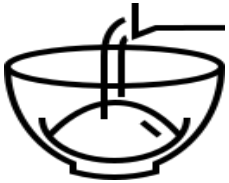
Exemplo de Template Method pattern

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

Exemplo de Template Method pattern

Vejamos um exemplo do padrão fazendo uma analogia com um exemplo da vida real. Vamos pensar em uma **pizzaria** e no processo de preparação de diferentes tipos de pizzas. Vamos tentar identificar as **etapas** que um cozinheiro deve seguir para **preparar e entregar uma pizza**. Poderiam ser:



Preparar a massa.

Pré-cozinhar a massa.

Preparar os ingredientes.

Adicionar os ingredientes.

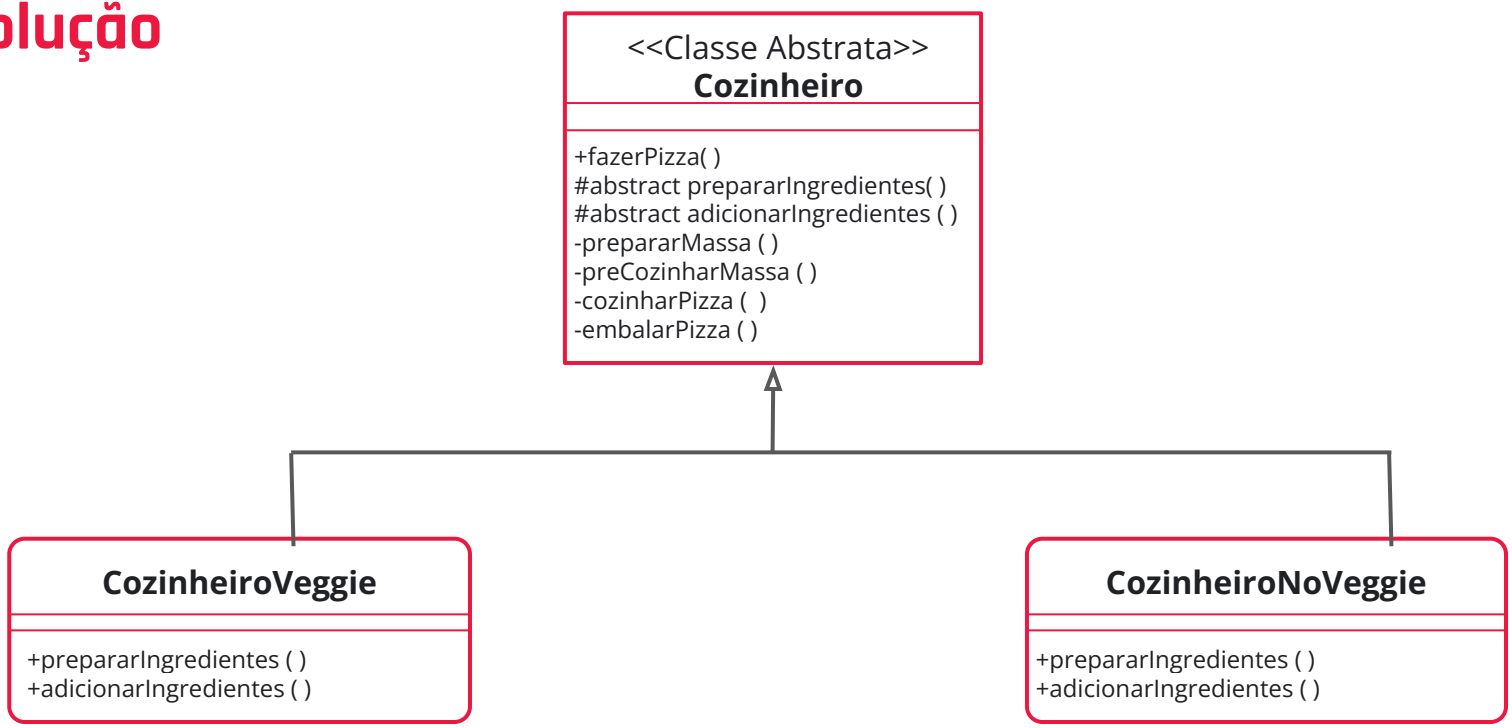
Cozinhar a pizza.

Embalar a pizza.

Para cada variedade de pizza, os cozinheiros precisam seguir todas essas etapas.

Aplicando o Template Method pattern, poderíamos criar o método esqueleto, com as etapas comuns a todas as pizzas e deixar os cozinheiros se preocuparem apenas com as etapas que não são comuns a todas as pizzas, neste caso, **preparar os ingredientes e adicionar os ingredientes**. Vamos ver como representamos essa solução em um diagrama UML.

Solução


<<Classe Abstrata>>
Cozinheiro
+fazerPizza( )
#abstract prepararIngredientes( )
#abstract adicionarIngredientes ( )
-prepararMassa ( )
-preCozinharMassa ( )
-cozinharPizza ( )
-embalarPizza ( )
CozinheiroVeggie
+prepararIngredientes ( )
+adicionarIngredientes ( )
CozinheiroNoVeggie
+prepararIngredientes ( )
+adicionarIngredientes ( )

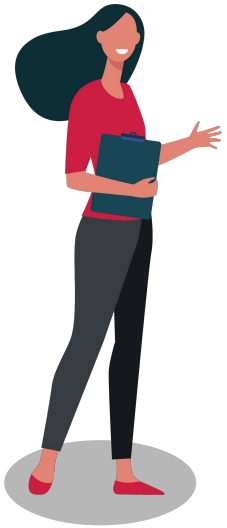
Implementação das classes do diagrama UML

```
public abstract class Cozinheiro {

    public void fazerPizza(){
        prepararMassa();
        preCozinharMassa();
        prepararIngredientes();
        adicionarIngredientes();
        cozinharPizza();
        embalarPizza();

    }
```

Código da classe abstrata que define o método esqueleto e os métodos abstratos a serem implementados pelas subclasses.

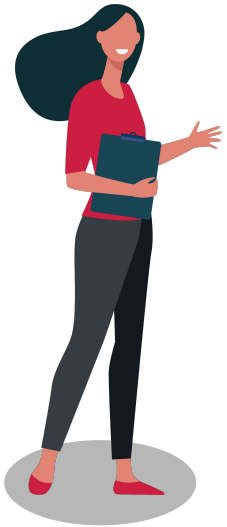
```
    protected abstract void prepararIngredientes();
    protected abstract void  adicionarIngredientes();
    private void prepararMassa(){
        System.out.println("Preparando a massa...");
    }
    private void preCozinharMassa(){
        System.out.println("Pré-cozinhando a massa...");
    }
    private void cozinharPizza(){
        System.out.println("Enviando a pizza ao forno");
    }
    private void embalarPizza(){
        System.out.println("Embalando a pizza");
    }
}
```

Código da classe abstrata que define o método esqueleto e os métodos abstratos que serão implementados pelas subclasses.



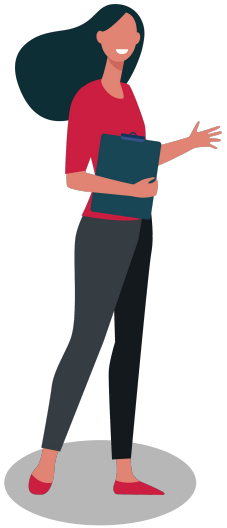
```java
public class CozinheiroNoVeggie extends Cozinheiro {
    @Override
    protected void prepararIngredientes() {
        System.out.println("Preparando queijo e presunto");
    }

    @Override
    protected void adicionarIngredientes() {
        System.out.println("Adicionando oa ingredientes");
    }
}
```

Código da subclasse **CozinheiroNoVeggie**



```java
public class CozinheiroVeggie extends Cozinheiro {


    @Override
    protected void prepararIngredientes() {
        System.out.println("Preparando tomate e queijos");
    }

    @Override
    protected void adicionarIngredientes() {
        System.out.println("Adicionando queijos e tomate");

    }
}
```

Código da subclasse **CozinheiroVeggie**



```java
public class Main {

    public static void main(String[] args) {
        Cozinheiro cozinheiroVeggie = new CozinheiroVeggie();
        Cozinheiro cozinheiroNoVeggie = new CozinheiroNoVeggie();

        cozinheiroVeggie.fazerPizza();
        cozinheiroNoVeggie.fazerPizza();
    }
}
```

Test


DigitalHouse>
Coding School