+Devs2Blu

Linguagem de programação JAVA

Profª. Heloisa Moura

Padrões de projetos (Design Patterns)

Padrão Estrutural
Bridge

Padrões de projetos Bridge

Problema

- Como é possível fazer com que a abstração e a implementação possam variar independentemente?
- Como esta implementação pode variar em tempo de execução?

Padrões de projetos Bridge

Exemplo de um cenário problema:

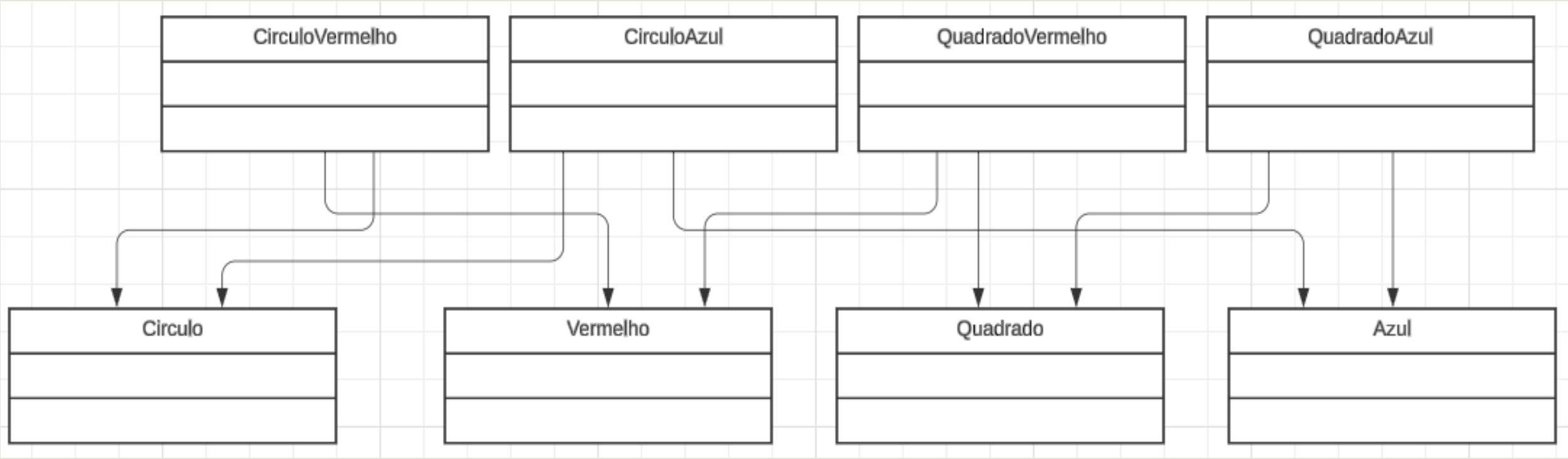
Imagine que você precisa de uma estrutura para representar formas geométricas como **Círculo** e **Quadrado**. Além disso, essas formas podem ter diferentes cores, como **Vermelho** ou **Azul**. Se você tentar criar classes para todas as combinações, terá algo como: **CírculoVermelho**, **CírculoAzul**, **QuadradoVermelho**, **QuadradoAzul**, e assim por diante, o que rapidamente cria muita complexidade.

Como mostra o diagrama de classes:

+Devs2Blu

Fundamentos avançados OOP

Diagrama de classes



Solução: Padrão de projeto: Bridge

- Definir um conjunto hierárquico para ambos os lados. Abstração e implementação.
- Em tempo de execução, será possível escolher a classe concreta para a abstração e para a implementação que são compatíveis graças as interfaces.

Solução: Padrão de projeto: Bridge

Definição: O Padrão de Projeto Bridge é um padrão estrutural que separa a abstração da implementação, permitindo que as duas evoluam independentemente.

- Ele é útil quando você quer evitar uma explosão de classes no código, especialmente quando há múltiplas combinações de abstrações e implementações.
  Em Java, você pode aplicar esse padrão usando interfaces e classes abstratas.

Solução: Padrão de projeto: Bridge

Voltando ao nosso problema de formas que podem ter diferentes cores.
O padrão Bridge ajuda a resolver esse problema separando as classes em duas hierarquias:

- **Abstração** (por exemplo, "Forma")
- **Implementação** (por exemplo, "Cor")
  Cada uma pode evoluir separadamente, facilitando a expansão do código.

Solução: Implementando o padrão Bridge em Java

1. Definindo a Interface de Implementação
Primeiro, criamos uma interface que representará a cor de uma forma.
Interface Cor
método: aplicarCor()

Classe concreta Vermelho
implementa: Cor
método: aplicarCor()

Classe concreta Azul
implementa: Cor
método: aplicarCor()

Solução: Implementando o padrão Bridge em Java

2. Criando a Abstração

Agora, criamos uma **classe abstrata** para representar a "Forma", que terá uma referência a um objeto do tipo Cor.

Classe abstrata: Forma
método abstrato: desenhar()

Classe concreta Circulo
extends: Forma
sobrescreve método: desenhar()

Classe concreta Quadrado
extends: Forma
sobrescreve método: desenhar()

Solução: Implementando o padrão Bridge em Java

3. Usando o Padrão Bridge na Classe Main

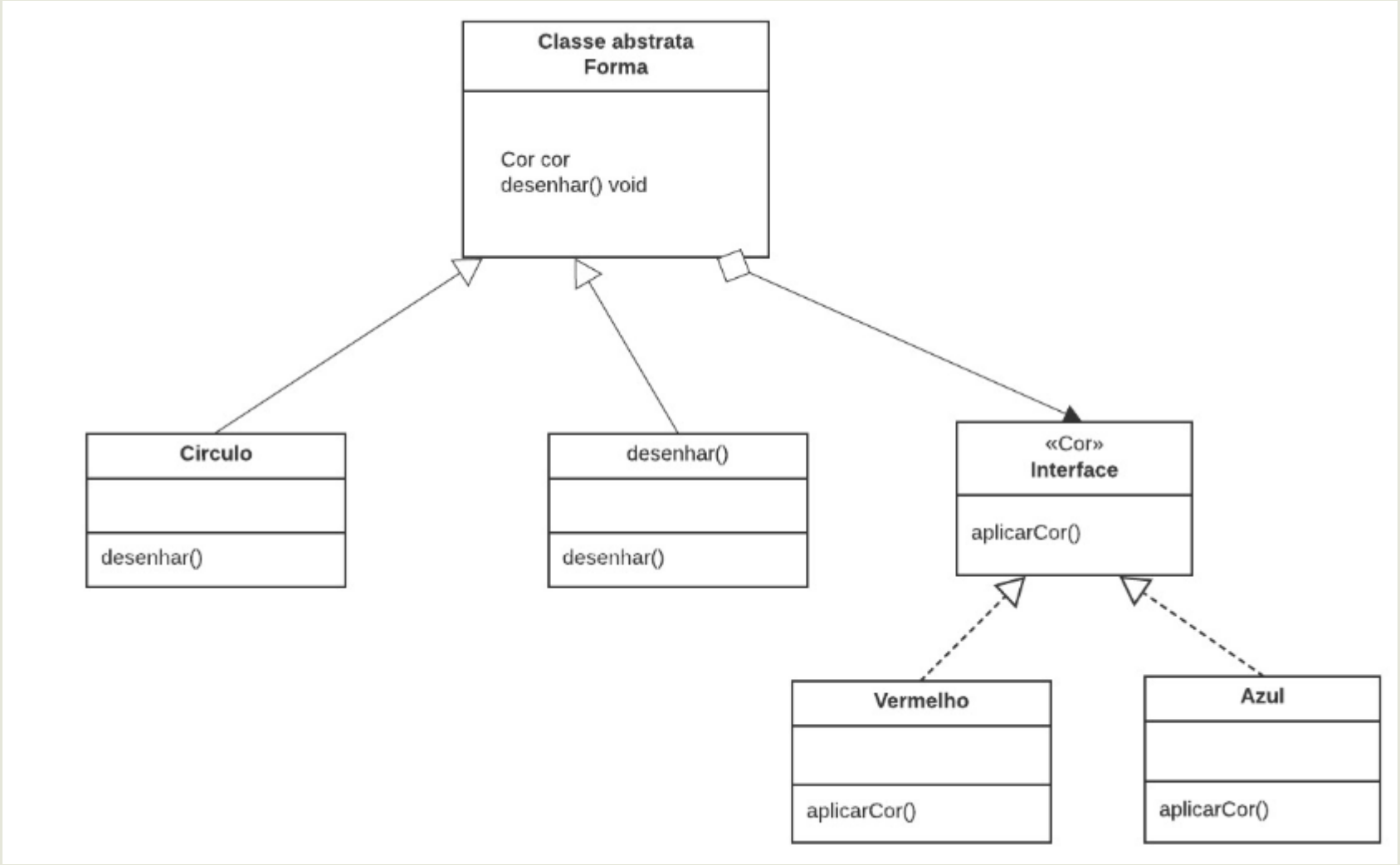
Agora podemos criar diferentes combinações de formas e cores sem criar uma nova classe para cada combinação.

Diagrama de classes com o padrão aplicado

+Devs2Blu

Fundamentos avançados OOP

Diagrama de classes



+Devs2Blu

Fundamentos avançados OOP

A implementação em Java

```java
//Interface de implementação
public interface Cor {
 public void aplicarCor();
}
```

```java
//Implementação Vermelho
public class Vermelho implements Cor {
 @Override
 public void aplicarCor() {
     System.out.println("Aplicando a cor vermelha");
 }
}
```

```java
//Implementação Azul
public class Azul implements Cor {
 @Override
 public void aplicarCor() {
     System.out.println("Aplicando a cor azul");
 }
}
```

```java
//Abstração
public abstract class Forma {
 protected Cor cor;

 // Construtor que recebe uma implementação de cor
 protected Forma(Cor cor) {
     this.cor = cor;
 }

public abstract void desenhar();
}
```

```java
//Abstração Refinada para um Círculo
public class Circulo extends Forma {
 public Circulo(Cor cor) {
     super(cor);
 }

 @Override
 public void desenhar() {
     System.out.print("Desenhando um círculo com ");
     cor.aplicarCor();
 }
}
```

A implementação em Java

```java
//Abstração Refinada para um Quadrado
public class Quadrado extends Forma {
 public Quadrado(Cor cor) {
      super(cor);
 }

 @Override
 public void desenhar() {
      System.out.print("Desenhando um quadrado com ");
      cor.aplicarCor();
 }
}
```

A implementação em Java

```
public class Main {
    public static void main(String[] args) {
        Forma circuloVermelho = new Circulo(new Vermelho());
        circuloVermelho.desenhar();

        Forma quadradoAzul = new Quadrado(new Azul());
        quadradoAzul.desenhar();
    }
}
```

Classes no Eclipse

Padrões de projetos Bridge

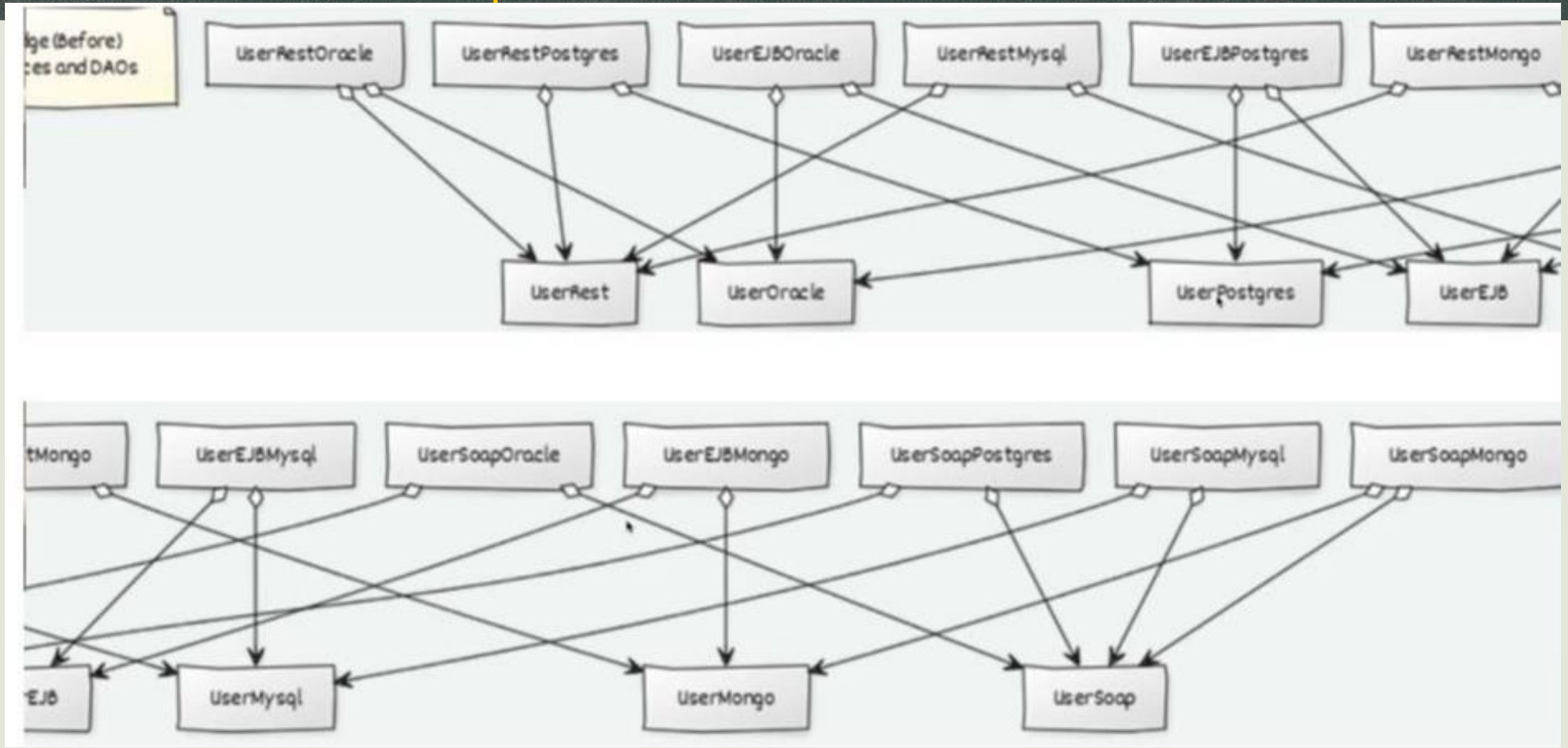
Outro cenário implementando o padrão Bridge:

Temos um sistema com varias tecnologias e queremos montar uma estrutura com varias combinações entre elas. Sem a utilização do padrão, todas essas combinações traria uma complexidade maior ao sistema.

Tecnologias que representam services	Tecnologias que representam DAOs
UseRest	User Oracle
UseEJB	UserPostgres
UseSoap	UserMySql
	UserMongo

+Devs2Blu

Fundamentos avançados OOP

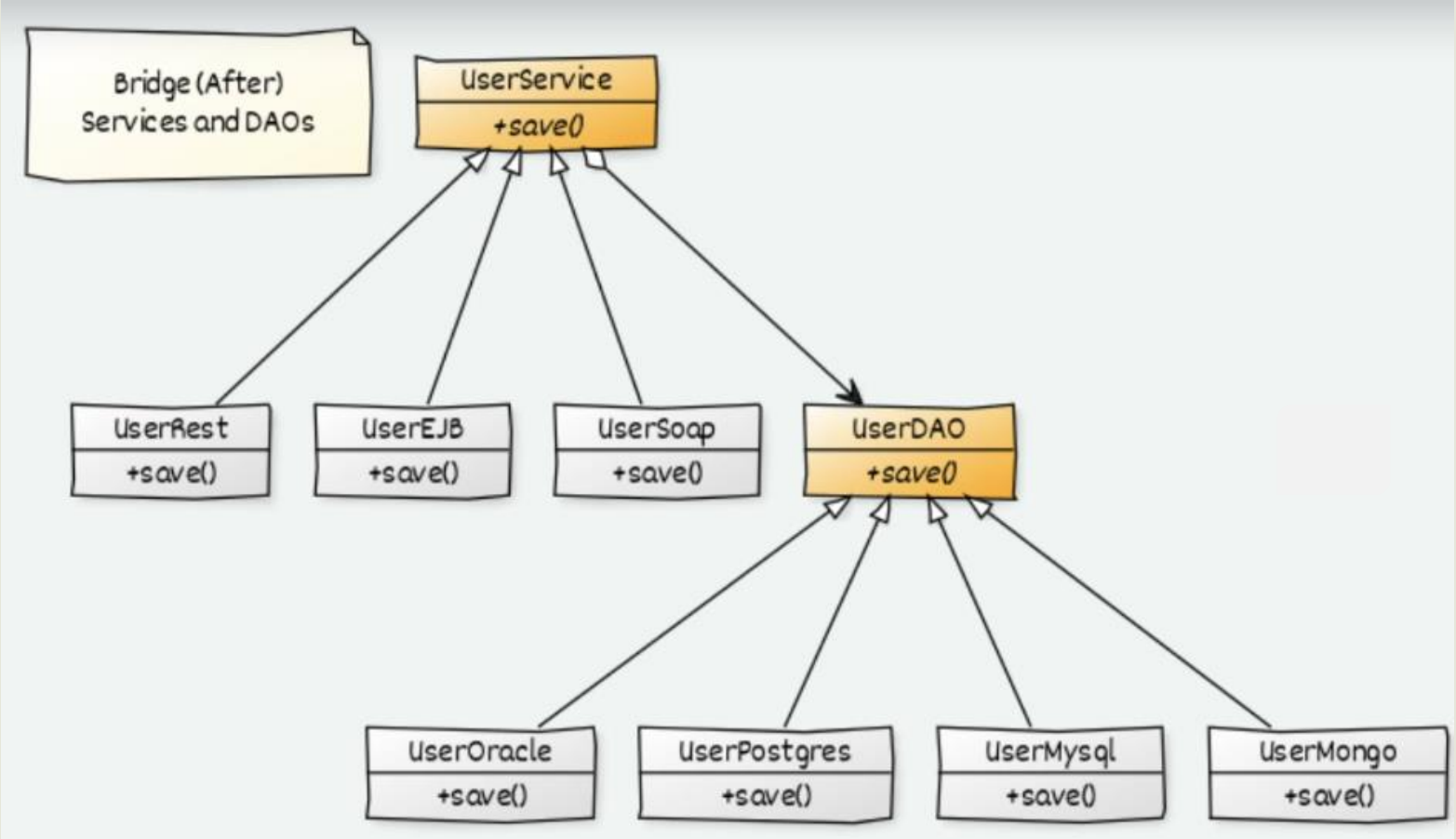


+Devs2Blu

Fundamentos avançados OOP

Utilizando o padrão temos essa estrutura no diagrama de classes

Vamos ver as implementações



Padrões de projetos Bridge

Resumo

O **Brigde** é um tipo de padrão extremamente útil pra que se consiga estabelecer a comunicação entre classes concreta sem ter uma explosão de combinações entre essas classes.

Padrões de projetos Bridge

Exercícios