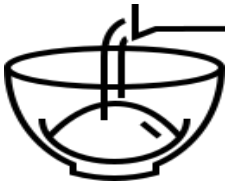
Ejemplo de template method

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

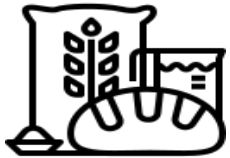
Ejemplo de patrón template method

Veamos un ejemplo del patrón haciendo una analogía con un ejemplo de la vida real. Pensemos en una **pizzería** y en el proceso de preparar diferentes tipos de pizzas. Intentemos identificar los **pasos** que un cocinero debe realizar para **cocinar y entregar una pizza**. Podrían ser:



Preparar la masa.

Precocinar la masa.



Preparar los ingredientes.

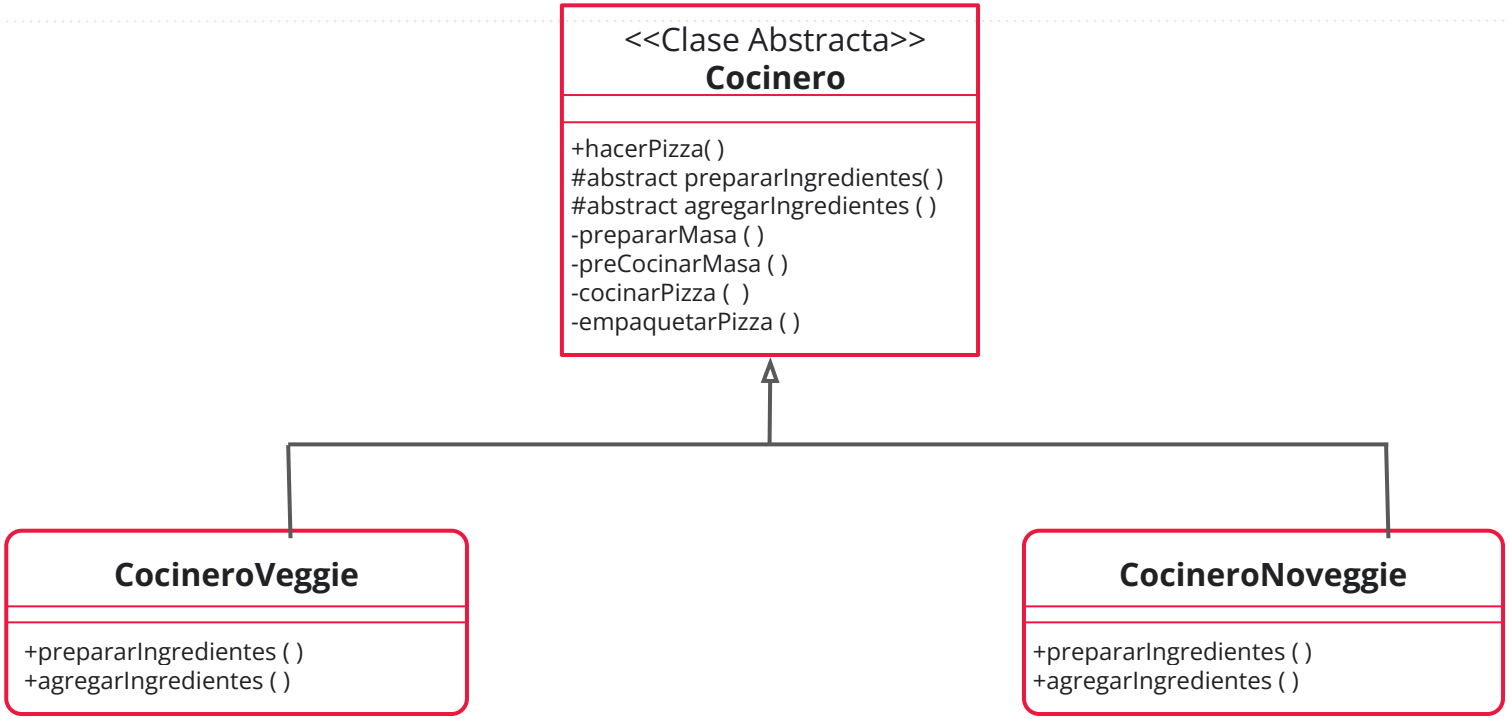
Agregar los ingredientes.

Cocinar la pizza.

Empaquetar la pizza

Por cada variedad de pizza, los cocineros tienen que hacer todos esos pasos. Aplicando el patrón template method, podríamos crear el método esqueleto, con los pasos comunes a todas las pizzas y dejar que los cocineros solo se preocupen por los pasos que no son comunes a todas las pizzas, en este caso, **preparar los ingredientes y agregar los ingredientes.** Veamos cómo representamos esta solución en un diagrama UML.

Solución


<<Clase Abstracta>>
Cocinero
+hacerPizza( )
#abstract prepararIngredientes( )
#abstract agregarIngredientes ( )
-prepararMasa ( )
-preCocinarMasa ( )
-cocinarPizza ( )
-empaquetarPizza ( )
CocineroVeggie
+prepararIngredientes ( )
+agregarIngredientes ( )
CocineroNoveggie
+prepararIngredientes ( )
+agregarIngredientes ( )

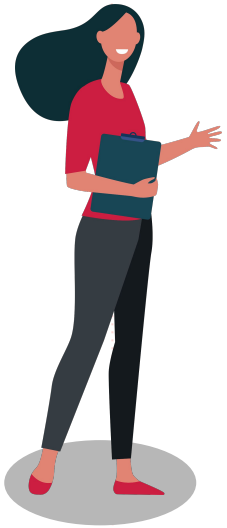
Implementación de las cases del diagrama UML

```
public abstract class Cocinero {

    public void hacerPizza(){
        prepararMasa();
        preCocinarMasa();
        prepararIngredientes();
        agregarIngredientes();
        cocinarPizza();
        empaquetarPizza();

    }
```

Código de la clase abstracta que define el método esqueleto y los métodos abstractos a implementar por las subclases.

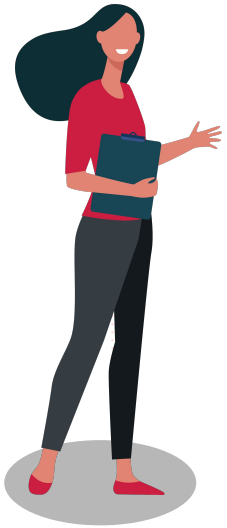
```
    protected abstract void prepararIngredientes();
    protected abstract void  agregarIngredientes();
    private void prepararMasa(){
        System.out.println("Preparando masa..");
    }
    private void preCocinarMasa(){
        System.out.println("Pre cocinando masa..");
    }
    private void cocinarPizza(){
        System.out.println("Enviando al horno la pizza");
    }
    private void empaquetarPizza(){
        System.out.println("Empaquetando pizza");
    }
}
```

Código de la clase abstracta que define el método esqueleto y los métodos abstractos a implementar por las subclases.



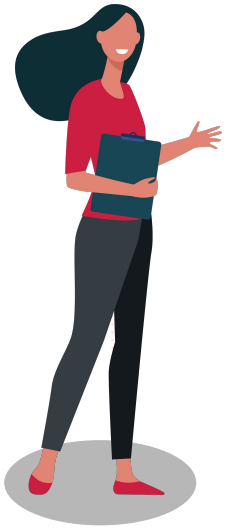
```java
public class CocineroNoVeggie extends Cocinero {
    @Override
    protected void prepararIngredientes() {
        System.out.println("Preparando queso y jamón,");
    }

    @Override
    protected void agregarIngredientes() {
        System.out.println("Agregando los ingredientes");
    }
}
```

Código de la subclase **CocineroNoVeggie**



```java
public class CocineroVeggie extends Cocinero {


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

    @Override
    protected void agregarIngredientes() {
        System.out.println("Agregando quesos y tomate");

    }
}
```

Código de la subclase **CocineroVeggie (esta clase validará los datos en Facebook)**



```java
public class Main {

    public static void main(String[] args) {
        Cocinero cocineroVeggie = new CocineroVeggie();
        Cocinero cocineroNoVeggie = new CocineroNoVeggie();

        cocineroVeggie.hacerPizza();
        cocineroNoVeggie.hacerPizza();
    }
}
```

Test


DigitalHouse>
Coding School