

Patrones de Diseño Singleton y Builder.

- **Patrón de diseño Singleton:** Es un patrón creacional que permite restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto, garantiza que una clase tenga una única instancia, permite acceder a un objeto desde cualquier parte del programa, los valores se mantienen y son compartidos por toda la aplicación.

Ejemplo: En una empresa varios empleados utilizan una sola impresora.



Si un empleado envía una solicitud a la impresora, Singleton pregunta (¿ya hay un objeto de la impresora? Si la respuesta es no, entonces crea uno. Para evitar el acceso y los cambios, las variables individuales y la impresora quedan como privadas”.

```
public class impresora {  
    private static impresora;  
    private int NúmeroPáginas;  
    private impresora() {  
    }  
    public static impresora getInstance() {  
        return impresora == Null ?  
            impresora = new impresora() :  
            impresora;  
    }  
    public void print(String text){  
        System.out.println(text +  
            "\n" + "número de páginas impresas hoy" + ++ NumeroPáginas +  
            "\n" + "-----");  
    }  
}
```

Aquí se encapsulan los empleados. Las cadenas para sus nombres, la posición y función quedan como “privados”.

```
public class Employee {
    private final String name;
    private final String position;
    private final String role;
    public empleado(String name, String position, String role) {
        this.nombre = nombre;
        this.posición = posición;
        this.función = función;
    }
    public void printCurrent función (){
        impresora = impresora.getInstance();
        impresora.print("empleado: " + nombre + "\n" +
            "Posición: " + posición + "\n" +
            " Función: " + función + "\n");
    }
}
```

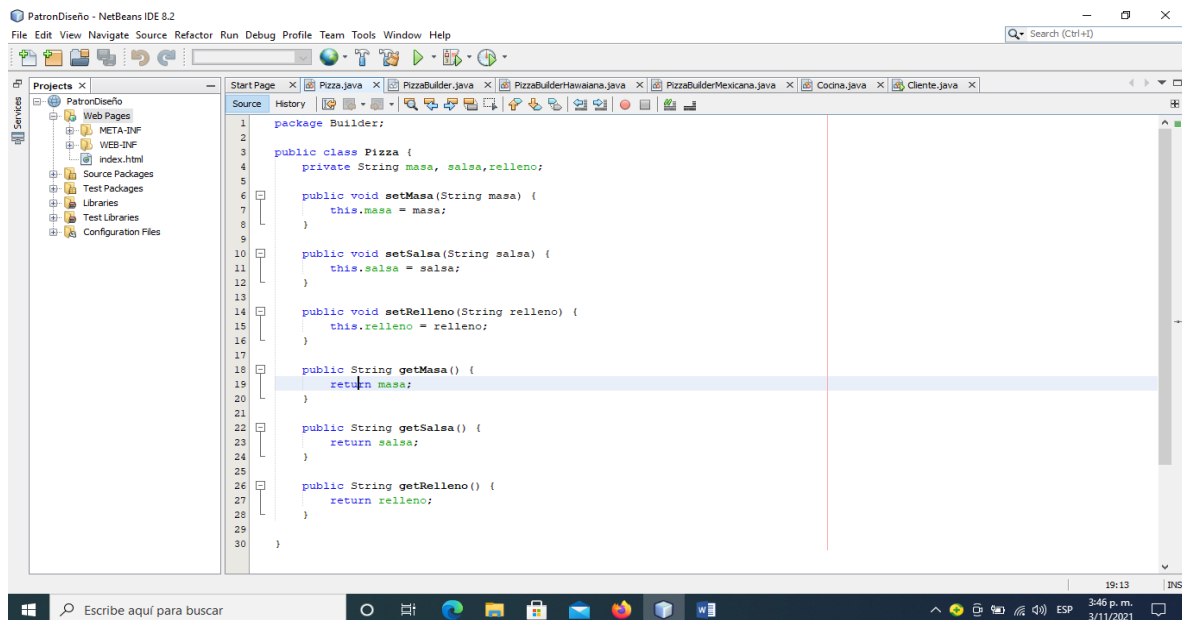
Finalmente, los dos Singleton se integran en una rutina de salida.

```
public class Main {
    public static void main(String[] args) {
        Empleado andreas = new empleado ("Andreas",
            "Jefe",
            "Gestiona la sucursal");
        Empleado julia = new empleado ("Julia",
            "Consultor",
            "Asesora a los clientes sobre las quejas");
        Empleado tom = new empleado ("Tom",
            "Venta",
            "Vende los productos");
        Empleado stefanie = new empleado ("Stefanie",
            "Desarrollador",
            " Mantenimiento informático en la sucursal.");
        Empleado matthias = new empleado ("Matthias",
            "Contable",
            "contabilidad financiera de la sucursal.");
        andreas.printCurrentFunción ();
        julia.printCurrentFunción ();
        tom.printCurrentFunción ();
        stefanie.printCurrentFunción ();
        matthias.printCurrentFunción ();
    }
}
```

- **Patrón de diseño Builder:** Se caracteriza por construir objetos de una sola clase, permite la creación de una variedad de objetos desde un objeto fuente, simplifica el desarrollo del software en procesos repetitivos.

Ejemplo: En una pizzería la creación de pizzas.

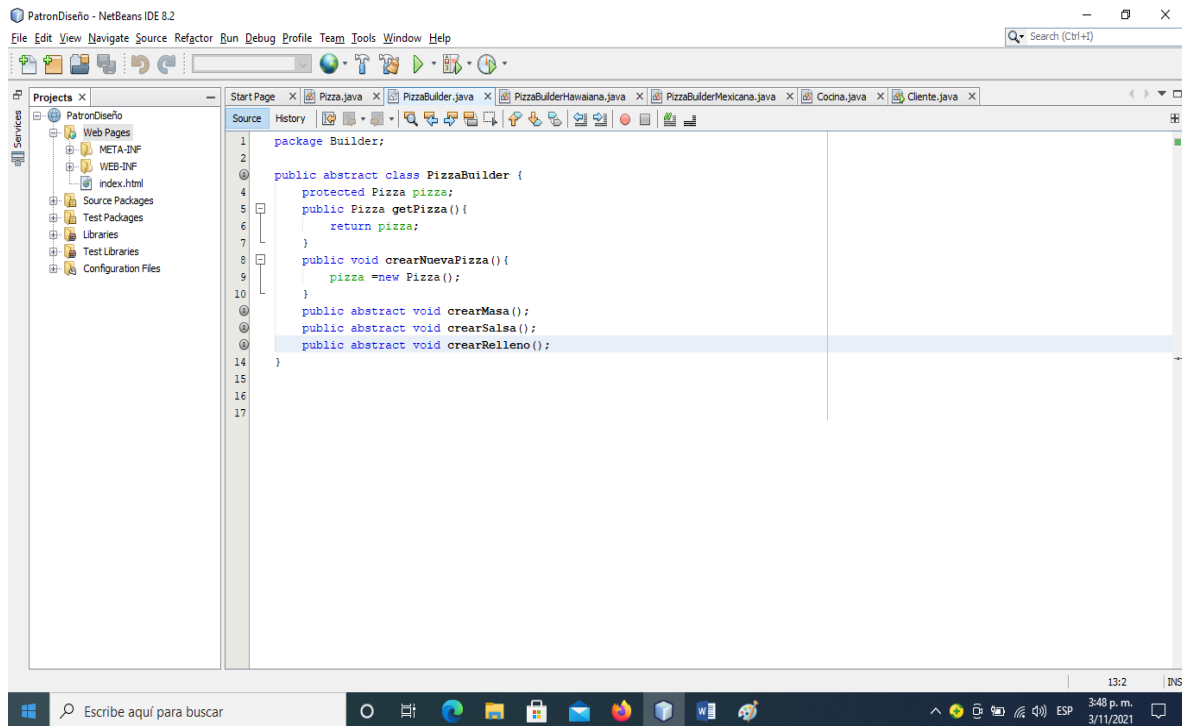
En el primer código se crea clase producto (Pizza), se crean solo los objetos de esta clase, Builder les asigna valores predeterminados a los atributos (masa, salsa, relleno).



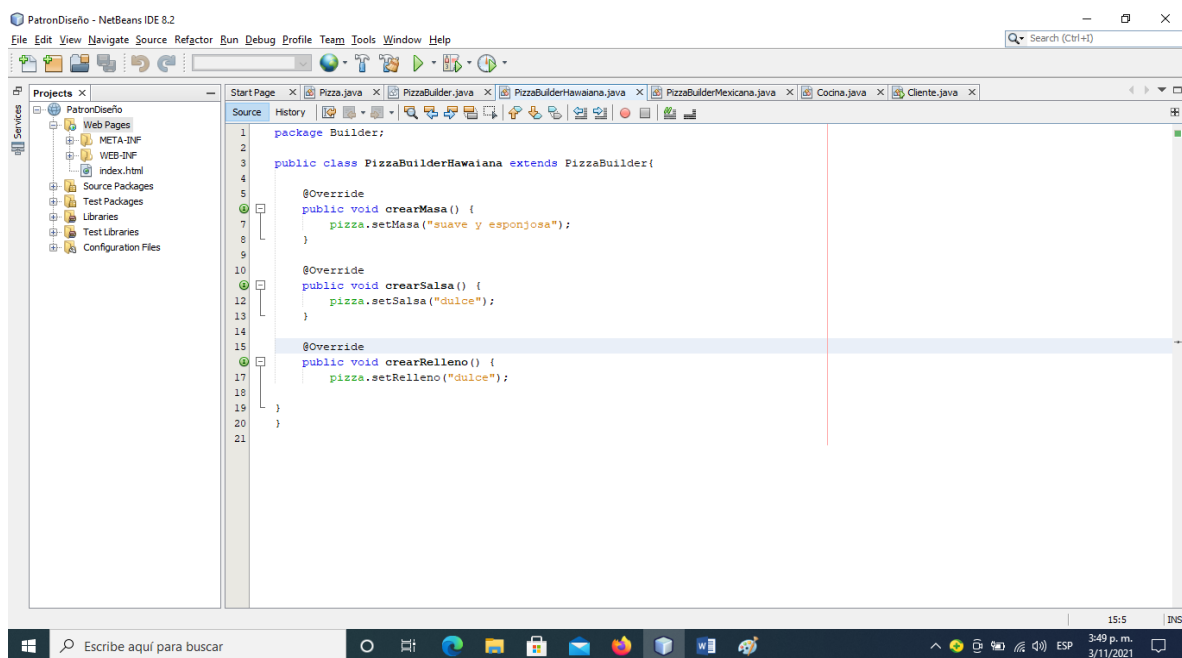
The screenshot shows the NetBeans IDE interface. The 'Projects' pane on the left displays a project named 'PatronDiseño' with various sub-projects like 'Web Pages', 'META-INF', 'index.html', 'Source Packages', 'Test Packages', 'Libraries', 'Test Libraries', and 'Configuration Files'. The main editor window shows the 'Pizza.java' file. The code defines a 'package Builder;' and a 'public class Pizza' with private attributes 'masa', 'salsa', and 'relleno'. It includes setter methods 'setMasa', 'setSalsa', and 'setRelleno', and getter methods 'getMasa', 'getSalsa', and 'getRelleno'. The status bar at the bottom indicates the time as 19:13 and the date as 3/11/2021.

```
1 package Builder;
2
3 public class Pizza {
4     private String masa, salsa, relleno;
5
6     public void setMasa(String masa) {
7         this.masa = masa;
8     }
9
10    public void setSalsa(String salsa) {
11        this.salsa = salsa;
12    }
13
14    public void setRelleno(String relleno) {
15        this.relleno = relleno;
16    }
17
18    public String getMasa() {
19        return masa;
20    }
21
22    public String getSalsa() {
23        return salsa;
24    }
25
26    public String getRelleno() {
27        return relleno;
28    }
29
30 }
```

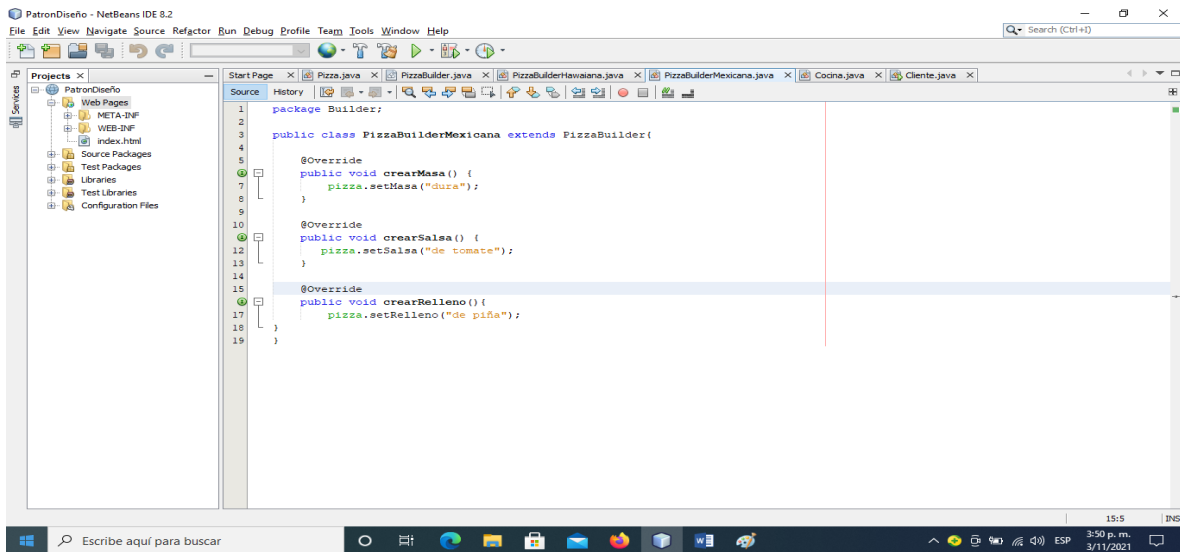
En el siguiente código encontramos el constructor abstracto (PizzaBuilder), tiene a un objeto de la clase producto (Pizza), tiene los métodos concretos (getPizza) para obtener el producto luego de que sea creado y un método para instanciar a pizza (crearNuevaPizza), tiene métodos abstractos que se encargan de asignar valores predeterminados.



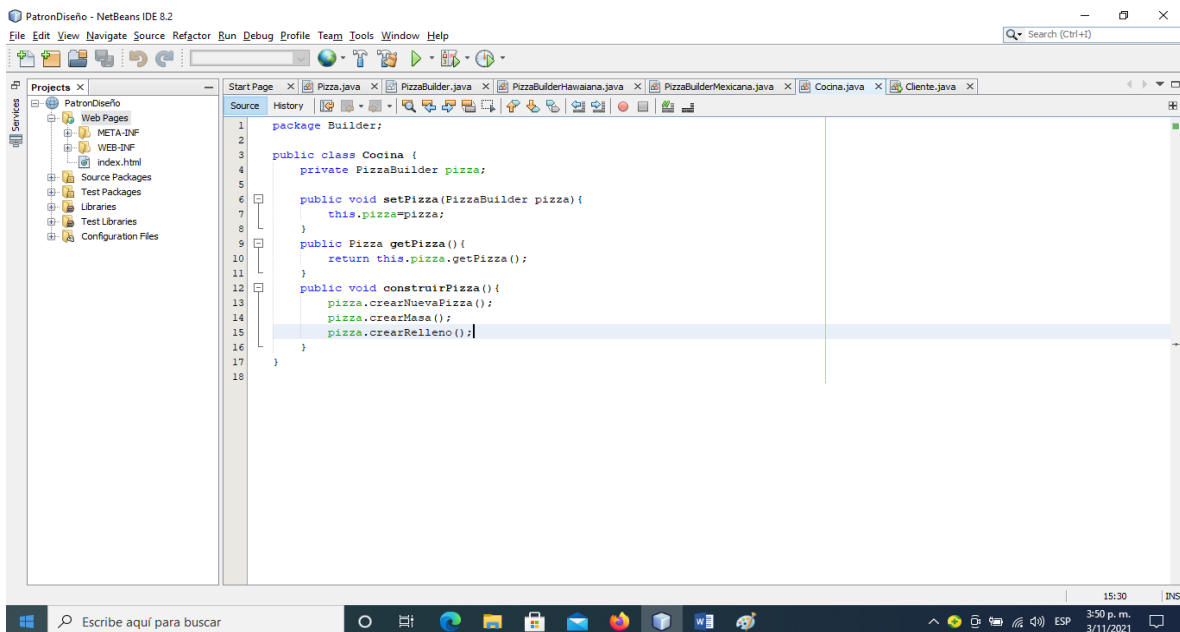
La clase `PizzaBuilderHawaiana` hereda de `PizzaBuilder`, tiene como atributo una pizza sobre la que se hacen las construcciones, métodos para devolver la pizza que se crea y para instanciarla, lo único que varía con la clase `PizzaBuilderMexicana` son los valores predeterminados asignados.



La clase `PizzaBuilderMexicana` hereda de `PizzaBuilder`, tiene como atributo una pizza sobre la que se hacen las construcciones, métodos para devolver la pizza que se crea y para instanciarla, lo único que varía con la clase `PizzaBuilderHawaiana` son los valores predeterminados asignados.

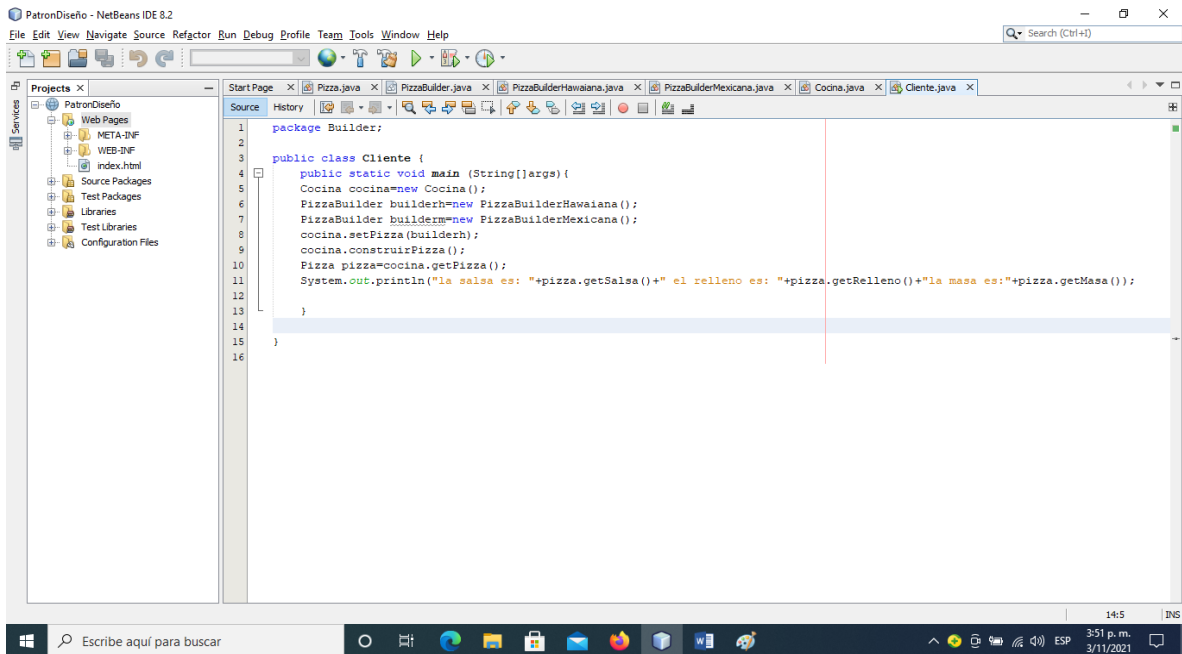


La clase `Cocina` se encarga de gestionar la creación de objetos de la clase (`Pizza`), se compone por un `PizzaBuilder`, tiene el método `construirPizza` que instancia el objeto `Pizza` y asigna los valores que le corresponden.

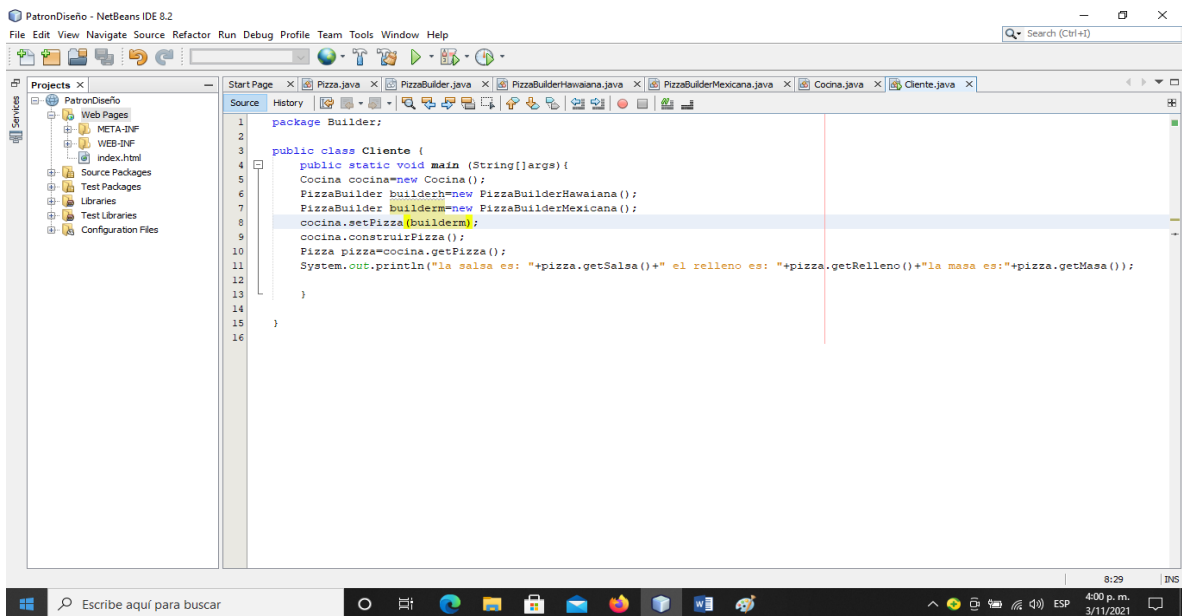


Desde la clase principal (cliente) se llama a construir la Pizza y los métodos para agregar valores a los atributos del producto se ejecuten. En esta clase es donde se hace uso del patrón de diseño Builder.

El producto se crea dependiendo del Pizzabuilder que se asigne. El producto creado se obtiene a través de get.Pizza.



```
1 package Builder;
2
3 public class Cliente {
4     public static void main (String[] args){
5         Cocina cocina=new Cocina();
6         PizzaBuilder builderh=new PizzaBuilderHawaiana();
7         PizzaBuilder builderm=new PizzaBuilderMexicana();
8         cocina.setPizza(builderh);
9         cocina.construirPizza();
10        Pizza pizza=cocina.getPizza();
11        System.out.println("la salsa es: "+pizza.getSalsa()+" el relleno es: "+pizza.getRelleno()+"la masa es:"+pizza.getMasa());
12    }
13 }
14
15
16
```



```
1 package Builder;
2
3 public class Cliente {
4     public static void main (String[] args){
5         Cocina cocina=new Cocina();
6         PizzaBuilder builderh=new PizzaBuilderHawaiana();
7         PizzaBuilder builderm=new PizzaBuilderMexicana();
8         cocina.setPizza(builderm);
9         cocina.construirPizza();
10        Pizza pizza=cocina.getPizza();
11        System.out.println("la salsa es: "+pizza.getSalsa()+" el relleno es: "+pizza.getRelleno()+"la masa es:"+pizza.getMasa());
12    }
13 }
14
15
16
```