Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №2**

по дисциплине: «Теория алгоритмов»

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС-22.06  Мельников М.Н. |
| Проверила:  старший преподаватель  кафедры ПО  Корнеева Е.И. |

Тверь 2024

Формулировка задач, описание условий.

17 вариант " Путешествие". Сложность: rare.

Задача:

Разработать программу для расчета стоимости путешествия на разном транспорте.

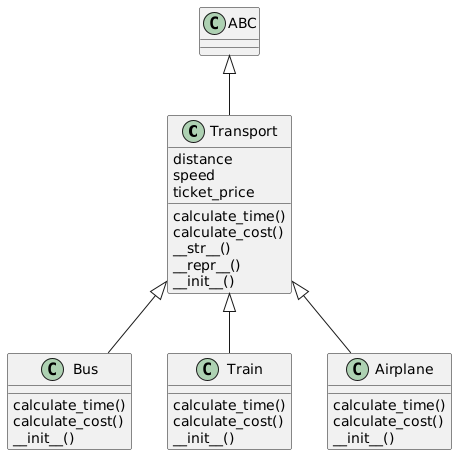
Условия выполнения:

* интерфейс с возможностями ввода требуемых параметров и отображения результатов расчёта
* вариант реализуется с использованием классов и объектов python
* возможность сохранить результаты в отчёт формата .doc или .xls (например, пакеты python-docx и openpyxl).

Ссылка на репозиторий:  
[TheoryAlgLabs/Labs/Lab2 at main · Zawiaha/TheoryAlgLabs](https://github.com/Zawiaha/TheoryAlgLabs/tree/main/Labs/Lab2)

Проделанная работа.

# UML – диаграммы.



Создание структуры классов:

* Реализованы классы Bus, Train, и Airplane обозначающие тип транспорта. Каждый класс наследуется от класса Transport.
* В классе Transport добавлены два абстрактных метода для расчета времени и стоимости, а также реализованы 3 dunder метода..

Интерфейс программы:

Реализован консольный интерфейс для взаимодействия с пользователем, позволяющий выбирать тип транспорта, их характеристик и подсчета цены.

Сохранение результатов в .docx и .xls:

Добавлена поддержка сохранения отчетов с использованием библиотеки python-docx. Программа позволяет сохранять данные в документе.

Вывод.

В ходе лабораторной работы была реализована программа, которая++ представляет собой инструмент для расчета времени и стоимости поездки на различных видах транспорта (автобус, поезд, самолет). Пользователь может выбрать тип транспорта, ввести расстояние, среднюю скорость, цену билета и количество пассажиров, после чего программа рассчитывает и выводит время поездки и ее стоимость. Также пользователю предоставляется возможность сохранить результаты в формате DOCX или XLS. Это полезный инструмент для планирования поездок и сравнения различных вариантов транспорта.

Код

# Calculations.py

from transport import Bus, Train, Airplane

def create\_transport(mode, distance, speed, ticket\_price):

    if mode == 'bus':

        return Bus(distance, speed, ticket\_price)

    elif mode == 'train':

        return Train(distance, speed, ticket\_price)

    elif mode == 'airplane':

        return Airplane(distance, speed, ticket\_price)

    else:

        raise ValueError("Invalid transport mode")

def get\_trip\_info(transport, passengers):

    time = transport.calculate\_time()

    cost = transport.calculate\_cost(passengers)

    return time, cost

# main.py

import calculations

import report

def main():

    results = {}

    print("1. Bus")

    print("2. Train")

    print("3. Airplane")

    choice = input("Выберите тип транспорта (1,2,3): ")

    if choice == '1':

        mode = 'bus'

    elif choice == '2':

        mode = 'train'

    elif choice == '3':

        mode = 'airplane'

    else:

        print("Неверное значение")

        return

    distance = float(input("Ввод растояния (в км): "))

    speed = float(input("Ввод средней скорости (км/ч): "))

    ticket\_price = float(input("Цена билета: "))

    passengers = int(input("Кол-во пасажиров: "))

    transport = calculations.create\_transport(mode, distance, speed, ticket\_price)

    time, cost = calculations.get\_trip\_info(transport, passengers)

    results[mode] = {"time": time, "cost": cost}

    print("Результат:")

    for mode, info in results.items():

        print(f'Транспорт: {mode}, Время: {info["time"]:.2f} часов, Стоимость: ${info["cost"]:.2f}')

    save\_choice = input("Хотите сохранить результат? (Y - да, N - нет): ")

    if save\_choice.lower() == 'y':

        format\_choice = input("Выберите формат (docx/xls): ")

        if format\_choice.lower() == 'docx':

            report.save\_to\_docx(results)

            print("Results saved to report.docx")

        elif format\_choice.lower() == 'xls':

            report.save\_to\_xls(results)

            print("Results saved to report.xlsx")

        else:

            print("Invalid format choice.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

# report.py

from docx import Document

import openpyxl

def save\_to\_docx(data, filename='report.docx'):

    doc = Document()

    doc.add\_heading('Trip Report', level=1)

    for mode, info in data.items():

        doc.add\_paragraph(f'Тип транспорта: {mode}')

        doc.add\_paragraph(f'Время пути: {info["time"]:.2f} hours')

        doc.add\_paragraph(f'Стоимость: ${info["cost"]:.2f}')

    doc.save(filename)

def save\_to\_xls(data, filename='report.xlsx'):

    wb = openpyxl.Workbook()

    ws = wb.active

    ws.title = "Trip Report"

    ws.append(["Transport", "Time (hours)", "Cost ($)"])

    for mode, info in data.items():

        ws.append([mode, info["time"], info["cost"]])

    wb.save(filename)

# Transport.py

from abc import ABC, abstractmethod

class Transport(ABC):

    def \_\_init\_\_(self, distance, speed, ticket\_price):

        self.distance = distance

        self.speed = speed

        self.ticket\_price = ticket\_price

    @abstractmethod

    def calculate\_time(self):

        pass

    @abstractmethod

    def calculate\_cost(self, passengers):

        pass

    def \_\_str\_\_(self):

        return f"{self.\_\_class\_\_.\_\_name\_\_} with distance {self.distance} km"

    def \_\_repr\_\_(self):

        return f"<Transport: {self.\_\_class\_\_.\_\_name\_\_}, Distance: {self.distance}, Speed: {self.speed}, Ticket Price: {self.ticket\_price}>"

class Bus(Transport):

    def calculate\_time(self):

        return self.distance / self.speed

    def calculate\_cost(self, passengers):

        return passengers \* self.ticket\_price

class Train(Transport):

    def calculate\_time(self):

        return self.distance / self.speed

    def calculate\_cost(self, passengers):

        return passengers \* self.ticket\_price

class Airplane(Transport):

    def calculate\_time(self):

        return self.distance / self.speed

    def calculate\_cost(self, passengers):

        return passengers \* self.ticket\_price