

Ejemplo de Patrón Flyweight

DigitalHouse >
Coding School



**Certified Tech
Developer**
The Ultimate Degree

Ejemplo de patrón flyweight

Supongamos que tenemos un **recetario** que contiene **recetas** que están en diferentes colecciones como carne; sopas; ensaladas; o en diferentes **categorías** como comida italiana; mexicana; argentina; rápidas.



Ejemplo de patrón flyweight

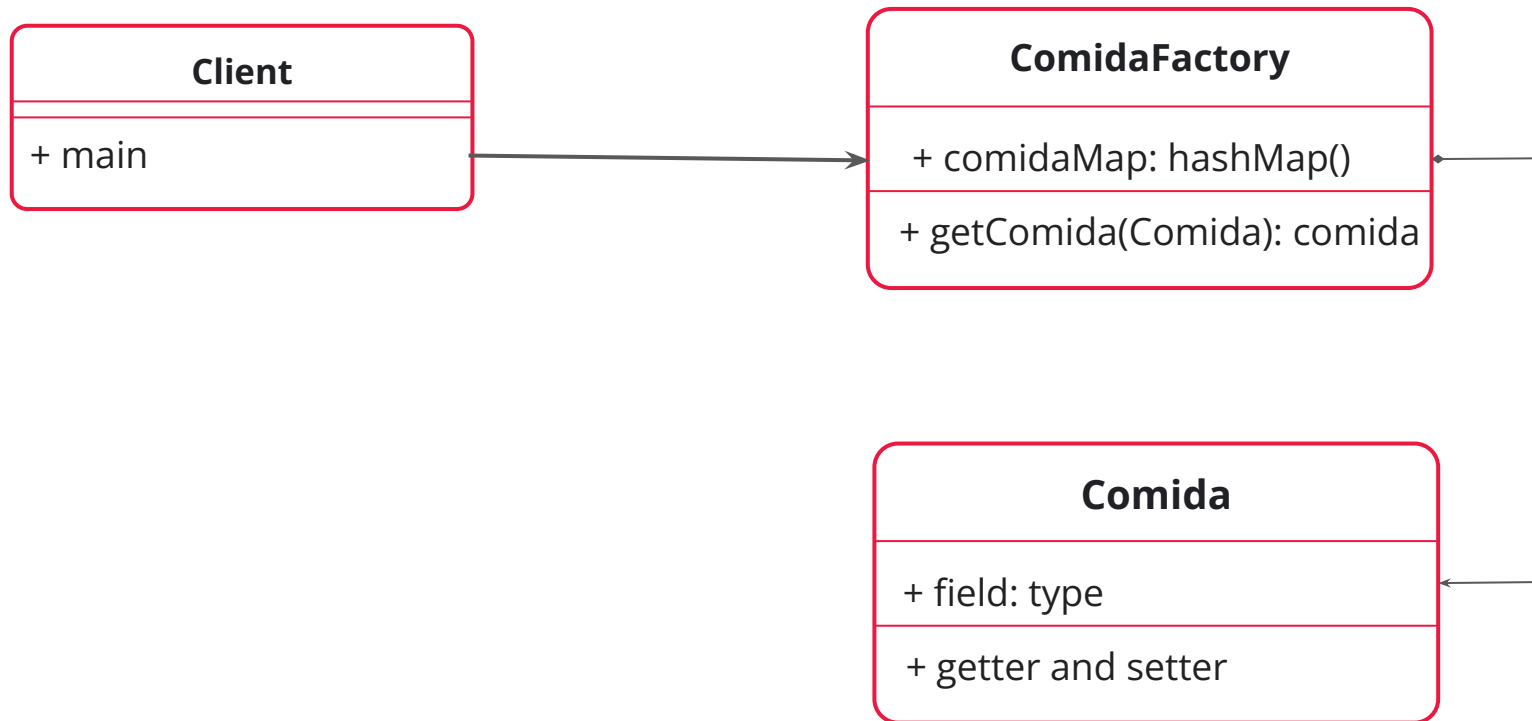
La receta para una hamburguesa podría estar en varias **secciones**: americana, carnes, rapidas, etc. Si necesitáramos tener un objeto receta hamburguesa en cada una de las colecciones sería muy poco performante y se usaría mucha memoria.

Entonces, el cliente pide un objeto a la FlyweightFactory, esta se fija si existe en el caché y en caso contrario crea uno nuevo. Flyweight comparte el estado de los objetos.

Veamos cómo representamos esta solución en un diagrama UML.



Solución



Implementación del diagrama UML

Código de la **clase** que define el método antes de crear el objeto. Valida si ya existe un objeto idéntico al que se le está solicitando. De ser así, regresa el objeto existente; de no existir, crea el nuevo objeto y lo almacena en caché para ser reutilizado más adelante.

```
import java.util.HashMap;  
  
public class ComidaFactory {  
    private static final HashMap<String, Comida> comidaMap = new HashMap();  
  
    public static Comida getComida(String tipoComida) {  
        Comida comida = (Comida) comidaMap.get(tipoComida);
```

Implementación del diagrama UML

```
if (comida == null) {  
    comida = new Comida(tipoComida);  
    comidaMap.put(tipoComida, comida);  
    System.out.println("Creando objeto comida de tipo: "  
+ tipoComida);  
}  
return comida;  
}  
}
```

Código de la
clase

Implementación del diagrama UML

```
public class Comida {  
    private String tipoComida;  
    private int precio;  
    private boolean tieneLechuga;  
  
    public Comida(String tipoComida) {  
        this.tipoComida = tipoComida;  
    }  
  
    public String getTipoComida () {  
        return tipoComida;  
    }  
}
```

Objeto: comida con atributos precio, tipo de comida y si tiene lechuga.

Implementación del diagrama UML

```
public int getPrecio() {  
    return precio;  
}  
  
public void setPrecio(int precio) {  
    this.precio = precio;  
}  
public boolean isTieneLechuga() {  
    return tieneLechuga;  
}  
  
public void setTieneLechuga(boolean tieneLechuga) {  
    this.tieneLechuga = tieneLechuga;  
}
```

Objeto: comida con
atributos precio, tipo de
comida y si tiene lechuga.

Implementación del diagrama UML

```
public void descripcionDeLaComida ()  
{System.out.println("Es un/una " + getTipoComida()  
+ " que sale: " + getPrecio());  
}}
```

Objeto: comida con
atributos precio, tipo de
comida y si tiene lechuga.

DigitalHouse>
Coding School