МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний Технічний Університет України
«Київський Політехнічний Інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №4
з дисципліни «Інженерія програмного забезпечення»
на тему: «Структурні шаблони проектування. Шаблони Flyweight, Adapter, Bridge, Facade»

Виконав:
студент 2-го курсу ФІОТ
групи ІВ-71
Мазан Я.В.
Номер залікової книжки: 7109
Варіант: 3

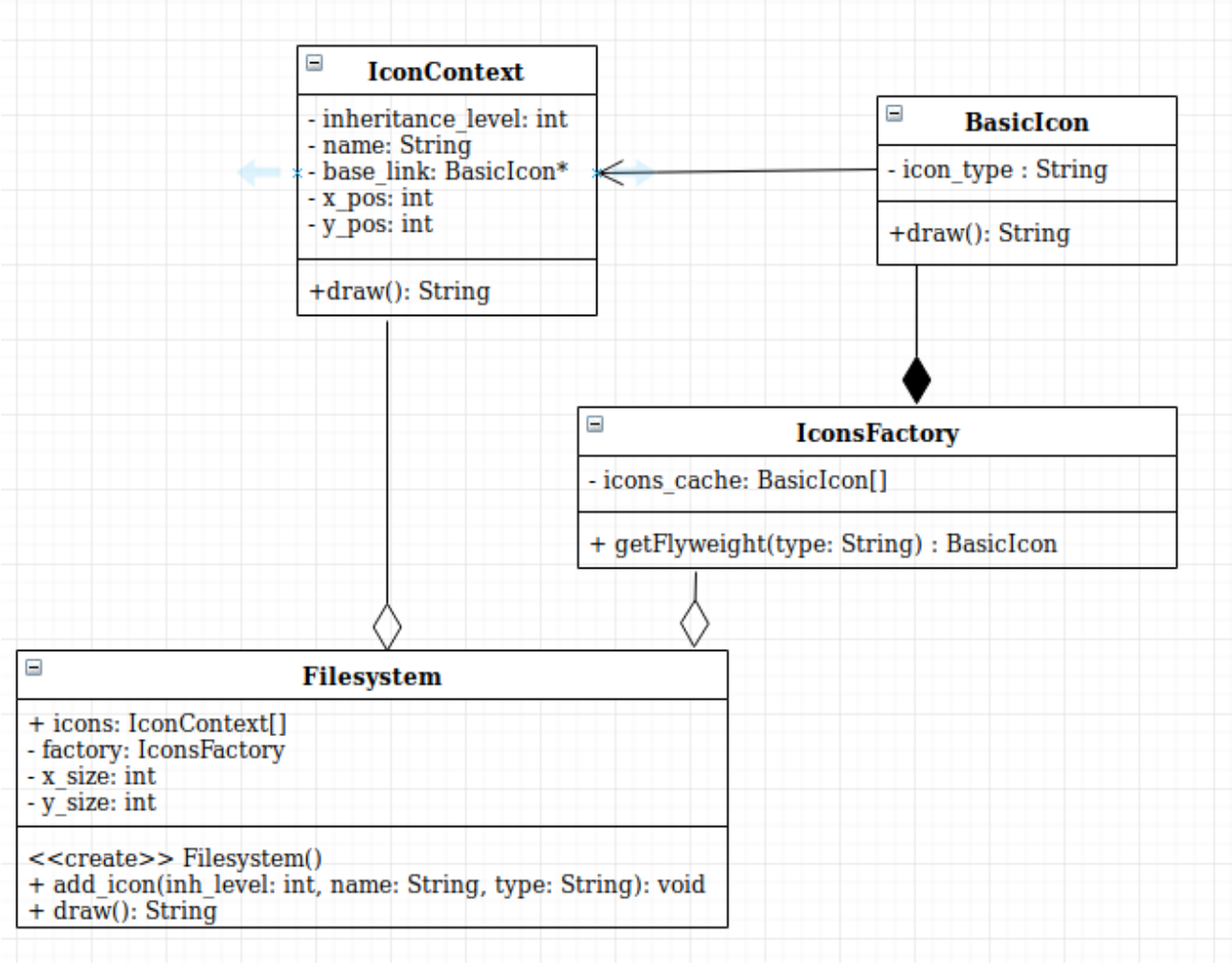
Перевірив:
доцент кафедри ОТ
Антонюк А.І.

Київ – 2018

Завдання

Варіант – 3
Визначити специфікації класів, які подають об’єкти-іконки для зображення елементів файлової системи при побудові графічного інтерфейсу користувача (GUI) – примітиви (файли) та їх композиції (директорії). Забезпечити ефективне використання пам’яті при роботі з великою кількістю графічних об’єктів. Реалізувати метод рисування графічного об’єкту.

Діаграма класів



Код програми

TestMain.java:

```
package com.lab111.labwork4;
/**
 * Template first class.
 * @author Yan Mazan
 *
 */
public final class TestMain {
  /**
   * Constructor.
   *
```

```
   */
  private TestMain() {
    super();
  }

  /**
   * Invokes at application startup.
   * @param args Parameters from command line
   */
  public static void main(final String[] args) {
      FilesystemImage filesystem = new FilesystemImage(1920,1080);
      filesystem.add_icon(0,"Directory", "Base", 0,0);
      filesystem.add_icon(1, "Directory", "Yan", 1,0);
      filesystem.add_icon(1, "File","load.exe",1,1);
      filesystem.add_icon(2,"File","Lab4_main.java",2,0);
      filesystem.add_icon(2,"Directory", "Saved pictures",2,1);
      System.out.println(filesystem.draw());
  }
}
```

BasicIcon.java:

```
package com.lab111.labwork4;
/**
 * Class of non-dynamic values of an icon (only tho types of icons exist: Directory,
File
 */
public class BasicIcon {
    private String icon_type;
    /**
     * Constructor of the class
     * @param type - type of an icon (File or Directory
     */
    public BasicIcon(String type) {
        icon_type = type;
    }
    /**
     * "Draws" an icon
     * @return - type of an icon
     */
    public String draw() {
        return "type: " + icon_type;
    }
}
```

IconContext.java:

```
package com.lab111.labwork4;
public class IconContext {
    private int inheritance_level;
    private String obj_name;
    private BasicIcon base_link;
    private int x_pos;
    private int y_pos;
    /**
     * A constructor of dynamic context attributes of an icon
     * @param type - type of a new icon (static: does it belong to File or to Directory)
     * @param name - name of our file/directory
     * @param inh_lev - inheritance level of file/directory inside a filesystem
     * @param x - x position of an icon on an image
     * @param y - y position of an icon on an image
     */
    public IconContext(BasicIcon type, String name, int inh_lev, int x, int y) {
        base_link = type;
        obj_name = name;
        inheritance_level = inh_lev;
        x_pos = x;
```

```
        y_pos = y;
    }
    /**
     * "Draws" an icon
     * @return - type, name, inheritance level, position on an image of an icon
     */
    public String draw() {
        return "Icon represents an object with " + base_link.draw() + " name: " +
obj_name + " inheritance level: " + inheritance_level + " coordinates on image: [" +
x_pos + ", " + y_pos + "]";
    }
}
```

IconsFactory.java:

```
package com.lab111.labwork4;
import java.util.HashMap;
/**
 * A constructor class of basic icons to make them do not repeat
 */
public class IconsFactory {
    private HashMap<String,BasicIcon> cache;
    /**
     * An empty constructor of the class
     */
    public IconsFactory() {
        cache = new HashMap<>();
    }
    /**
     * A method that returns necessary basic icon type for construction of context icon
     * @param type - type of icon that we must receive
     * @return - required BasicIcon that corresponds to input type
     */
    public BasicIcon getFlyweight(String type) {
        if (!cache.containsKey(type))
            cache.put(type,new BasicIcon(type));
        return cache.get(type);
    }
}
```

FilesystemImage.java:

```
package com.lab111.labwork4;
import java.util.ArrayList;
/**
 * Class of basic image onto which the image is drawn
 */
public class FilesystemImage {
    public ArrayList<IconContext> icons;
    private IconsFactory factory;
    private int x_size;
    private int y_size;
    /**
     *
     * @param x - x resolution of an image
     * @param y - y resolution of an image
     */
    public FilesystemImage(int x, int y) {
        factory = new IconsFactory();
        icons = new ArrayList<>();
        x_size = x;
        y_size = y;
    }
    /**
     * Public method that adds an icon to the image
     * @param inh_level - inheritance level of the object an icon represents
```

```
     * @param type - object's type (File or Directory)
     * @param name - name of our file/directory
     * @param x - x position of an icon on an image
     * @param y - y position of an icon on an image
     */
    public void add_icon(int inh_level, String type, String name, int x, int y) {
        BasicIcon icon_type = factory.getFlyweight(type);
        IconContext icon = new IconContext(icon_type, name, inh_level, x, y);
        icons.add(icon);
    }
    /**
     * Final method that "draws" our filesystem
     * @return - description of icons added
     */
    public String draw() {
        StringBuilder res = new StringBuilder("Image of filesystem with resolution " +
x_size + "x" + y_size + "px with included icons:\n");
        for (IconContext i : icons) {
            res.append(i.draw() + "\n");
        }
        return res.toString();
    }
}
```

Висновок

В даній лабораторній роботі мені було необхідно розробити класи, які подають іконки для зображення елементів файлової системи в побудові графічного інтерфейсу. Існує два типи іконок: для файлів (File) та папок (Directory). Для оптимального використання пам’яті при роботі з великою кількістю графічних об’єктів мені довелось застосувати шаблон Flyweight. Під час виконання роботи проблем не виникло.