Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики та обчислювальної техніки

Звіт з лабораторної роботи 4
з дисципліни
«Технології розробки програмного забезпечення - 1. Основи розробки програмного забезпечення на платформі Java»

Виконав:
студент 3-го курсу
групи ІП-52, ФІОТ
Набоков Едруард
Максимович
5212

Київ 2017

Завдання

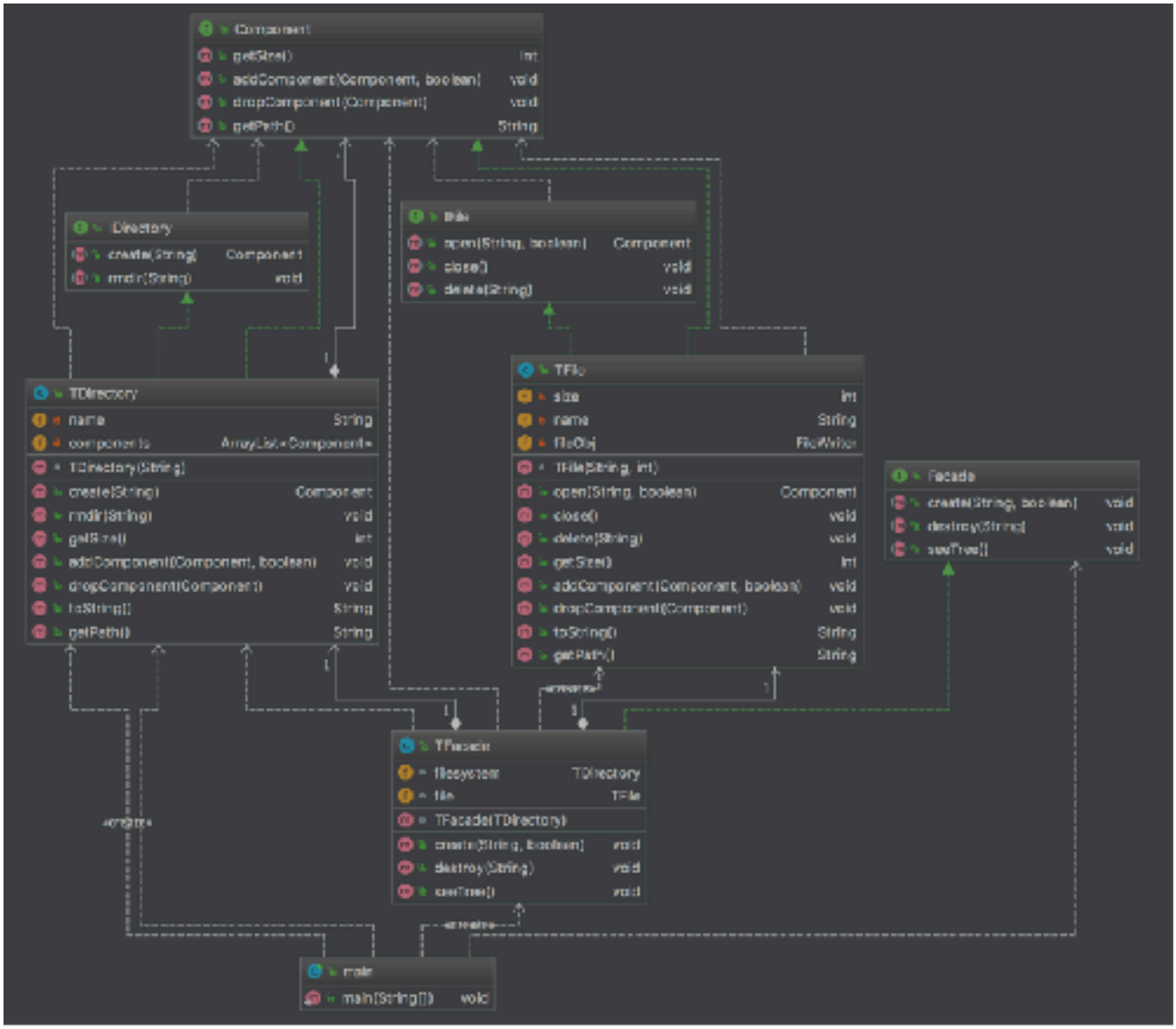
1. Закріпити призначення шаблонів проектування ПЗ, їх класифікацію. Знати назву і коротку характеристику кожного з шаблонів, що відносяться до певного класу.
2. Повторити структурні шаблони проектування ПЗ. Знати загальну характеристику структурних шаблонів та призначення кожного з них.
3. Детально вивчити структурні шаблони проектування Flyweight, Adapter, Bridge, Facade. Для кожного з них:
   - вивчити Шаблон, його призначення, альтернативні назви, мотивацію, випадки, коли його застосування є доцільним, та результати такого застосування;
   - знати особливості реалізації Шаблону, споріднені шаблони, відомі випадки його застосування в програмних додатках;
   - вільно володіти структурою Шаблону, призначенням його класів та відносинами між ними;
   - вміти розпізнавати Шаблон в UML діаграмі класів та будувати сирцеві коди Java-класів, що реалізують шаблон.
4. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork4. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.3). У класах, що розробляються, повністю реалізувати методи, пов'язані з функціюванням Шаблону. Методи, що реалізують бізнес-логіку, закрити заглушками з виводом на консоль інформації про викликаний метод та його аргументи.
5. За допомогою автоматизованих засобів виконати повне документування розроблених класів (також методів і полів), при цьому документація має в достатній мірі висвітлювати роль певного класу в загальній структурі Шаблону та особливості конкретної реалізації.

Варіант 9. 5212 mod 11 = 9

9.Визначити специфікації класів, які подають об'єкти для маніпулювання елементами файлової системи – файлами та директоріями. Інтерфейс файлу містить методи open(String path, boolean createIfNotExist), close() та delete(String path) для відкриття, закриття та видалення файлу (при createIfNotExist=true файл буде створений, якщо він не існує або обрізаний до нульової довжини, якщо існує). Інтерфейс директорії містить методи create(String path), та rmdir(String path) для створення та видалення директорії. Задати підсистему з 3-ох файлів та 2-х директорій.

Забезпечити можливість створення та видалення такої підсистеми через методи create(), destroy() та зміни структури підсистеми без впливу на її користувача.

ДІАГРАМА КЛАСІВ



ПРОГРАМНИЙ КОД

Component.java

```
package com.solutions.labwork4;

/**
 * Interface Component.
 * It's common access to handle
 * multiple objects as one object.
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 23.10.17
 */

public interface Component {

    /**
     * Calculate size or return given size
     * @return size (int)
     */
    int getSize();

    /**
     * Add component to the list
     * @param obj (Component)
     */
    void addComponent(Component obj, boolean createIfNotExists);

    /**
     * Drop component from the list
     * @param obj (Component)
     */
    void dropComponent(Component obj);

    /**
     * Get Path of folder or a file
     */
    String getPath();

}
```

Facade.java

```
package com.solutions.labwork4;

/**
 * Interface Facade.
 * Simplified interface for usage.
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 9.11.17
 */
```

```
public interface Facade {
    void create(String path, boolean createIfExists);
    void destroy(String path);
    void seeTree();
}
```

Directory.java

```
package com.solutions.labwork4;

//import java.io.File;

public interface IDirectory {

    Component create(String path);

    void rmdir(String path);
}
```

File.java

```
package com.solutions.labwork4;

/**
 * Interface File.
 * It's common access to handle
 * files.
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 9.11.17
 */
public interface IFile {
    Component open(String path, boolean createIfNotExists);
    void close();
    void delete(String path);
}
```

Director.java

```
package com.solutions.labwork4;

import java.io.File;
import java.util.ArrayList;

/**
 * Class Directory.
 * Implements common methods
 * for handling.
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 9.11.17
 */
public class TDirectory implements IDirectory, Component {
```

```
private String name = "default";
private ArrayList<Component> components = new ArrayList<>();

TDirectory(String name){
    this.name = name;
}

@Override
public Component create(String path){
    File dir = new File(path);

    String[] entries = path.split("/");
    String name = entries[entries.length - 1];

    if(!dir.exists()){
        dir.mkdir();
        System.out.println("Folder "+name+" was created");
    }

    return new TDirectory(name);
}

@Override
public void rmdir(String path){
    File dir = new File(path);
    String parent = dir.getParent();

    File currentFile = new File(parent, name);
    currentFile.delete();
}


@Override
public int getSize() {
    int size = 0;
    for(Component component: components) {
        size += component.getSize();
    }

    return size;
}

@Override
public void addComponent(Component obj, boolean createIfNotExists) {
    this.components.add(obj);
}

@Override
public void dropComponent(Component obj) {
    this.components.remove(obj);
}

@Override
public String toString() {
    String content = getPath()+": ";
    if (!components.isEmpty()) {
        for (Component component : components) {
            content += "\n  " + component;
        }
    }
```

```
        else {
            return content + "nothing";
        }

        return content;
    }

    @Override
    public String getPath(){
        return this.name;
    }
}
```

Facade.java

```
package com.solutions.labwork4;

import java.io.File;

/**
 * Class Facade.
 * Implements methods.
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 9.11.17
 */
public class TFacade implements Facade {

    TDirectory filesystem = null;
    TFile file = new TFile("default", 10);

    TFacade(TDirectory filesystem) {
        this.filesystem = filesystem;
    }

    @Override
    public void create(String path, boolean createIfNotExists) {
        String[] entries = path.split("/");

        String namedirCurr = entries[0];

        Component currFolder = this.filesystem.create(namedirCurr);
        Component rootFolder = currFolder;
        Component nextFolder = null;

        if(entries.length > 1) {
            for (int i = 1; i < entries.length; i++) {
                namedirCurr += "/" + entries[i];

                if (!entries[i].contains(".")) {
                    nextFolder = this.filesystem.create(namedirCurr);
                    currFolder.addComponent(nextFolder, createIfNotExists);
                    currFolder = nextFolder;
                } else {
                    this.file = new TFile(entries[i], 10);
                    Component file_ = this.file.open(namedirCurr, createIfNotEx-
ists);
                    currFolder.addComponent(file_, createIfNotExists);
                }
            }
```

```
        }
        else {
            this.file = new TFile(path, 10);
            Component file = this.file.open(path, createIfNotExists);
            currFolder.addComponent(file, createIfNotExists);
        }


        this.filesystem.addComponent(rootFolder, createIfNotExists);
    }

    @Override
    public void destroy(String path) {
        File dir = new File(path);

        String[] entries = path.split("/");
        String name = entries[entries.length - 1];

        String parent = dir.getParent();

        if(dir.isDirectory()) {
            for (File c : dir.listFiles())
                destroy(c.getPath());

            File currentFile = new File(parent, name);
            currentFile.delete();
            this.filesystem.rmdir(path);
        }
        else{
            this.file = new TFile(path, 10);
            this.file.close();
            this.file.delete(path);
        }

        File currentFile = new File(parent, name);
        currentFile.delete();

        System.out.println("Folder "+currentFile.toString()+" was deleted");
    }

    @Override
    public void seeTree(){
        System.out.println("\nTOTAL");
        System.out.println(this.filesystem.toString());
    }
}
```

File.java

```
package com.solutions.labwork4;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

/**
 * Class File.
 * Implements common methods.
 *
 * @author Eduard Nabokov
 * @version 0.1
 * @since 9.11.17
```

```
 */

public class TFile implements IFile, Component{

    private int size;
    private String name = "default";
    private FileWriter fileObj;

    TFile(String name, int size){
        this.size = size;
        this.name = name;
    }

    @Override
    public Component open(String path, boolean createIfNotExists) {
        File file = new File(path);
        if (file.exists()) {
            try {
                this.fileObj = new FileWriter(path);
            } catch (IOException e) {
                e.printStackTrace();
            }
            System.out.println("File exists");
            System.out.println("File was opened");
        }
        else{
            if (createIfNotExists) {
                try {
                    this.fileObj = new FileWriter(path);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                System.out.println("File didn't exist, but was created");
                System.out.println("File was opened");
            }
            else {
                System.out.println("File didn't exist and was not created");
                System.out.println("File was not opened");
            }
        }
        return this;
    }

    @Override
    public void close(){
        if (this.fileObj != null) {
            try {
                this.fileObj.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
            System.out.println("File was closed");
        }
        else{
            System.out.println("File was not open or not exists yet");
        }
    }

    @Override
    public void delete(String path) {
        if (new File(path).delete()){
```

```
            System.out.println("File was dropped");
        }
        else{
            System.out.println("File was not dropped");
        }
    }

    @Override
    public int getSize() {
        return this.size;
    }

    @Override
    public void addComponent(Component obj, boolean createIfNotExists) {
        System.out.println("It's a file. You cannot add anything");
    }

    @Override
    public void dropComponent(Component obj) {
        System.out.println("It's a file. You cannot drop anything.");
    }

    @Override
    public String toString(){
        return this.name;
    }

    @Override
    public String getPath(){
        return this.name;
    }
}
```

ВИСНОВОК

Під час виконання даної лабораторної роботи я детально ознайомився зі структурними шаблонами проектування Flyweight, Adapter, Bridge, Facade.

Закріпив знання реалізацією підсистеми «Картина», з використанням шаблону Facade.