МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
КАФЕДРА ИИТ

Лабораторная работа №3

По дисциплине: «Современные платформы программирования»

Выполнил:
Студент 3 курса
группы ПО-8:
Макаревич Е.С.
Проверил:
Крощенко А.А.

Брест 2024

Цель работы: научиться создавать и использовать классы в программах на языке программирования Java.

Вариант 17

Задание 1. Множество символов ограниченной мощности – Предусмотреть возможность объединения двух множеств, вывода на печать элементов множества, а так же метод, определяющий, принадлежит ли указанное значение множеству. Класс должен содержать методы, позволяющие добавлять и удалять элемент в/из множества. Конструктор должен позволить создавать объекты с начальной инициализацией. Мощность множества задается при создании объекта. Реализацию множества осуществить на базе одномерного массива. Реализовать метод equals, выполняющий сравнение объектов данного типа.

Код программы:

Main.java

```
package task01;

public class Main {
    public static void main(String[] args) {
        Task01 set1 = new Task01(5);
        set1.add('a');
        set1.add('b');
        set1.add('c');
        System.out.println("Set 1: " + set1);

        Task01 set2 = new Task01(5);
        set2.add('b');
        set2.add('c');
        set2.add('d');
        System.out.println("Set 2: " + set2);

        set1.remove('b');
        System.out.println("Set 1 after removing 'b': " + set1);

        set2.remove('c');
        System.out.println("Set 2 after removing 'c': " + set2);

        System.out.println("Is 'c' in Set 1? " + set1.contains('c'));
        System.out.println("Is 'd' in Set 1? " + set1.contains('d'));

        System.out.println("Are Set 1 and Set 2 equal? " + set1.equals(set2));

    }
}
```

Task01.java

```
package task01;

import java.util.Arrays;

public class Task01 {
    private char[] symbols;
    private int size;
```

```
    public Task01(int capacity) {
        symbols = new char[capacity];
        size = 0;
    }

    public void add(char element) {
        if (size >= symbols.length) {
            System.out.println("Array is full. Cannot add more elements.");
            return;
        }

        if (contains(element)) {
            System.out.println("Element " + element + " already exists in the
array.");
            return;
        }

        symbols[size++] = element;
        System.out.println("Added element " + element);
    }

    public void remove(char element) {
        for (int i = 0; i < size; i++) {
            if (symbols[i] == element) {
                symbols[i] = symbols[size - 1];
                symbols[size - 1] = 0;
                size--;
                System.out.println("Removed element " + element);
                return;
            }
        }
        System.out.println("Element " + element + " not found in the array.");
    }

    public boolean contains(char element) {
        for (int i = 0; i < size; i++) {
            if (symbols[i] == element) {
                return true;
            }
        }
        return false;
    }

    @Override
    public String toString() {
        return Arrays.toString(Arrays.copyOf(symbols, size));
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }
        if (!(obj instanceof Task01)) {
            return false;
        }
        Task01 other = (Task01) obj;
        return Arrays.equals(Arrays.copyOf(symbols, size),
Arrays.copyOf(other.symbols, other.size));
    }
}
```

Результат программы:

```
Added element a
Added element b
Added element c
Set 1: [a, b, c]
Added element b
Added element c
Added element d
Set 2: [b, c, d]
Removed element b
Set 1 after removing 'b': [a, c]
Removed element c
Set 2 after removing 'c': [b, d]
Is 'c' in Set 1? true
Is 'd' in Set 1? false
Are Set 1 and Set 2 equal? false
```

Задание 2. Система оповещений на дорожном вокзале

Автоматизированная информационная система на железнодорожном вокзале содержит сведения об отправлении поездов дальнего следования.

Составить программу, которая должна хранить расписание поездов в структурированном, отсортированном по времени отправления виде (используя бинарное дерево).
• Обеспечивает первоначальный ввод данных в информационную систему о текущем расписании из файла и формирование дерева;
• Печатает все расписание на экран по команде;
• Выводит информацию о поезде по номеру поезда;
• По названию станции назначения выводит данные обо всех поездах, которые следуют до этой станции;
• Список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;
• Список поездов, отправляющихся до заданного пункта назначения и имеющих общие места;
• За 10, 5, 3 минуты до отправления поезда показывает информационное сообщение об отправлении поезда.

Код программы:

Main.java

```java
package task02;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;
```

```
public class Main {
    public static void main(String[] args) {
        TrainSchedule schedule = new TrainSchedule();

        try (BufferedReader reader = new BufferedReader(new
FileReader("src/task02/trains.txt"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split(", ");
                int id = Integer.parseInt(parts[0]);
                int trainNumber = Integer.parseInt(parts[1]);
                String departureStation = parts[2];
                String destinationStation = parts[3];
                String departureTime = parts[4];
                int seatsAvailable = Integer.parseInt(parts[5]);
                schedule.addTrain(new Train(id, trainNumber, departureStation,
destinationStation, departureTime, seatsAvailable));
            }
        } catch (IOException e) {
            System.err.println("Error reading from file: " + e.getMessage());
        }

        Scanner scanner = new Scanner(System.in);

        int choice;
        do {
            System.out.println("\nMenu:");
            System.out.println("1. Print train schedule");
            System.out.println("2. Find train by number");
            System.out.println("3. Find trains by destination");
            System.out.println("4. Find trains by destination and time");
            System.out.println("5. Find trains by destination and seats
available");
            System.out.println("6. Check trains departure in 10, 5, and 3
minutes");
            System.out.println("7. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    schedule.printSchedule();
                    break;
                case 2:
                    System.out.print("Enter train number: ");
                    int trainNumber = scanner.nextInt();
                    schedule.findTrainByNumber(trainNumber);
                    break;
                case 3:
                    scanner.nextLine();
                    System.out.print("Enter destination station: ");
                    String destination = scanner.nextLine();
                    schedule.findTrainsByDestination(destination);
                    break;
                case 4:
                    scanner.nextLine();
                    System.out.print("Enter destination station: ");
                    String dest = scanner.nextLine();
                    System.out.print("Enter time: ");
                    String time = scanner.nextLine();
                    schedule.findTrainsByDestinationAndTime(dest, time);
                    break;
                case 5:
```

```
                    scanner.nextLine();
                    System.out.print("Enter destination station: ");
                    String destStation = scanner.nextLine();
                    System.out.print("Enter minimum seats available: ");
                    int minSeats = scanner.nextInt();
                    showTrainsByDestinationAndSeats(schedule, destStation,
minSeats);
                    break;
                case 6:
                    checkDepartureInMinutes(schedule);
                    break;
                case 7:
                    System.out.println("Exiting program...");
                    break;
                default:
                    System.out.println("Invalid choice! Please enter a number
between 1 and 7.");
            }
        } while (choice != 7);
        scanner.close();
    }

    private static void showTrainsByDestinationAndSeats(TrainSchedule schedule,
String destinationStation, int seats) {
        schedule.findTrainsByDestinationAndSeats(destinationStation, seats);
    }

    private static void checkDepartureInMinutes(TrainSchedule schedule) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter current time in minutes: ");
        int currentTimeInMinutes = scanner.nextInt();

        schedule.checkDepartureInMinutes(currentTimeInMinutes, 10);
        schedule.checkDepartureInMinutes(currentTimeInMinutes, 5);
        schedule.checkDepartureInMinutes(currentTimeInMinutes, 3);
    }
}
```

Train.java

```
package task02;

import java.time.LocalTime;

public class Train {
    private String departureTime;
    private int id;
    private int trainNumber;
    private String departureStation;
    private String destinationStation;
    private int seatsAvailable;

    public Train(int id, int trainNumber, String departureStation, String
destinationStation, String departureTime, int seatsAvailable) {
        this.id = id;
        this.trainNumber = trainNumber;
        this.departureStation = departureStation;
        this.destinationStation = destinationStation;
        this.departureTime = departureTime;
        this.seatsAvailable = seatsAvailable;
    }

    public int getTrainNumber() {
        return trainNumber;
```

```
    }
    public String getDestinationStation() {
        return destinationStation;
    }

    public String getDepartureTime() {
        return departureTime;
    }

    public int getSeatsAvailable() {
        return seatsAvailable;
    }

    public int getDepartureTimeAsTotalMinutes() {
        LocalTime time = LocalTime.parse(departureTime);
        return time.getHour() * 60 + time.getMinute();
    }

    @Override
    public String toString() {
        return "ID: " + id + ", Train Number: " + trainNumber + ", Departure
Station: " + departureStation + ", Destination Station: " +
destinationStation
                + ", Departure Time: " + departureTime + ", Seats Available:
" + seatsAvailable;
    }
}
```

TrainSchedule.java

```
package task02;

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

public class TrainSchedule {
    private List<Train> trains;

    public TrainSchedule() {
        this.trains = new ArrayList<>();
    }

    public void addTrain(Train train) {
        this.trains.add(train);
    }

    public void printSchedule() {
        if (trains.isEmpty()) {
            System.out.println("Train schedule is empty.");
        } else {
            System.out.println("Train Schedule:");
            for (Train train : trains) {
                System.out.println(train);
            }
        }
    }

    public void findTrainByNumber(int trainNumber) {
        boolean found = false;
        for (Train train : trains) {
            if (train.getTrainNumber() == trainNumber) {
                System.out.println("Train found: " + train);
                found = true;
                break;
```

```
            }
        }
        if (!found) {
            System.out.println("Train with number " + trainNumber + " not
found.");
        }
    }

    public void findTrainsByDestination(String destinationStation) {
        List<Train> matchingTrains = new ArrayList<>();
        for (Train train : trains) {
            if
(train.getDestinationStation().equalsIgnoreCase(destinationStation)) {
                matchingTrains.add(train);
            }
        }
        if (matchingTrains.isEmpty()) {
            System.out.println("No trains found for destination: " +
destinationStation);
        } else {
            System.out.println("Trains to " + destinationStation + ":");
            for (Train train : matchingTrains) {
                System.out.println(train);
            }
        }
    }

    public void findTrainsByDestinationAndTime(String destinationStation,
String time) {
        List<Train> matchingTrains = new ArrayList<>();
        for (Train train : trains) {
            if
(train.getDestinationStation().equalsIgnoreCase(destinationStation) &&
                    train.getDepartureTime().compareTo(time) > 0) {
                matchingTrains.add(train);
            }
        }
        if (matchingTrains.isEmpty()) {
            System.out.println("No trains found for destination " +
destinationStation + " after " + time);
        } else {
            System.out.println("Trains to " + destinationStation + " after "
+ time + ":");
            for (Train train : matchingTrains) {
                System.out.println(train);
            }
        }
    }

    public void findTrainsByDestinationAndSeats(String destinationStation,
int seats) {
        List<Train> matchingTrains = new ArrayList<>();
        for (Train train : trains) {
            if
(train.getDestinationStation().equalsIgnoreCase(destinationStation) &&
                    train.getSeatsAvailable() >= seats) {
                matchingTrains.add(train);
            }
        }
        if (matchingTrains.isEmpty()) {
            System.out.println("No trains found for destination " +
destinationStation + " with at least " + seats + " seats available.");
        } else {
            System.out.println("Trains to " + destinationStation + " with at
```

```
least " + seats + " seats available:");
            for (Train train : matchingTrains) {
                System.out.println(train);
            }
        }
    }

    public void checkDepartureInMinutes(int currentTotalMinutes, int minutes)
{
        for (Train train : trains) {
            int departureTotalMinutes =
train.getDepartureTimeAsTotalMinutes();
            if (departureTotalMinutes - currentTotalMinutes == minutes) {
                System.out.println("Train number " + train.getTrainNumber() +
" is departing in " + minutes + " minutes.");
            }
        }
    }
}
```

trains.txt

```
1, 123, Moscow, St. Petersburg, 10:00, 200
2, 456, Moscow, Novosibirsk, 12:30, 150
3, 789, St. Petersburg, Moscow, 11:15, 180
4, 101, Novosibirsk, Moscow, 20:17, 220
5, 111, London, Paris, 08:45, 180
6, 222, New York, Los Angeles, 13:20, 210
7, 333, Sydney, Melbourne, 09:30, 190
8, 444, Tokyo, Kyoto, 16:45, 230
```

Результат программы:

```
Menu:
1. Print train schedule
2. Find train by number
3. Find trains by destination
4. Find trains by destination and time
5. Find trains by destination and seats available
6. Check trains departure in 10, 5, and 3 minutes
7. Exit
```

```
Enter your choice: 1
Train Schedule:
ID: 1, Train Number: 123, Departure Station: Moscow, Destination Station: St. Petersburg, Departure Time: 10:00, Seats Available: 200
ID: 2, Train Number: 456, Departure Station: Moscow, Destination Station: Novosibirsk, Departure Time: 12:30, Seats Available: 150
ID: 3, Train Number: 789, Departure Station: St. Petersburg, Destination Station: Moscow, Departure Time: 11:15, Seats Available: 180
ID: 4, Train Number: 101, Departure Station: Novosibirsk, Destination Station: Moscow, Departure Time: 20:17, Seats Available: 220
ID: 5, Train Number: 111, Departure Station: London, Destination Station: Paris, Departure Time: 08:45, Seats Available: 180
ID: 6, Train Number: 222, Departure Station: New York, Destination Station: Los Angeles, Departure Time: 13:20, Seats Available: 210
ID: 7, Train Number: 333, Departure Station: Sydney, Destination Station: Melbourne, Departure Time: 09:30, Seats Available: 190
ID: 8, Train Number: 444, Departure Station: Tokyo, Destination Station: Kyoto, Departure Time: 16:45, Seats Available: 230
```

```
Enter your choice: 2
Enter train number: 123
Train found: ID: 1, Train Number: 123, Departure Station: Moscow, Destination Station: St. Petersburg, Departure Time: 10:00, Seats Available: 200
```

```
Enter your choice: 3
Enter destination station: Moscow
Trains to Moscow:
ID: 3, Train Number: 789, Departure Station: St. Petersburg, Destination Station: Moscow, Departure Time: 11:15, Seats Available: 180
ID: 4, Train Number: 101, Departure Station: Novosibirsk, Destination Station: Moscow, Departure Time: 20:17, Seats Available: 220
```

```
Enter your choice: 4
Enter destination station: Moscow
Enter time: 20
Trains to Moscow after 20:
ID: 4, Train Number: 101, Departure Station: Novosibirsk, Destination Station: Moscow, Departure Time: 20:17, Seats Available: 220
```

```
Enter your choice: 5
Enter destination station: Moscow
Enter minimum seats available: 200
Trains to Moscow with at least 200 seats available:
ID: 4, Train Number: 101, Departure Station: Novosibirsk, Destination Station: Moscow, Departure Time: 20:17, Seats Available: 220
```

```
Enter your choice: 6
Enter current time in minutes: 790
Train number 222 is departing in 10 minutes.
```

Вывод: Научились создавать и использовать классы в программах на языке программирования Java.