МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
**Кафедра ИИТ**

# ОТЧЁТ

по лабораторной работе №6
за 1 семестр 3 курса

Выполнил:
студент группы ПО-9(1)
3 курса
Зейденс Никита
Вячеславович

Проверил:
Крощенко
А. А.

Брест, 2024

**Цель:** Приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка Java.

**Вариант 4**

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

**Код программы:**

```
import java.util.ArrayList;
// Класс, представляющий тур
class Tour {
    private Transportation transportation;
    private Accommodation accommodation;
    private Meals meals;
    private ArrayList<Sightseeing> sightseeingList;
    public Tour() { sightseeingList = new ArrayList<>(); }
    public void setTransportation(Transportation transportation)
{ this.transportation = transportation; }
    public void setAccommodation(Accommodation accommodation)
{ this.accommodation = accommodation; }
    public void setMeals(Meals meals) { this.meals = meals; }
    public void addSightseeing(Sightseeing sightseeing)
{ sightseeingList.add(sightseeing); }
    public double getTotalCost() {
        double totalCost = 0;
        if (transportation != null) { totalCost +=
transportation.getCost(); }
        if (accommodation != null) { totalCost += accommodation.getCost(); }
        if (meals != null) { totalCost += meals.getCost(); }
        for (Sightseeing sightseeing : sightseeingList) { totalCost +=
sightseeing.getCost(); }
        return totalCost;
    }
}
// Класс, представляющий транспорт
class Transportation {
    private double cost;
    public Transportation(double cost) { this.cost = cost; }
    public double getCost() { return cost; }
}
// Класс, представляющий проживание
class Accommodation {
    private double cost;
    public Accommodation(double cost) { this.cost = cost; }
    public double getCost() { return cost; }
}
// Класс, представляющий питание
class Meals {
    private double cost;
    public Meals(double cost) { this.cost = cost; }
    public double getCost() { return cost; }
}
// Класс, представляющий экскурсию
class Sightseeing {
    private double cost;
    public Sightseeing(double cost) { this.cost = cost; }
    public double getCost() { return cost; }
}
// Класс, отвечающий за построение тура
```

```
class TourBuilder {
    private Tour tour;
    public TourBuilder() { tour = new Tour(); }
    public TourBuilder setTransportation(Transportation transportation) {
        tour.setTransportation(transportation);
        return this;
    }
    public TourBuilder setAccommodation(Accommodation accommodation) {
        tour.setAccommodation(accommodation);
        return this;
    }
    public TourBuilder setMeals(Meals meals) {
        tour.setMeals(meals);
        return this;
    }
    public TourBuilder addSightseeing(Sightseeing sightseeing) {
        tour.addSightseeing(sightseeing);
        return this;
    }
    public Tour build() { return tour; }
}

public class Main {
    public static void main(String[] args) {
        Tour tour = new TourBuilder()
                .setTransportation(new Transportation(100.0))
                .setAccommodation(new Accommodation(200.0))
                .setMeals(new Meals(50.0))
                .addSightseeing(new Sightseeing(30.0))
                .addSightseeing(new Sightseeing(40.0))
                .build();
        double totalCost = tour.getTotalCost();
        System.out.println("Total cost of the tour: $" + totalCost);
    }
}
```

**Результат работы:**

```
Total cost of the tour: $420.0
```

**Задание 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.).

**Код программы:**

```
 import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
// Общий интерфейс компонента (каталога или файла)
interface FileSystemComponent {
    void showDetails(int depth);
}
// Класс, представляющий файл
class File implements FileSystemComponent {
    private String name;
    private int size;
    private Date createdDate;
    private boolean canRead;
```

```
    private boolean canWrite;
    private boolean canExecute;

    public File(String name, int size, Date createdDate, boolean canRead,
boolean canWrite, boolean canExecute) {
        this.name = name;
        this.size = size;
        this.createdDate = createdDate;
        this.canRead = canRead;
        this.canWrite = canWrite;
        this.canExecute = canExecute;
    }

    @Override
    public void showDetails(int depth) {
        StringBuilder indent = new StringBuilder();
        for (int i = 0; i < depth; i++) {
            indent.append("\t");
        }
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy
HH:mm");
        String formattedDate = dateFormat.format(createdDate);

        System.out.print(indent + " " + name);
        System.out.print(" " + formattedDate);
        System.out.print(" " + size + " b ");
        if(this.canRead) { System.out.print("r"); }
        else { System.out.print("-"); }
        if(this.canWrite) { System.out.print("w"); }
        else { System.out.print("-"); }
        if(this.canExecute) { System.out.print("x\n"); }
        else { System.out.print("-\n"); }
    }
}

// Класс, представляющий директорию
class Directory implements FileSystemComponent {
    private String name;
    private List<FileSystemComponent> components;
    public Directory(String name) {
        this.name = name;
        components = new ArrayList<>();
    }
    public void addComponent(FileSystemComponent component)
{ components.add(component); }
    public void removeComponent(FileSystemComponent component)
{ components.remove(component); }
    @Override
    public void showDetails(int depth) {
        StringBuilder indent = new StringBuilder();
        for (int i = 0; i < depth; i++) { indent.append("\t"); }
        System.out.println(indent + " " + name);
        for (FileSystemComponent component : components)
{ component.showDetails(depth + 1); }
    }
}
// Пример использования
public class Main {
    public static void main(String[] args) {
        File file1 = new File("file1.txt", 100, new Date(), true, true,
false);
        File file2 = new File("file2.txt", 50, new Date(), true, false,
false);
        File file3 = new File("file3.txt", 75, new Date(), true, true, true);
```

```
        File file4 = new File("file4.txt", 120, new Date(), false, false,
false);

        Directory dir1 = new Directory("dir1");
        Directory dir2 = new Directory("dir2");
        Directory dir3 = new Directory("dir3");
        Directory dir4 = new Directory("dir4");

        dir1.addComponent(file1);
        dir1.addComponent(file2);

        dir2.addComponent(dir1);
        dir2.addComponent(file3);

        dir3.addComponent(file4);

        dir4.addComponent(dir2);
        dir4.addComponent(dir3);

        dir4.showDetails(0);
    }
}
```

**Результат работы:**

```
dir4
    dir2
        dir1
            file1.txt 15.05.2024 03:52 100 b rw-
            file2.txt 15.05.2024 03:52 50 b r--
        file3.txt 15.05.2024 03:52 75 b rwx
    dir3
        file4.txt 15.05.2024 03:52 120 b ---
```

**Задание 3:** Реализовать вывод ФС из 2-й группы заданий. Вывод файлов/директорий должен осуществляться в случайном порядке. Вывести основные атрибуты каждого файла/директории.

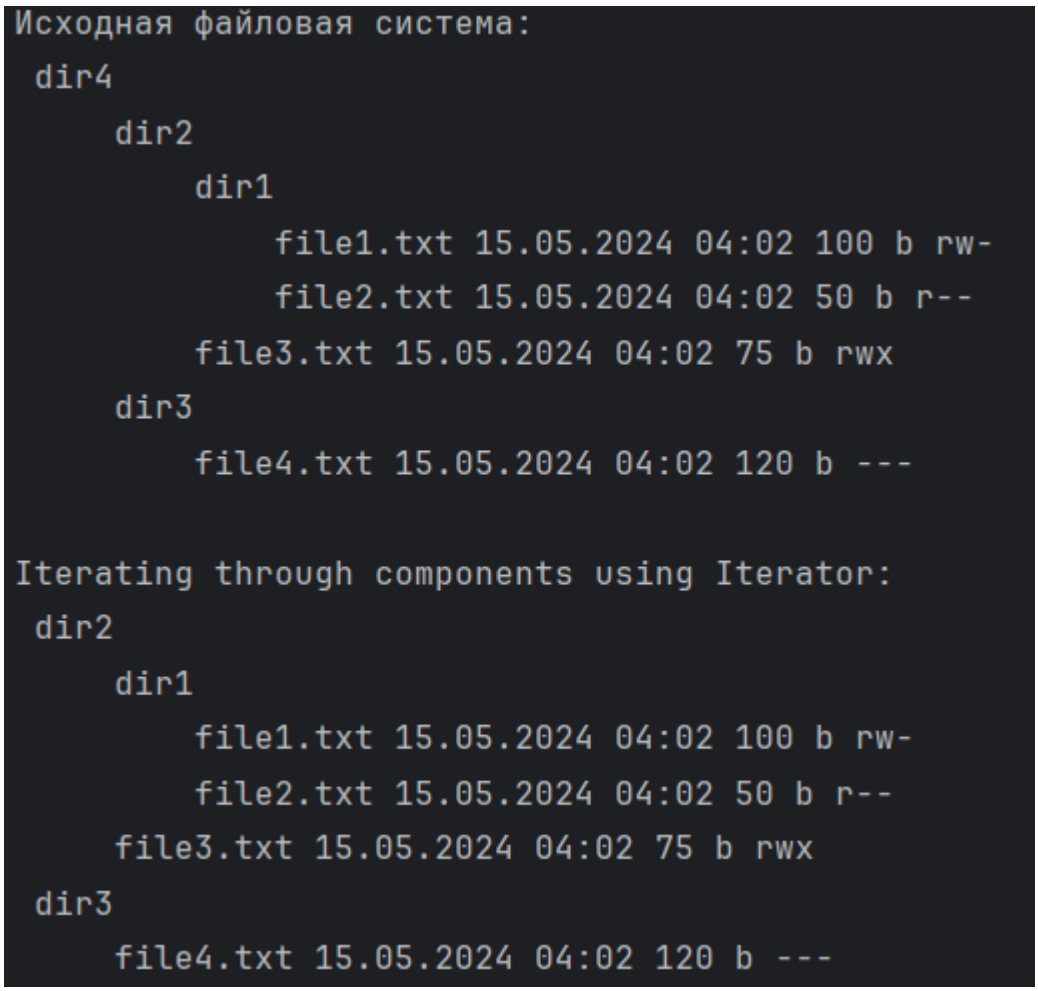
**Код программы:**

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

// Общий интерфейс компонента (каталога или файла)
interface FileSystemComponent {
    void showDetails(int depth);
}

// Класс, представляющий файл
class File implements FileSystemComponent {
    private String name;
    private int size;
    private Date createdDate;
    private boolean canRead;
```

```
    private boolean canWrite;
    private boolean canExecute;
    public File(String name, int size, Date createdDate, boolean canRead,
boolean canWrite, boolean canExecute) {
        this.name = name;
        this.size = size;
        this.createdDate = createdDate;
        this.canRead = canRead;
        this.canWrite = canWrite;
        this.canExecute = canExecute;
    }
    @Override
    public void showDetails(int depth) {
        StringBuilder indent = new StringBuilder();
        for (int i = 0; i < depth; i++) {
            indent.append("\t");
        }
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy
HH:mm");
        String formattedDate = dateFormat.format(createdDate);

        System.out.print(indent + " " + name);
        System.out.print(" " + formattedDate);
        System.out.print(" " + size + " b ");
        if(this.canRead) { System.out.print("r"); }
        else { System.out.print("-"); }
        if(this.canWrite) { System.out.print("w"); }
        else { System.out.print("-"); }
        if(this.canExecute) { System.out.print("x\n"); }
        else { System.out.print("-\n"); }
    }
}

// Класс, представляющий директорию
class Directory implements FileSystemComponent {
    private String name;
    private List<FileSystemComponent> components;
    public Directory(String name) {
        this.name = name;
        components = new ArrayList<>();
    }
    public void addComponent(FileSystemComponent component)
{ components.add(component); }
    public void removeComponent(FileSystemComponent component)
{ components.remove(component); }
    @Override
    public void showDetails(int depth) {
        StringBuilder indent = new StringBuilder();
        for (int i = 0; i < depth; i++) { indent.append("\t"); }
        System.out.println(indent + " " + name);
        //shuffleComponents();
        for (FileSystemComponent component : components)
{ component.showDetails(depth + 1); }
    }
    private void shuffleComponents() { Collections.shuffle(components); }
    public Iterator<FileSystemComponent> createIterator() { return new
DirectoryIterator(components.iterator()); }
    // Итератор для перебора компонентов директории
    private class DirectoryIterator implements Iterator<FileSystemComponent>
{
        private Iterator<FileSystemComponent> iterator;
        public DirectoryIterator(Iterator<FileSystemComponent> iterator)
{ this.iterator = iterator; }
        @Override
```

```
        public boolean hasNext() { return iterator.hasNext(); }
        @Override
        public FileSystemComponent next() { return iterator.next(); }
    }
}

// Пример использования
public class Main {
    public static void main(String[] args) {
        File file1 = new File("file1.txt", 100, new Date(), true, true,
false);
        File file2 = new File("file2.txt", 50, new Date(), true, false,
false);
        File file3 = new File("file3.txt", 75, new Date(), true, true, true);
        File file4 = new File("file4.txt", 120, new Date(), false, false,
false);

        Directory dir1 = new Directory("dir1");
        Directory dir2 = new Directory("dir2");
        Directory dir3 = new Directory("dir3");
        Directory dir4 = new Directory("dir4");

        dir1.addComponent(file1);
        dir1.addComponent(file2);

        dir2.addComponent(dir1);
        dir2.addComponent(file3);

        dir3.addComponent(file4);

        dir4.addComponent(dir2);
        dir4.addComponent(dir3);

        //Исходная ФС
        System.out.print("Исходная файловая система:\n");
        dir4.showDetails(0);

        // Перебор компонентов директории с помощью итератора
        System.out.println("\nIterating through components using Iterator:");
        Iterator<FileSystemComponent> iterator = dir4.createIterator();
        while (iterator.hasNext()) {
            FileSystemComponent component = iterator.next();
            component.showDetails(0);
        }
    }
}
```

**Результат работы:**




**Вывод:** Практические навыки применения паттернов проектирования при решении практических задач с использованием языка Java были приобретены.