projeto (Sugestão : Astah Community). Obs.: Adicione aqui o diagrama para que seja disponibilizado no teams.