Wydział Elektryczny

POLITECHNIKA WARSZAWSKA

Języki i metody programowania 2

`lava`: Podział na moduły i wzorce projektowe

Oliwia Pawelec, Jakub Żebrowski

prowadzący zajęcia:

Dr inż. Radosław Roszczyk

27.05.2024

Wzorce projektowe

W projekcie wykorzystaliśmy różne wzorce projektowe i między nimi:

- Prosta fabryka (ang. *Simple factory*)
- Singleton
- Kompozyt (ang. *Composit*)
- Adapter

Prosta fabryka

Wykorzystana do zarządzania wczytywaniem z plików (istnieje fabryka, o analogicznym mechanizmie, do zapisu danych z labiryntu do plików). Wykorzystuje interfejs `Loader`, zawierający wspólne, wymagane metody dla wszystkich obiektów zajmujących się wczytywaniem z pliku.

```
public interface Loader {
    public static enum LoadResult {
        SUCCESS,
        BAD_DIMS,
        BAD_CHARS,
        INVALID_STRUCT
    };
    public LoadResult Load(File in);
}
```

Obiekty zajmujące się wczytem danych implementują dany interfejs.

```
public class BinaryLoader implements Loader {
    @Override
    public LoadResult Load(File in) {
    ...
```

```
public class TextLoader implements Loader {
    @Override
    public Loader.LoadResult Load(File in) {
    ...
```

Obiekt fabryki tworzy odbiekt, zależnie od tego jaki typ wczytu należy obsłużyć.

```
public class LoaderFactory {
    public enum LoadType {
        BINARY
        TEXT
    }
```

```
    public static Loader CreateLoader(LoadType type) {
        switch (type) {
            default:
            case TEXT: return new TextLoader();
            case BINARY: return new BinaryLoader();
        }
    }
}
```

Singleton

Labirynt jest traktowany jako singleton. Podczas działania programu wymagana jest tylko jedna instancja labiryntu - tylko jeden labirynt na raz jest obsługiwany. Wymienianie są tylko jego parametry podczas załadowywania/usuwania danych.

```
public class Maze {
    private static Maze mazeObject = null;

    private Maze() {
        isEmpty = true;
    }

    public static Maze getInstance() {
        if ( mazeObject == null)
            mazeObject = new Maze();
        return mazeObject;
    }

    public void NewData(int[][] maze, int width, int height, Point
         start, Point end) {
        this.maze = maze;
        this.width = width;
        this.height = height;
        this.start = start;
        this.end = end;

        isEmpty = false;
    }

    public void Reset() {
        mazeObject = null;
    }
```

Kompozyt

Wykorzystany zaimplementowany kompozyt jest elementem komponentów biblioteki `Swing`. Jest to JMenuBar, do którego można dodać komponenty typu

`JMenu`, do których można dodać komponenty typu `JMenuItem`. Każdy z nich może mieć zdefiniowane przez programistę działanie. Każdy komponent można traktować jako oddzielny obiekt. W ten sposób, można zdefiniować funkcje dla poszczególnej opcji menu (np. `Save As > binary` do otwierania i załadowania danych z wyznaczonego pliku binarnego).

```
void menuBarSetup() {
        bar = new JMenuBar();

        openItem = new JMenu("Open From");
        openTextItem = new JMenuItem("text");
        openBinaryItem = new JMenuItem("binary");

        openItem.add(openTextItem);
        openItem.add(openBinaryItem);
        bar.add(openItem);

        saveItem = new JMenu("Save As");
        saveTextItem = new JMenuItem("text");
        saveBinaryItem = new JMenuItem("binary");
        saveImageItem = new JMenuItem("image");

        saveItem.add(saveTextItem);
        saveItem.add(saveBinaryItem);
        saveItem.add(saveImageItem);
        bar.add(saveItem);

        openBinaryItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                openFilePattern(LoaderFactory.LoadType.BINARY);
            }
        });
```

Adapter

Adapterem w programie jest `MazePanel` (dziedziczący cechy po Swingowym komponencie `JPanel`), który dostosowuje surowe dane z pozyskanego z plików labiryntu do widoku dla klienta. Metoda adaptująca rysuje labirynt na panelu głównym.

```
public class MazePanel extends JPanel {
    BufferedImage MazeImage;
    public static int cellSize = 10;

    private void drawMaze(Graphics g) {
        for (int y = 0; y < Maze.getInstance().getHeight(); ++y) {
            for (int x = 0; x < Maze.getInstance().getWidth(); ++x) {
                int cell = Maze.getInstance().getCell(x, y);
```

```
            ...

            g.fillRect(x * cellSize, y * cellSize, cellSize,
                cellSize);
            g.setColor(Color.GRAY);
            g.drawRect(x * cellSize, y * cellSize, cellSize,
                cellSize);
        }
    }
}
```

Podział na moduły

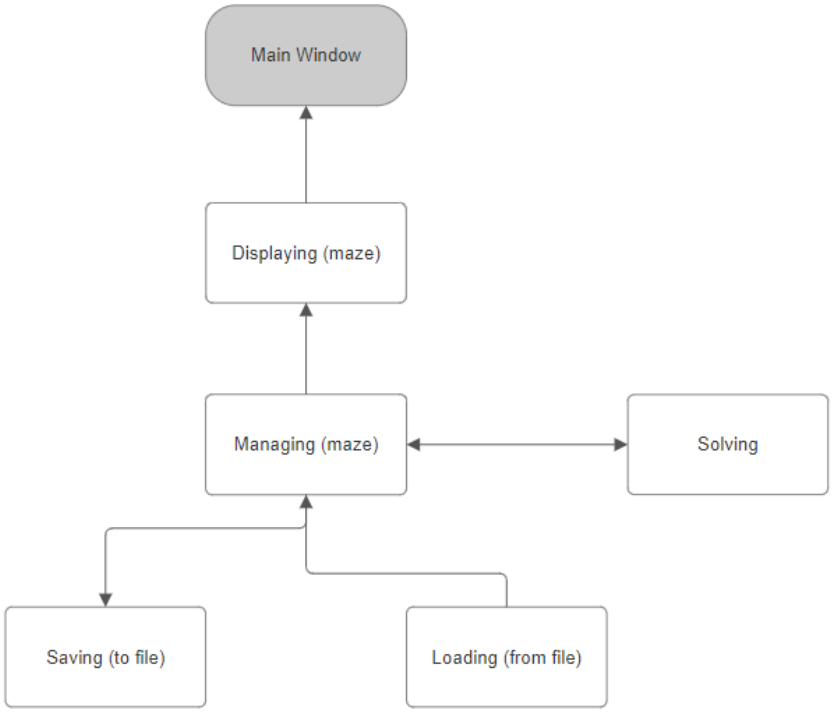

Figure 1: Wizualizacja podziału na moduły - kierunek strzałek wyznacza kierunek przepływu przekazywanych najważniejszych danych

Najważniejsze moduły i ich funkcje:

- Głównego okna

- Wyświetlania (labiryntu)
- Zarządzania (labiryntem)
- Rozwiązania
- Wczytu
- Zapisu

Głównego okna

Obejmuje wszystkie metody i obiekty działające wokół głównego okna: obsługa UI (przycisków, paneli, pól) i zarządzanie interakcjami z użytkownikiem - wywoływanie kolejnych modułów, jeśli użytkownik wywołał jakąś funkcjonalność, np. wybrał plik z labiryntem do załadowania.

Wyświetlania (labiryntu)

Obejmuje wszystkie metody i obiekty zajmujące się renderowaniem panelu wyświetlającego labirynt. Jego funkcją jest również zapis labiryntu, którego graficzne odwzorcowanie znajduje się w polach jednego z obiektów modułu, do pliku graficznego.

Zarządzania (labiryntem)

Moduł ten przechowuje wszystkie dane na temat labiryntu (struktura, wymiary, punkty początku i końca). Również zajmuje się załadowywaniem nowych danych do labiryntu oraz usuwaniem tych, które znajdują się już w programie.

Rozwiązania (labiryntu)

Obejmuje wszystkie metody zajmujące się szukaniem rozwiązania: szukanie najkrótszej ścieżki, informowanie o istnieniu/nieistnieniu ścieżki, nanoszenie na labirynt znalezionej najkrótszej ścieżki, jeśli takowa została znaleziona.

Wczytu

Jego przykładowe zaimplementowane elementy można znaleźć w sekcji Prosta fabryka.
Zajmuje się wczytywaniem labiryntu z plików o formatowaniu tekstowym lub binarnym. Dostarcza danych do modułu zarządzania labiryntem.

Zapisu

Zajmuje się zapisem głównych danych z labiryntu do plików o formatowaniu tekstowym, binarnym