Лабораторна робота №3.

Об’єктно-орієнтована декомпозиція. Основи введення/виведення Java SE.

Мета

Використання об’єктно-орієнтованого підходу для розробки об’єкта предметної (прикладної) галузі. Оволодіння навичками управління введенням/виведенням даних з використанням класів Java SE.

Вимоги

1. Використовуючи об’єктно-орієнтований аналіз, реалізувати класи для представлення сутностей відповідно списку прикладних задач - domain-об’єктів (Прикладні задачі. Список №2. 20 варіантів)

2. Забезпечити та продемонструвати коректне введення та відображення кирилиці.

3. Продемонструвати можливість управління масивом domain-об’єктів.

4. Забезпечити можливість збереження і відновлення масива об’єктів рішення завдання з Прикладні задачі. Список №2. 20 варіантів.

5. Забороняється використання стандартного протокола серіалізації .

6. Продемонструвати використання моделі Long Term Persistence .

7. Забезпечити діалог з користувачем у вигляді текстового меню.

8. При збереженні та відновленні даних забезпечити діалоговий режим вибору директорії з відображенням вмісту і можливістю переміщення по підкаталогах.

Розробник: Гринишин Анастасія , КН-108, номер варіанту індивідуального завдання – 9.

Задача:

Квиткова каса. Дані про маршрут: маршрут - необмежений набір значень у вигляді “назва станції, час прибуття (для проміжних і кінцевої), час відправлення (для початкової та проміжних), кількість вільних місць”; загальна кількість місць; дні тижня; номер рейсу.

Ієрархія та структура класів:

1.Клас Main, який містить 1 функцію – main.

2.Клас Filego, який містить 2 функції – doFile, make_info.

3.Клас Demain,який містить 3 поля – numberFlight, date, numberFlight і їхні гетери та сетери.

4.Клас Station, який містить 4 поля – nameStation, dateArrival, dateDeparture, freeSeat.

5.Клас Save, який містить 2 функції – save, resave.

Фажливий фрагмент коду:

```
static int N=1;
ArrayList<Station> route = new ArrayList<>();
private String numberOfSeats;
private String date;
private String numberFlight;
FileWriter nFile = new FileWriter( fileName: "file1.txt");


public Demain() throws IOException {
    Station s2;
    Scanner in =new Scanner(System.in);
    System.out.println("Введіть номер рейсу " + " : ");
    String number = in.nextLine();
    setNumberFlight(number);
    System.out.println("Введіть дату прибуття: ");
    String date = in.nextLine();
    setDate(date);
    System.out.println("Введіть кількість місць");
    String places = in.nextLine();
    setNumberOfSeats(places);
    System.out.println("Введіть кіскість зупинок(включно з початковоі і кінцевої): ");
    N = in.nextInt();
    for (int i = 0; i < N; i++) {
        Station station = new Station();
        route.add(station);
    }
}



public  Demain(ArrayList<String> lines) throws IOException {
    System.out.println(i);
        setNumberFlight(lines.get(i));
        i++;
        setDate(lines.get(i));
        i++;
        setNumberOfSeats(lines.get(i));
        i++;
        N= Integer.parseInt(lines.get(i));
        i++;
        for (int j = 0;j<N;j++){
            Station station = new Station(lines,i);
            route.add(station);
            i+=4;
        }

}


public String getNumberOfSeats() { return numberOfSeats; }
public void setNumberOfSeats(String places) { numberOfSeats=places; }

public String getDate() { return date; }
public void setDate(String date) { this.date = date; }

public String getNumberFlight() { return numberFlight; }
public void setNumberFlight(String numberFlight) { this.numberFlight=numberFlight; }
```

Demain

Висновок

У ході роботи дізналася про домейн-файли, як їх зберігати. Використовувати гетери та сетери.