02

Fábricas e o problema de criação de objetos

Em linguagens orientadas a objetos, estamos bastante acostumados a instanciar objetos o tempo todo. E isso é geralmente feito de forma fácil: basta dar um `new` no objeto e fazer uso da instância retornada.

Mas, às vezes essa criação pode não ser tão simples assim. Por exemplo, veja o código abaixo, onde nosso programa principal captura uma conexão com o banco de dados (fazendo uso de JDBC tradicional) e faz uso dela:

```
public class MeuAplicativo {

    public static void main(String[] args) throws SQLException {
        Connection conexao =
                DriverManager.getConnection("jdbc:mysql://localhost:3306/banco", "usuario", "se

        PreparedStatement ps = conexao.prepareStatement("select * from ...");
        // codigo continua aqui
    }
}
```

Veja a linha que cria uma conexão: `DriverManager.getConnection()` . Mas veja que só nessa linha temos muitas informações importantes: tipo do banco de dados (MySql), endereço do banco (localhost), nome do banco (banco), usuário e senha.

Agora imagine que devemos usar essa linha em todo lugar que precisa de uma conexão com o banco de dados. O que aconteceria se precisássemos trocar, por exemplo, o endereço do banco de dados?

Precisaríamos propagar a mudança para todos os pontos que pedem uma conexão. Isso significa um trabalho grande demais, e que com certeza, não faríamos corretamente.

Para resolver esse problema, podemos isolar esse processo de criação do banco de dados em uma classe específica, que só faz isso. Por exemplo:

```
public class ConnectionFactory {

    public Connection getConnection() {
        try {
            Connection conexao =
                    DriverManager.getConnection("jdbc:mysql://localhost:3306/banco", "usuario",

            return conexao;
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Veja que a classe acima agora cria a Conexão e retorna o objeto criado. Agora, todos que quiserem fazer uso de uma conexão devem fazer `new ConnectionFactory().getConnection()` .

Isso quer dizer que, se precisarmos mudar a string de conexão, basta agora mudar na classe que escrevemos, e a mudança será automaticamente propagada. Muito mais fácil!

Quando precisamos isolar o processo de criação de um objeto, para facilitar a troca dele no futuro, levamos o processo de instanciação dessa classe para uma `Factory`.

No primeiro curso, tínhamos também um exemplo de um objeto que é difícil de ser criado. Demos o exemplo da classe `NotaFiscal`. Lá, uma nota fiscal era composta por nome da pessoa, ítens da nota, valor do imposto, e etc. Tudo isso tornava o objeto difícil de ser criado, e portanto fizemos uso de um `Builder`.

Factories e Builders são classes cuja responsabilidade é lidar com o processo de criação de objetos complexos. Faça uso de Factories sempre que a criação de um objeto possa mudar em algum momento.