Implementación del Patrón Builder en el Sistema de Citas Médicas

1. INTRODUCCIÓN

El patrón Builder es un patrón de diseño creacional que permite construir objetos complejos paso a paso. Este patrón es especialmente útil cuando un objeto tiene muchos atributos, algunos de los cuales son opcionales o requieren una configuración específica. En el contexto de un sistema de citas médicas, el patrón Builder se utiliza para crear objetos Cita de manera flexible y legible.

2. ¿QUÉ ES EL PATRÓN BUILDER?

El patrón Builder separa la construcción de un objeto de su representación, lo que permite que el mismo proceso de construcción pueda crear diferentes representaciones. En lugar de usar constructores con muchos parámetros o métodos set dispersos, el Builder proporciona una interfaz fluida para configurar el objeto paso a paso.

3. PARTICIPANTES DEL PATRÓN BUILDER

En la implementación del patrón Builder para nuestro sistema de citas medicas los participantes son:

- **Producto (**Cita**)**:
  - Representa el objeto complejo que se está construyendo.
  - Contiene atributos como id, idPaciente, idMedico, fecha, hora, consultorio, tipo, y disponible.
  -
- **Builder (**CitaBuilder**)**:
  - Define los pasos para construir el producto.
  - Proporciona métodos para configurar cada atributo de la Cita.
  -
- **ConcreteBuilder (**CitaBuilder**)**:
  - Implementa los métodos definidos en el Builder.
  - Mantiene una instancia del producto que se está construyendo.
  -
- **Cliente**:
  - Utiliza el Builder (y opcionalmente el Director) para construir el producto.
  - En este caso, el cliente es el código que crea una Cita.

4. ¿CÓMO FUNCIONA?

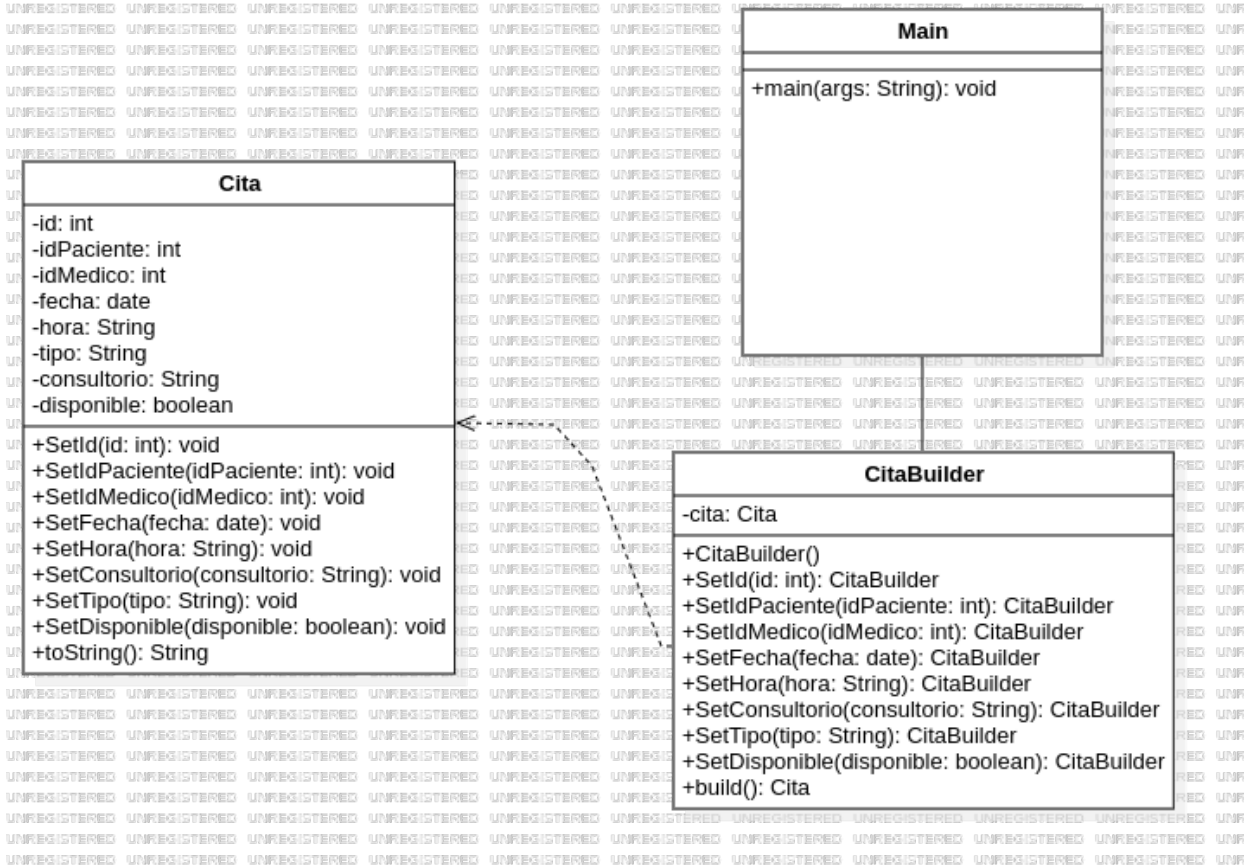
En nuestro proyecto de gestion de citas el patrón builder funciona así:

1. El Cliente crea una instancia del CitaBuilder.
2. El Cliente llama a los métodos del CitaBuilder para configurar los atributos de la Cita y asi poder una cita personalizada.
3. Despues el Cliente llama al método build() para obtener el objeto Cita completamente construido.

5. IMPLEMENTACIÓN EN EL PROYECTO

En el sistema de gestion citas médicas, el patrón Builder se implementará para crear objetos Cita de manera flexible. Esto es especialmente útil porque una Cita tiene muchos atributos, algunos de los cuales pueden ser opcionales.

6. DIAGRAMA UML



7. CÓDIGO DE IMPLEMENTACIÓN

- **Clase Cita**

```
public class Cita {
    private int id;
    private int idPaciente;
    private int idMedico;
```

```
private String fecha;
private String hora;
private String consultorio;
private String tipo;
private boolean disponible;

// Constructor privado para evitar instanciación directa
private Cita() {}

// Getters y Setters
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public int getIdPaciente() {
    return idPaciente;
}

public void setIdPaciente(int idPaciente) {
    this.idPaciente = idPaciente;
}

public int getIdMedico() {
    return idMedico;
}

public void setIdMedico(int idMedico) {
    this.idMedico = idMedico;
}

public String getFecha() {
    return fecha;
}

public void setFecha(String fecha) {
    this.fecha = fecha;
}

public String getHora() {
    return hora;
}

public void setHora(String hora) {
    this.hora = hora;
}

public String getConsultorio() {
    return consultorio;
}

public void setConsultorio(String consultorio) {
    this.consultorio = consultorio;
}

public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
```

```
    }

    public boolean isDisponible() {
        return disponible;
    }

    public void setDisponible(boolean disponible) {
        this.disponible = disponible;
    }

    @Override
    public String toString() {
        return "Cita{" +
                "id=" + id +
                ", idPaciente=" + idPaciente +
                ", idMedico=" + idMedico +
                ", fecha='" + fecha + '\'' +
                ", hora='" + hora + '\'' +
                ", consultorio='" + consultorio + '\'' +
                ", tipo='" + tipo + '\'' +
                ", disponible=" + disponible +
                '}';
    }
}
```

- **Clase CitaBuilder**

```
public class CitaBuilder {
    private Cita cita;

    public CitaBuilder() {
        this.cita = new Cita();
    }

    public CitaBuilder setId(int id) {
        cita.setId(id);
        return this;
    }

    public CitaBuilder setIdPaciente(int idPaciente) {
        cita.setIdPaciente(idPaciente);
        return this;
    }

    public CitaBuilder setIdMedico(int idMedico) {
        cita.setIdMedico(idMedico);
        return this;
    }

    public CitaBuilder setFecha(String fecha) {
        cita.setFecha(fecha);
        return this;
    }

    public CitaBuilder setHora(String hora) {
        cita.setHora(hora);
        return this;
    }

    public CitaBuilder setConsultorio(String consultorio) {
        cita.setConsultorio(consultorio);
        return this;
    }
```

```
    public CitaBuilder setTipo(String tipo) {
        cita.setTipo(tipo);
        return this;
    }

    public CitaBuilder setDisponible(boolean disponible) {
        cita.setDisponible(disponible);
        return this;
    }

    public Cita build() {
        return cita;
    }
}
```

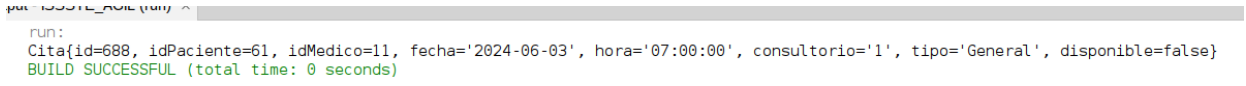
- **Cliente**

```
private void AceptarMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:\
        hora = Hora.getSelectedItem().toString();
        int idCita=Formato_Citas.obtenerIdCita(conn, idMedico, fecha, hora);
        LocalDateTime now = LocalDateTime.now();

        // Definir un formato personalizado para la fecha y hora
        DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd");
        DateTimeFormatter timeFormatter =
DateTimeFormatter.ofPattern("HH:mm:ss");

        // Formatear la fecha y la hora como cadenas de texto
        String fechaActual = now.format(dateFormatter);
        String horaActual = now.format(timeFormatter);
        Cita cita = new CitaBuilder()
                        .setId(idCita)
                        .setIdPaciente(user.getId())
                        .setIdMedico(idMedico)
                        .setFecha(fecha)
                        .setHora(hora)
                        .setConsultorio(consultorio)
                        .setTipo(tipo1)
                        .setDisponible(false)
                        .build();
        Formato_Citas.cambiarEstadoCita(conn, idCita,
user.getId(),user.getNombre(),fechaActual,horaActual,nomDoc,fecha,hora,consultor
io,tipo1,idMedico,cita);
        Inicio_Usuario ho = new Inicio_Usuario(user,conn);
        this.setVisible(false);
        ho.setVisible(true);
    }
```

8. SÁLIDA DEL CÓDIGO



9. CONCLUSIÓN

El patrón Builder implementado en nuestro proyecto puede ser bastante útil, ya que podemos aplicarlo en nuestra creación de citas. Aprendimos a que este patrón al ser aplicado puede ayudarnos a la creación eficiente de objetos como

las citas, que involucran múltiples atributos, algunos de los cuales son opcionales o requieren una configuración específica.