Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

ОТЧЕТ
по лабораторной работе №6
по дисциплине СПП

Выполнил:
студент
группы ПО-5
Харкевич
Д.А.

Проверил:
Крощенко А.А.
Ст.преп. кафедры ИИТ

Брест 2021

Цель работы: приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка Java.

Задание:

Общее задание

• Прочитать задания, взятые из каждой группы.
• Определить паттерн проектирования, который может использоваться при реализации задания. Пояснить свой выбор.
• Реализовать фрагмент программной системы, используя выбранный паттерн. Реализовать все необходимые дополнительные классы.

Вариант 4

Задание 1

1) Проект «Туристическое бюро». Реализовать возможность выбора программы тура (проезд, проживание, питание, посещение музеев, выставок, экскурсии и т.д.). Должна формироваться итоговая стоимость заказа. – Паттерн Строитель

Реализация:

Tour.java

```
public class Tour {
    private int duration;

    private float transition;
    private float accomodation;
    private float meels;
    private float excursion;
    private float museum;
    private float theatre;

    private float cost;

    public float getCost() {
        return cost;
    }

    public static class Builder {
        private Tour newTour;

        public Builder() {
            newTour = new Tour();
            newTour.theatre = newTour.accomodation =
                    newTour.cost = newTour.excursion = newTour.meels =
                            newTour.transition = newTour.museum = 0;
            newTour.duration = 0;
        }

        public Builder setDuration(int duration) {
            newTour.duration = duration;
            return this;
        }

        public Builder byPlain() {
            newTour.transition = 30;
            return this;
        }
```

```
        public Builder byTrain() {
            newTour.transition = 20;
            return this;
        }

        public Builder byBus() {
            newTour.transition = 15;
            return this;
        }

        public Builder withAccomodation() {
            newTour.accomodation = newTour.duration * 5;
            return this;
        }

        public Builder withMeels() {
            newTour.meels = newTour.duration * 15;
            return this;
        }

        public Builder withExcursion() {
            newTour.excursion = 5;
            return this;
        }

        public Builder withTheatre() {
            newTour.theatre = 7;
            return this;
        }

        public Builder withMuseum() {
            newTour.museum = 3;
            return this;
        }

        public Tour build() {
            newTour.cost = newTour.excursion + newTour.theatre +
                    newTour.museum + newTour.meels + newTour.accomodation +
newTour.transition;
            return newTour;
        }
    }
}
```

Main.java

```
public class Main {

    public static void main(String[] args) {
        Tour tour1 = new Tour.Builder()
                .setDuration(3)
                .byBus()
                .withAccomodation()
                .withMeels()
                .withMuseum()
                .build();
        Tour tour2 = new Tour.Builder()
                .setDuration(2)
                .byPlain()
                .withAccomodation()
                .withTheatre()
                .withMuseum()
                .withExcursion()
```

```
                .build();
        Tour tour3 = new Tour.Builder()
                .setDuration(5)
                .byTrain()
                .withAccomodation()
                .withMeels()
                .withMuseum()
                .withExcursion()
                .withTheatre()
                .build();
        System.out.println("Tour 1: " + tour1.getCost() + "\nTour 2: " +
tour2.getCost() + "\nTour 3: " + tour3.getCost());
    }
}
```

Результат работы

```
Tour 1: 78.0
Tour 2: 55.0
Tour 3: 135.0

Process finished with exit code 0
```

Задания 2-3

2) Проект «Файловая система». Реализуйте модель работы файловой системы. Должна поддерживаться иерархичность ФС на уровне директорий и отдельных файлов. Файлы могут иметь все основные присущие им атрибуты (размер, расширение, дата создания и т.д.). – Паттерн Компоновщик
3) Реализовать вывод ФС из 2-й группы заданий. Вывод файлов/директорий должен осуществляться в случайном порядке. Вывести основные атрибуты каждого файла/директории. – Паттерн Итератор

Реализация

BaseClass.java

```
import java.util.Date;

abstract class BaseClass implements AbstractFile {
    String name;
    Date dateOfCreate;
    float  size;
    String type;

    public BaseClass(String name, Date dateOfCreate, float size, String type) {
        this.name = name;
        this.dateOfCreate = dateOfCreate;
        this.size = size;
        this.type = type;
    }

    public float getSize() {
        return size;
    }

    public String getName() {
        return name;
    }
```

```
    public Date getDateOfCreate() {
        return dateOfCreate;
    }

    public String getType() {
        return type;
    }

    public void setSize(float size) {
        this.size = size;
    }
}
```

File.java

```
import java.text.SimpleDateFormat;
import java.util.Date;

class File extends BaseClass implements AbstractFile {

    public File(String name, Date dateOfCreate, float size, String type) {
        super(name, dateOfCreate, size, type);
    }

    public void ls() {
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd.MM.yyyy");
        System.out.println(Main.compositeBuilder + name+ type + "    \t" + size +
"Mb\t" + simpleDateFormat.format(dateOfCreate));
    }
}
```

Directory.java

```
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

class Directory extends BaseClass implements AbstractFile {
    private ArrayList<BaseClass> includedFiles = new ArrayList<>();

    public Directory(String name, Date dateOfCreate) {
        super(name, dateOfCreate, 0, "dir");
    }

    public void add(BaseClass obj) {
        includedFiles.add(obj);
        size += obj.size;
    }

    public void ls() {
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd.MM.yyyy");
        System.out.println(Main.compositeBuilder + name + "  \t" + size + "Mb\t"
+ simpleDateFormat.format(dateOfCreate));
        Main.compositeBuilder.append("    ");
        for (BaseClass includedFile : includedFiles) includedFile.ls();
        Main.compositeBuilder.setLength(Main.compositeBuilder.length() - 3);
    }

    public ArrayList<BaseClass> getIncludedFiles() {
        ArrayList<BaseClass> files = new ArrayList<>();
        files.addAll(includedFiles);
        for (BaseClass b: includedFiles) {
            if (b.getClass() == Directory.class) {
                Directory dir = (Directory) b;
```

```
                files.addAll(dir.getIncludedFiles());
            }
        }
        return files;
    }
}
```

FileIterator.java

```
import java.util.ArrayList;
import java.util.List;

public class FileIterator implements AbstractFileIterator<BaseClass>{

    private List<BaseClass> files;
    private int position;

    public FileIterator(List<BaseClass> files) {
        this.files = files;
        position = 0;
    }

    @Override
    public boolean hasNext() {
        return position < files.size();
    }

    @Override
    public BaseClass next() {
        return files.get(position++);
    }

    @Override
    public BaseClass currentItem() {
        return files.get(position);
    }

    @Override
    public void reset() {
        position = 0;
    }


}
```

FileList.java

```
import java.util.ArrayList;
import java.util.List;

public class FilesList implements AbstaractFilesList<BaseClass> {
    private List<BaseClass> files = new ArrayList<>();

    public FilesList(Directory dir) {
        files.add(dir);
        files.addAll(dir.getIncludedFiles());
    }

    public AbstractFileIterator<BaseClass> iterator() {
        return new FileIterator(files);
    }
}
```

Main.java

```
import java.util.Date;

public class Main {

    public static StringBuffer compositeBuilder = new StringBuffer();

    public static void main(String[] args) {
        Directory music = new Directory("MUSIC", new Date());
        Directory beyonce = new Directory("Beyonce", new Date());
        Directory pop = new Directory("Pop", new Date());

        File file1 = new File("Kids", new Date(), (float)15.5, ".mp3");
        File file2 = new File("Teenage Dream", new Date(), (float)12.5, ".mp3");
        File file3 = new File("Flawless", new Date(), (float)18.2, ".mp3");
        File file4 = new File("Halo", new Date(), (float)10.1, ".mp3");

        music.add(file1);
        music.add(pop);
        pop.add(beyonce);
        beyonce.add(file3);
        beyonce.add(file4);
        pop.add(file2);
        music.ls();

        AbstaractFilesList<BaseClass> filesList = new FilesList(music);
        AbstractFileIterator<BaseClass> iterator = filesList.iterator();

        System.out.println('\n');
        while(iterator.hasNext()) {
            BaseClass cur = iterator.next();
            System.out.println(cur.getName() + '\t' + cur.getSize() + " Mb");
        }
    }
}
```

Результат работы

```
MUSIC   15.5Mb  22.05.2019
   Kids.mp3     15.5Mb  22.05.2019
   Pop      12.5Mb  22.05.2019
      Beyonce   28.300001Mb 22.05.2019
         Flawless.mp3       18.2Mb  22.05.2019
         Halo.mp3       10.1Mb  22.05.2019
      Teenage Dream.mp3     12.5Mb  22.05.2019


MUSIC   15.5 Mb
Kids    15.5 Mb
Pop 12.5 Mb
Beyonce 28.300001 Mb
Teenage Dream   12.5 Mb
Flawless    18.2 Mb
Halo    10.1 Mb

Process finished with exit code 0
```

Вывод: приобрели навыки применения паттернов проектирования при решении практических задач с использованием языка Java.