Implementación del patrón Prototype en el sistema de Citas Medicas

1.Introducción

El patrón de diseño prototipo tiene como finalidad crear nuevos objetos clonando una instancia creada previamente. Este patrón especifica la clase de objetos a crear mediante la clonación de un prototipo que es una instancia ya creada.

2.¿Qué es el patrón Prototype?

Prototype es un patrón de diseño creacional que nos permite copiar objetos existentes sin que el código dependa de sus clases.

Basa su funcionalidad en la clonación de objetos, estos nuevos objetos son creados mediante un pool de prototipos elaborados previamente y almacenados. Este patrón es especialmente útil cuando necesitamos crear objetos basados en otros ya existentes o cuando se necesita la creación de estructuras de objetos muy grandes, este patrón nos ayuda también a ocultar la estrategia utilizada para clonar un objeto.

3.Participantes:

- Client: Componente que interactúa con los prototipos.
- IPrototype: Este componente por lo general es una interface y define los atributos mínimos de un prototipo, esta interface debe contar por lo menos con alguno de los dos tipos de clonación. Clonación superficial (clone) o clonación en profundidad (deepClone) los cuales explicaremos más adelante.
- ConcretePrototype: Implementaciones concretas de IPrototype los cuales podrán ser clonados.
- PrototypeFactory: Componente que utilizaremos para mantener el cache de los prototipos existentes, así como para crear clonaciones de los mismos.

4.Problema:

Tenemos dos objetos y queremos crear una copia exacta de ellos. En primer lugar, debemos crear un nuevo objeto de la misma clase. Después debes recorrer todos los campos del objeto original y copiar sus valores en el nuevo objeto.

No todos los objetos se pueden copiar de este modo, porque algunos de los campos del objeto pueden ser privados e invisibles desde fuera del propio objeto.

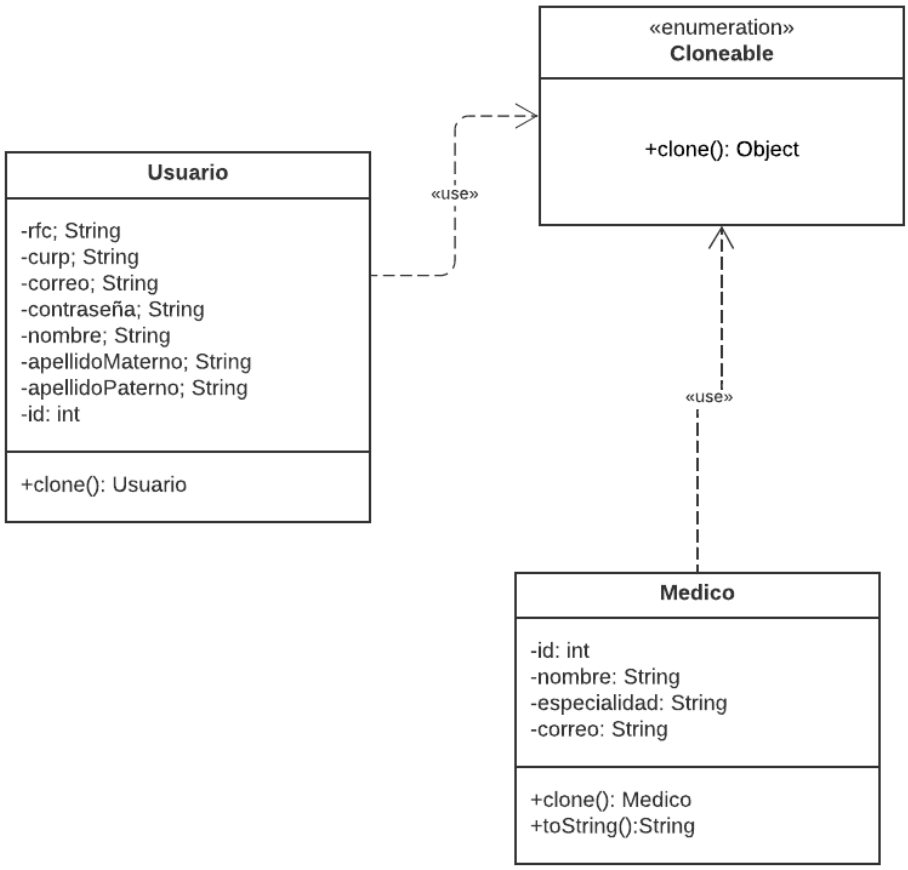
5.Solución:

El patrón declara una interfaz común (Cloneable) para todos los objetos que soportan la clonación. Esta interfaz nos permite clonar un objeto sin acoplar el código a la clase de ese objeto. Normalmente, dicha interfaz contiene un único método clonar.

La implementación del método clonar es muy parecida en todas las clases. El método crea un objeto a partir de la clase actual y lleva todos los valores de campo del viejo objeto, al nuevo. Se puede incluso copiar campos privados, porque la mayoría de los lenguajes de programación permite a los objetos acceder a campos privados de otros objetos que pertenecen a la misma clase.

Al modificar estas clase se modificara la Vista_Registro_Usuario para poder guardar los clones en la base de datos.

6.Diagrama UML



7.Código de implementación

- Clase Medico:

```
public class Medico implements Cloneable {
    private int id;
    private String nombre;
    private String especialidad;
    private String correo;

    public Medico(int id, String nombre, String especialidad, String correo) {
        this.id = id;
        this.nombre = nombre;
        this.especialidad = especialidad;
        this.correo = correo;
    }

    public int getId() {
        return id;
    }

    public String getCorreo() {
        return correo;
    }

    public void setCorreo(String correo) {
        this.correo = correo;
    }

    public String getNombre() {
        return nombre;
    }

    public String getEspecialidad() {
        return especialidad;
    }

    // Implementación del patrón Prototype
    @Override
    public Medico clone() {
        try {
            return (Medico) super.clone(); // Clonación superficial
        } catch (CloneNotSupportedException e) {
            throw new RuntimeException("Error al clonar Medico", e);
        }
    }

    @Override
```

```
    public String toString() {
        return nombre + " (" + especialidad + ")";
    }
}
```

- Clase Usuario

```
public class Usuario implements Cloneable {
    private String rfc, curp, correo, contraseña, nombre, apellidoMaterno, apellidoPaterno, tipo;
    private int id;

    public Usuario(String rfc, String curp, String correo, String contraseña, String nombre, String
apellidoMaterno, String apellidoPaterno) {
        this.rfc = rfc;
        this.curp = curp;
        this.correo = correo;
        this.contraseña = contraseña;
        this.nombre = nombre;
        this.apellidoMaterno = apellidoMaterno;
        this.apellidoPaterno = apellidoPaterno;
    }

    public int getId() {
        return id;
    }

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

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

    public String getTipo() {
        return tipo;
    }

    public String getRfc() {
        return rfc;
    }

    public String getCurp() {
        return curp;
```

```
}

public String getCorreo() {
   return correo;
}

public String getContraseña() {
   return contraseña;
}

public String getNombre() {
   return nombre;
}

public String getApellidoMaterno() {
   return apellidoMaterno;
}

public String getApellidoPaterno() {
   return apellidoPaterno;
}

public void setRfc(String rfc) {
   this.rfc = rfc;
}

public void setCurp(String curp) {
   this.curp = curp;
}

public void setCorreo(String correo) {
   this.correo = correo;
}

public void setContraseña(String contraseña) {
   this.contraseña = contraseña;
}

public void setNombre(String nombre) {
   this.nombre = nombre;
}

public void setApellidoMaterno(String apellidoMaterno) {
   this.apellidoMaterno = apellidoMaterno;
```

```
    }

    public void setApellidoPaterno(String apellidoPaterno) {
        this.apellidoPaterno = apellidoPaterno;
    }

    // Implementación del patrón Prototype
    @Override
    public Usuario clone() {
        try {
        Usuario clon = (Usuario) super.clone();
        clon.setId(0); // Para que la base de datos asigne un nuevo ID

        // Modificar correo para evitar conflictos en la base de datos
        clon.setCorreo(this.correo.replace("@", "+clon@"));

        // Modificar RFC y CURP (opcional)
        clon.setRfc(this.rfc + "C");
        clon.setCurp(this.curp + "C");

        return clon;
    } catch (CloneNotSupportedException e) {
        throw new RuntimeException("Error al clonar Usuario", e);
    }
    }
}
```