Carolina Chinchilla

Taller 5

Patrón
Prototype

¿Para qué se usa este patrón?
El patrón Portotype es un patrón de diseño creacional que delega el proceso de clonación de objetos a los objetos reales que se están clonando, cada objeto que soporte la clonación se llamará prototipo. Este patrón se usará cuando las clases a instanciar manejan un tiempo de ejecución, y cuando un sistema deba de ser independiente de cómo se crean y componen sus productos. Este permitirá utilizar los campos privados de los objetos

¿Cómo se usa el patrón en el proyecto?
En primer lugar, para usar este patrón se debe declarar una interfaz en común para todos los objetos que soporten la clonación, donde la interfaz suele tener un solo método de clonación, luego se crea un nuevo objeto de la clase actual y se copiará todos los valores del antiguo objeto sobre el nuevo objeto.

Clase Prototype

```java
package com.sudoku;

/**
 * Author Kamil Seweryn
 */

public class Prototype<T> implements Cloneable {
    @Override
    public T clone() throws CloneNotSupportedException {
        return (T)super.clone();
    }
}
```

Clase SudokuBoard

```java
package com.sudoku;

import java.util.ArrayList;

/**
 * Author Kamil Seweryn
 */

public class SudokuBoard extends Prototype {
    private List<SudokuRow> board = new ArrayList<>();
    public List<Backtrack> backtrack = new ArrayList<>();

    public final static int MIN_INDEX = 1;
    public final static int MAX_INDEX = 9;

    public SudokuBoard() {
        createBoard();
    }

    public List<SudokuRow> getBoard() {
        return board;
    }

    public SudokuRow getRow(int row) {
        return board.get(row);
    }

    public void addRow(SudokuRow sudokuRow) {
        board.add(sudokuRow);
    }

    public List<SudokuRow> createBoard() {
```

Responsabilidades:
Clonar la clase SudokuBoard en el método deepCopy de la misma clase.

Poder utilizar la interfaz clone para clonar cualquier objeto posible.

Rol del patrón:
Emplear una interfaz que dado el empleo de su único método clone () se puedan clonar objetos sin necesitar acceder a sus campos privados y permitiendo un sistema independiente.

Participantes:
Hay dos clases que permiten reflejar el patrón. En primer lugar, la clase Prototype, donde está la interfaz con el método de clonación. En segundo lugar, SudokuBoard que es una clase que en su método deepCopy() utiliza la clonación para duplicar información y agregarlas en listas que permitirán unificar la mayor parte del proyecto.

URL
https://github.com/kamisc/Sudoku-Solver.git

¿Por qué tiene sentido haber utilizado el patrón en ese punto del proyecto?
Utilizar el patrón en esa parte del código se hizo dado que la clase SudokuBoard termina integrando la mayoría de las clases, por lo que serviría usarlo en ese momento dado que solo se requeriría utilizar el método de clonado en sólo una parte.

¿Qué ventajas tiene?
Al clonar cada parte del sudoku y agregarlas en una misma zona permite que no sea necesario tener que hallar una forma de sobrescribir en algún lado cada objeto y evita tener que pasar por complicaciones al no tener que acceder a los elementos privados.

¿Qué desventajas tiene haber utilizado el patrón en ese punto del proyecto?
Más adelante si se requiere modificar alguna parte en específico de las partes clonadas será complicado dado que como se mencionó anteriormente SudokuBoard maneja una integración de las demás clases.

¿De qué otras formas se le ocurre que se podrían haber solucionado, en este caso particular, los problemas que resuelve el patrón?
Se pudo simplemente haber sobrescrito y poner un método para acceder a los campos privados de los objetos dado que son aproximadamente 3 clases de las cuales se requiere una clonación.