Parcial de Diagramas UML

Por:

Samuel David Colmenarez Quero

Docente:

Christian David Jaimes Acevedo

Facultad de Ingenierías

Ingeniería de Software

Tecnológico de Antioquia - Institución Universitaria

Medellín, Colombia

2025

Descripción de la actividad

La empresa gastronómica **CookMaster** necesita el diseño de un sistema para gestionar recetas culinarias. El sistema debe permitir crear, organizar y consultar recetas de diferentes tipos. El objetivo es modelar el sistema en **UML de clases** aplicando **uno o dos patrones de diseño** vistos en clase: Singleton, Factory o Builder.

Diagrama de Clase

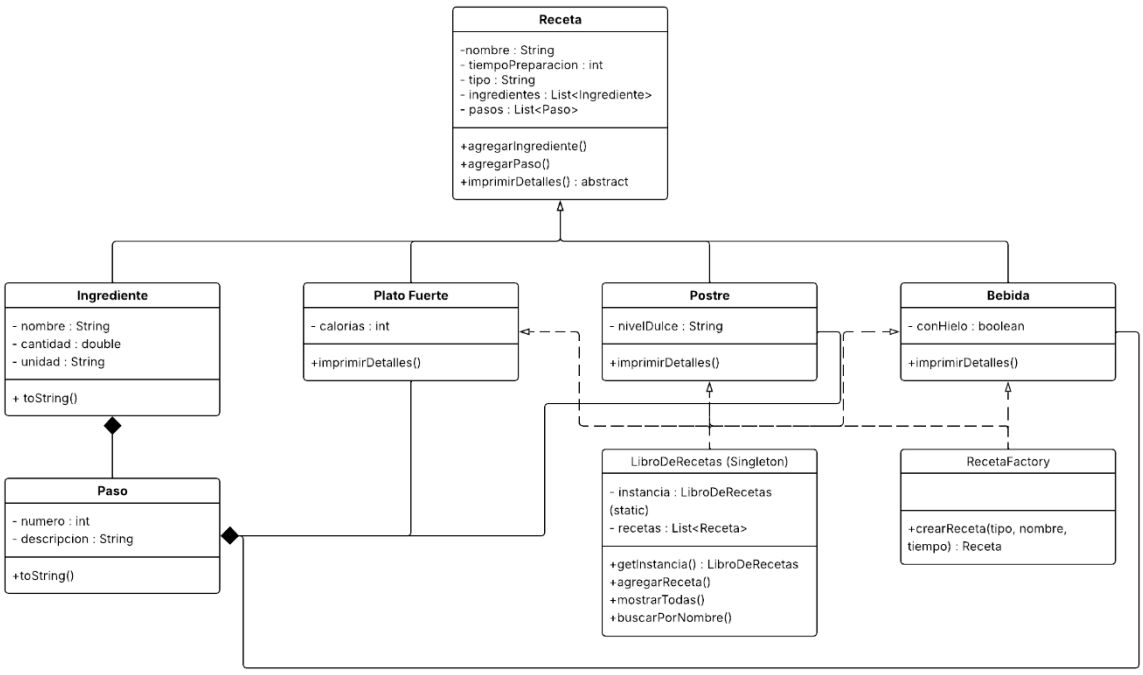

Ilustración 1. Diagrama de clases del ejercicio

Código

```java
package Semana2.cookmaster.recetas;
import Semana2.cookmaster.ingredientes.Ingrediente;
import Semana2.cookmaster.pasos.Paso;
import java.util.ArrayList;
import java.util.List;

public abstract class Receta {
    protected String nombre;
    protected int tiempoPreparacion;
    protected String tipo;

    protected List<Ingrediente> ingredientes = new ArrayLis
    protected List<Paso> pasos = new ArrayList<>();

    public Receta(String nombre, int tiempo) {
        this.nombre = nombre;
        this.tiempoPreparacion = tiempo;
    }

    public void agregarIngrediente(Ingrediente ing) {
        ingredientes.add(ing);
    }

    public void agregarPaso(Paso paso) {
        pasos.add(paso);
    }

    public String getNombre() { return nombre; }

    public abstract void imprimirDetalles();
}
```

```java
package Semana2.cookmaster.ingredientes;

public class Ingrediente {
        private String nombre;
    private double cantidad;
    private String unidad;

    public Ingrediente(String nombre, double cantidad, String unidad) {
        this.nombre = nombre;
        this.cantidad = cantidad;
        this.unidad = unidad;
    }

    @Override
    public String toString() {
        return nombre + " " + cantidad + " " + unidad;
    }

}
```

```java
package Semana2.cookmaster.pasos;

public class Paso {
    private int numero;
    private String descripcion;

    public Paso(int numero, String descripcion) {
        this.numero = numero;
        this.descripcion = descripcion;
    }

    @Override
    public String toString() {
        return numero + ". " + descripcion;
    }

}
```

```java
package Semana2.cookmaster.recetas.tipos;
import Semana2.cookmaster.recetas.Receta;

public class Bebida extends Receta {
    private boolean conHielo;

    public Bebida(String nombre, int tiempo, boolean conHielo) {
        super(nombre, tiempo);
        this.tipo = "Bebida";
        this.conHielo = conHielo;
    }

    @Override
    public void imprimirDetalles() {
        System.out.println(x: "=== Detalle de Receta ===");
        System.out.println("Nombre: " + nombre);
        System.out.println("Tipo: " + tipo);
        System.out.println("Tiempo de preparación: " + tiempoPreparacion + " min");

        System.out.println(x: "\nIngredientes:");
        int i = 1;
        for (var ing : ingredientes)
            System.out.println(" " + i++ + ") " + ing);

        System.out.println(x: "\nPasos:");
        for (var p : pasos)
            System.out.println(" " + p);

        System.out.println(x: "\nAtributos adicionales:");
        System.out.println(" - Con hielo: " + (conHielo ? "Sí" : "No"));
    }
```

```java
package Semana2.cookmaster.recetas.tipos;
import Semana2.cookmaster.recetas.Receta;

public class PlatoFuerte extends Receta {
    private int calorias;

    public PlatoFuerte(String nombre, int tiempo, int calorias) {
        super(nombre, tiempo);
        this.tipo = "Plato Fuerte";
        this.calorias = calorias;
    }

    @Override
    public void imprimirDetalles() {
        System.out.println(x: "=== Detalle de Receta ===");
        System.out.println("Nombre: " + nombre);
        System.out.println("Tipo: " + tipo);
        System.out.println("Tiempo de preparación: " + tiempoPreparacion + " min");

        System.out.println(x: "\nIngredientes:");
        int i = 1;
        for (var ing : ingredientes)
            System.out.println(" " + i++ + ") " + ing);

        System.out.println(x: "\nPasos:");
        for (var p : pasos)
            System.out.println(" " + p);

        System.out.println(x: "\nAtributos adicionales:");
        System.out.println(" - Calorías: " + calorias);
    }
```

```java
package Semana2.cookmaster.recetas.tipos;
import Semana2.cookmaster.recetas.Receta;

public class Postre extends Receta {
    private String nivelDulce;

    public Postre(String nombre, int tiempo, String nivelDulce) {
        super(nombre, tiempo);
        this.tipo = "Postre";
        this.nivelDulce = nivelDulce;
    }

    @Override
    public void imprimirDetalles() {
        System.out.println(x: "=== Detalle de Receta ===");
        System.out.println("Nombre: " + nombre);
        System.out.println("Tipo: " + tipo);
        System.out.println("Tiempo de preparación: " + tiempoPreparacion + " min");

        System.out.println("\nIngredientes (" + ingredientes.size() + "):");
        int i = 1;
        for (var ing : ingredientes)
            System.out.println(" " + i++ + ") " + ing);

        System.out.println("\nPasos (" + pasos.size() + "):");
        for (var p : pasos)
            System.out.println(" " + p);

        System.out.println(x: "\nAtributos adicionales:");
        System.out.println(" - Nivel de dulce: " + nivelDulce);
    }
}
```

```java
package Semana2.cookmaster.recetas.factory;
import Semana2.cookmaster.recetas.Receta;
import Semana2.cookmaster.recetas.tipos.*;

public class RecetaFactory {

    public static Receta crearReceta(String tipo, String nombre, int tiempo) {

        switch (tipo.toUpperCase()) {

            case "POSTRE":
                return new Postre(nombre, tiempo, nivelDulce: "Medio");

            case "BEBIDA":
                return new Bebida(nombre, tiempo, conHielo: true);

            case "PLATO":
            case "PLATOFUERTE":
                return new PlatoFuerte(nombre, tiempo, calorias: 500);

            default:
                throw new IllegalArgumentException("Tipo de receta desconocido: " + tipo);
        }
    }
}
```

```java
package Semana2.cookmaster.libro;
import Semana2.cookmaster.recetas.Receta;

import java.util.ArrayList;
import java.util.List;

public class LibroDeRecetas {

    private static LibroDeRecetas instancia;

    private List<Receta> recetas = new ArrayList<>();

    private LibroDeRecetas() {}

    public static LibroDeRecetas getInstancia() {
        if (instancia == null)
            instancia = new LibroDeRecetas();
        return instancia;
    }

    public void agregarReceta(Receta r) {
        recetas.add(r);
        System.out.println(x: "Receta guardada en el Libro de Recetas.");
    }

    public void mostrarTodas() {
        System.out.println(x: "\nListado de recetas:");
        int i = 1;
        for (Receta r : recetas)
            System.out.println(i++ + ". " + r.getNombre());
    }
```

```java
package Semana2.cookmaster;
import Semana2.cookmaster.pasos.Paso;
import Semana2.cookmaster.ingredientes.Ingrediente;
import Semana2.cookmaster.recetas.Receta;
import Semana2.cookmaster.recetas.factory.RecetaFactory;
import Semana2.cookmaster.libro.LibroDeRecetas;

public class Main {
    Run | Debug
    public static void main(String[] args) {

        LibroDeRecetas libro = LibroDeRecetas.getInstancia();

        // Crear postre
        Receta tarta = RecetaFactory.crearReceta(tipo: "POSTRE", nombre: "Tiramisú", tiempo: 40);
        tarta.agregarIngrediente(new Ingrediente(nombre: "Café", cantidad: 200, unidad: "ml"));
        tarta.agregarIngrediente(new Ingrediente(nombre: "Azúcar", cantidad: 80, unidad: "g"));

        tarta.agregarPaso(new Paso(numero: 1, descripcion: "Preparar café fuerte."));
        tarta.agregarPaso(new Paso(numero: 2, descripcion: "Mezclar mascarpone."));

        libro.agregarReceta(tarta);

        // Crear bebida
        Receta limonada = RecetaFactory.crearReceta(tipo: "BEBIDA", nombre: "Limonada Natural", tiemp
        limonada.agregarIngrediente(new Ingrediente(nombre: "Agua", cantidad: 500, unidad: "ml"));
        limonada.agregarIngrediente(new Ingrediente(nombre: "Limón", cantidad: 2, unidad: "u"));

        limonada.agregarPaso(new Paso(numero: 1, descripcion: "Exprimir limones."));
        limonada.agregarPaso(new Paso(numero: 2, descripcion: "Mezclar y servir."));

        libro.agregarReceta(limonada);
```

Breve justificación del diseño utilizado

El diseño del sistema CookMaster utiliza **Singleton** para asegurar que exista una única instancia central de administración, evitando inconsistencias en la gestión de recetas, usuarios y pedidos. Esto garantiza un control global y un acceso coordinado a los recursos compartidos del taller. Por otro lado, se emplea **Factory Method** para la creación flexible de distintos tipos de platillos y procesos de cocina. Este patrón permite extender el sistema sin modificar el código existente, facilitando la incorporación de nuevas recetas o variantes. En conjunto, ambos patrones aportan orden, escalabilidad y facilidad de mantenimiento.