МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**КАФЕДРА ИИТ**

ОТЧЁТ

по лабораторной работе №6

Выполнила:

студентка 3 курса

группы ПО-9

Шубич Дарья Константинова

Проверил:

Крощенко А.А.

Брест 2024

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

**Вариант 4**

**Задание 1**

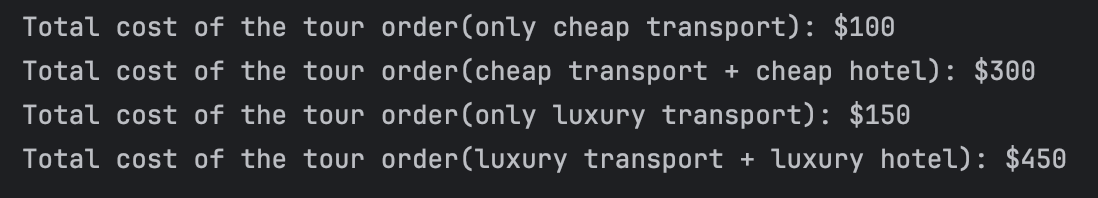
Проект «Туристическое бюро». Реализовать возможность выбора программы тура (проезд, проживание, питание, посещение музеев, выставок, экскурсии и т.д.). Должна формироваться итоговая стоимость заказа.

Паттерн Стратегия позволяет определить семейство алгоритмов, инкапсулировать каждый из них и обеспечивать их взаимозаменяемость. В данном случае, каждая программа тура (проезд, проживание, питание, музеи, экскурсии и др.) может рассматриваться как отдельная стратегия. Клиент (пользователь) может выбирать нужные стратегии, и они будут применяться к заказу.

**Входные данные:**

TourPackageStrategy transportationStrategy = new TransportationStrategy();  
TourPackageStrategy accommodationStrategy = new AccommodationStrategy();  
TourPackageStrategy luxuryAccommodationStrategy = new LuxuryHotelAccommodationStrategy();  
TourPackageStrategy firstClassTransportation = new FirstClassTrainTransportationStrategy();  
  
TourOrder tourOrder = new TourOrder(transportationStrategy);  
  
int totalCost = tourOrder.calculateTotalCost();  
System.*out*.println("Total cost of the tour order(only cheap transport): $" + totalCost);  
  
tourOrder.setStrategy(accommodationStrategy);  
  
totalCost += tourOrder.calculateTotalCost();  
System.*out*.println("Total cost of the tour order(cheap transport + cheap hotel): $" + totalCost);  
  
tourOrder.setStrategy(firstClassTransportation);  
  
totalCost = tourOrder.calculateTotalCost();  
System.*out*.println("Total cost of the tour order(only luxury transport): $" + totalCost);  
  
tourOrder.setStrategy(luxuryAccommodationStrategy);  
  
totalCost += tourOrder.calculateTotalCost();  
System.*out*.println("Total cost of the tour order(luxury transport + luxury hotel): $" + totalCost);  
  
System.*out*.println();

**Результат программы**



**Код программы:**

interface TourPackageStrategy {  
 int calculateCost();  
}

public class TransportationStrategy implements TourPackageStrategy{  
  
 @Override  
 public int calculateCost() {  
 return 100;  
 }  
}

public class LuxuryHotelAccommodationStrategy implements TourPackageStrategy {  
 @Override  
 public int calculateCost() {  
 return 300; // Пример стоимости проживания в отеле высокого класса  
 }  
}

public class FirstClassTrainTransportationStrategy implements TourPackageStrategy{  
 @Override  
 public int calculateCost() {  
 return 150; // Пример стоимости транспорта первого класса  
 }  
}

public class AccommodationStrategy implements TourPackageStrategy{  
 @Override  
 public int calculateCost() {  
 return 200;  
 }  
}

**Задание 2**

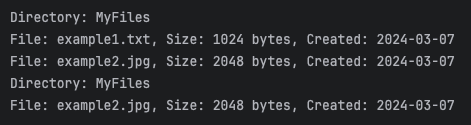
Проект «Файловая система». Реализуйте модель работы файловой системы. Должна поддерживаться иерархичность ФС на уровне директорий и отдельных файлов. Файлы могут иметь все основные присущие им атрибуты (размер, расширение, дата создания и т.д.).

Для реализации проекта "Файловая система" можно использовать структурный паттерн проектирования "Компоновщик" (Composite). Этот паттерн позволяет клиентам единообразно работать с индивидуальными объектами и их композициями (группами объектов).

**Входные данные:**

File file1 = new File("example1", 1024, "txt", "2024-03-07");  
File file2 = new File("example2", 2048, "jpg", "2024-03-07");  
  
Directory directory = new Directory("MyFiles");  
directory.addChild(file1);  
directory.addChild(file2);  
  
directory.showInfo();  
directory.removeChild(file1);  
directory.showInfo();

**Выходные данные:**



**Код программы:**

package org.example;  
  
interface FileSystemComponent {  
 void showInfo();  
}

public class File implements FileSystemComponent {  
 private String name;  
 private int size;  
 private String extension;  
 private String createdDate;  
  
 public File(String name, int size, String extension, String createdDate) {  
 this.name = name;  
 this.size = size;  
 this.extension = extension;  
 this.createdDate = createdDate;  
 }  
  
 @Override  
 public void showInfo() {  
 System.*out*.println("File: " + name + "." + extension + ", Size: " + size + " bytes, Created: " + createdDate);  
 }  
}

class Directory implements FileSystemComponent {  
 private String name;  
 private List<FileSystemComponent> children;  
  
 public Directory(String name) {  
 this.name = name;  
 this.children = new ArrayList<>();  
 }  
  
 public void addChild(FileSystemComponent child) {  
 children.add(child);  
 }  
  
 public void removeChild(FileSystemComponent child) {  
 children.remove(child);  
 }  
  
 @Override  
 public void showInfo() {  
 System.*out*.println("Directory: " + name);  
 for (FileSystemComponent child : children) {  
 child.showInfo();  
 }  
 }  
}

**Задание 3**

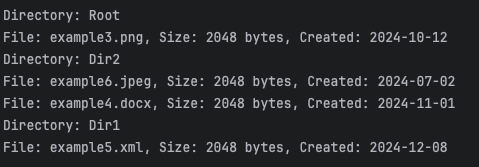
Реализовать вывод ФС из 2-й группы заданий. Вывод файлов/директорий должен осуществляться в случайном порядке. Вывести основные атрибуты каждого файла/директории.

Для реализации задания, связанного с файловой системой и выводом ее содержимого в случайном порядке, можно воспользоваться паттерном проектирования "Компоновщик" (Composite). Данный паттерн позволяет создавать структуры из простых и составных объектов, обрабатывая их единообразно. В контексте файловой системы, файлы и директории могут рассматриваться как компоненты, и мы можем легко реализовать вывод их содержимого в случайном порядке.

**Входные данные:**

System.*out*.println();  
DirectoryThird root = new DirectoryThird("Root");  
  
File file3 = new File("example3", 2048, "png", "2024-10-12");  
File file4 = new File("example4", 2048, "docx", "2024-11-01");  
  
  
DirectoryThird dir1 = new DirectoryThird("Dir1");  
File file5 = new File("example5", 2048, "xml", "2024-12-08");  
dir1.add(file5);  
  
  
DirectoryThird dir2 = new DirectoryThird("Dir2");  
File file6 = new File("example6", 2048, "jpeg", "2024-07-02");  
dir2.add(file6);  
  
root.add(file3);  
root.add(file4);  
root.add(dir1);  
root.add(dir2);  
  
root.showInfo();

**Выходные данные:**

****

**Код программы:**

Класс File и интерфейс FileSystemComponent такие же как и во втором задании.

package org.example;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
public class DirectoryThird implements FileSystemComponent{  
 private String name;  
 private List<FileSystemComponent> children = new ArrayList<>();  
  
 public DirectoryThird(String name) {  
 this.name = name;  
 }  
  
 public void add(FileSystemComponent component) {  
 children.add(component);  
 }  
  
 public void remove(FileSystemComponent component) {  
 children.remove(component);  
 }  
  
 @Override  
 public void showInfo() {  
 System.*out*.println("Directory: " + name);  
 List<FileSystemComponent> shuffledChildren = new ArrayList<>(children);  
 Collections.*shuffle*(shuffledChildren);  
 for (FileSystemComponent child : shuffledChildren) {  
 child.showInfo();  
 }  
  
 }  
}