МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчёт
по лабораторной работе №3

Выполнил:
студент группы ПО-9
Ступак Д.Р

Проверил:
Крощенко А. А.

Брест 2024

Цель работы: приобрести базовые навыки работы с файловой системой в Java

Вариант 7

Задание 1

Для класса

• Создать поля классов

• Создать методы классов

• Добавьте необходимые get и set методы (по необходимости)

• Укажите соответствующие модификаторы видимости

• Добавьте конструкторы

• Переопределить методы toString() и equals()

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 (CharacterSet.java)

```
public class CharacterSet {
   private char[] set;
   private int size;
   public CharacterSet(int capacity) {
      set = new char[capacity];
      size = 0;
   }
   public void add(char element) {
      if (!contains(element)) {
         if (size < set.length) {
            set[size++] = element;
         }
      }
   }
   public void remove(char element) {
      for (int i = 0; i < size; i++) {
         if (set[i] == element) {
            set[i] = set[size - 1];
            size--;
            return;
         }
      }
   }
   public boolean contains(char
element) {
      for (int i = 0; i < size; i++) {
         if (set[i] == element) {
            return true;
         }
      }
      return false;
   }
```

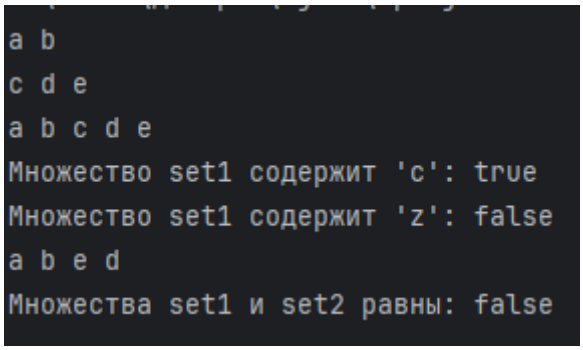
```
    public void union(CharacterSet
otherSet) {
        for (int i = 0; i < otherSet.size; i++) {
            if (!contains(otherSet.set[i])) {
                add(otherSet.set[i]);
            }
        }
    }
    public void print() {
        for (int i = 0; i < size; i++) {
            System.out.print(set[i] + " ");
        }
        System.out.println();
    }
    public boolean equals(CharacterSet
otherSet) {
        if (this.size != otherSet.size) {
            return false;
        }
        for (int i = 0; i < size; i++) {
            if (this.set[i] != otherSet.set[i]) {
                return false;
            }
        }
        return true;
    }
}
```

Код программы (файл Lab3_1.java)

```
public class MainCharacterSet {
    public static void
main(String[] args) {
        CharacterSet set1 = new
CharacterSet(5);
        set1.add('a');
        set1.add('b');
        CharacterSet set2 = new
CharacterSet(5);
        set2.add('c');
        set2.add('d');
        set2.add('e');
        set1.print();
        set2.print();
        set1.union(set2);
        set1.print();

System.out.println("Множество set1
содержит 'c': " +
set1.contains('c'));

System.out.println("Множество set1
содержит 'z': " +
set1.contains('z'));
        set1.remove('c');
        set1.print();

System.out.println("Множества set1
```

```
и set2 равны: " +
                set1.equals(set2));
    }
}
```

Вывод



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 (Node.java)

```
public class Node {
    Train train;
    Node left;
    Node right;

    public Node(Train train) {
        this.train = train;
        this.left = this.right = null;
    }
}
```

Код программы (Train.java)

```
class Train {
    int trainNumber;
    String destination;
    String departureTime;
    int availableSeats;

    public Train(int trainNumber, String destination, String departureTime, int availableSeats) {
```

```
        this.trainNumber = trainNumber;
        this.destination = destination;
        this.departureTime = departureTime;
        this.availableSeats = availableSeats;
    }

    public String toString() {
        return "Train{trainNumber=" + this.trainNumber + ", destination='" + this.destination + "',
departureTime='" + this.departureTime + "', availableSeats=" + this.availableSeats + "}";
    }
}
```

Код программы (TrainSchedule.java)

```
public class TrainSchedule {
    Node root = null;

    public TrainSchedule() {
    }

    public void insert(Train train) {
        this.root = this.insertRec(this.root, train);
    }

    private Node insertRec(Node root, Train train) {
        if (root == null) {
            root = new Node(train);
            return root;
        } else {
            if (train.departureTime.compareTo(root.train.departureTime) < 0) {
                root.left = this.insertRec(root.left, train);
            } else if (train.departureTime.compareTo(root.train.departureTime) > 0) {
                root.right = this.insertRec(root.right, train);
            }

            return root;
        }
    }

    public void printSchedule() {
        this.printInOrder(this.root);
    }

    private void printInOrder(Node root) {
        if (root != null) {
            this.printInOrder(root.left);
            System.out.println(root.train);
            this.printInOrder(root.right);
        }

    }
```

```
    public void findTrainByNumber(int trainNumber) {
        Node result = this.findTrain(this.root, trainNumber);
        if (result == null) {
            System.out.println("Поезд с номером " + trainNumber + " не найден.");
        } else {
            System.out.println("Найден поезд: " + String.valueOf(result.train));
        }

    }

    private Node findTrain(Node root, int trainNumber) {
        if (root != null && root.train.trainNumber != trainNumber) {
            return trainNumber < root.train.trainNumber ? this.findTrain(root.left, trainNumber) :
this.findTrain(root.right, trainNumber);
        } else {
            return root;
        }
    }

    public void printTrainsToDestination(String destination) {
        this.printTrainsToDestination(this.root, destination);
    }

    private void printTrainsToDestination(Node root, String destination) {
        if (root != null) {
            this.printTrainsToDestination(root.left, destination);
            if (root.train.destination.equals(destination)) {
                System.out.println(root.train);
            }

            this.printTrainsToDestination(root.right, destination);
        }

    }

    public void printTrainsToDestinationAfterTime(String destination, String time) {
        this.printTrainsToDestinationAfterTime(this.root, destination, time);
    }

    private void printTrainsToDestinationAfterTime(Node root, String destination, String time) {
        if (root != null) {
            this.printTrainsToDestinationAfterTime(root.left, destination, time);
            if (root.train.destination.equals(destination) && root.train.departureTime.compareTo(time) > 0) {
                System.out.println(root.train);
            }

            this.printTrainsToDestinationAfterTime(root.right, destination, time);
        }

    }
```

```
    public void printTrainsToDestinationWithAvailableSeats(String destination) {
        this.printTrainsToDestinationWithAvailableSeats(this.root, destination);
    }

    private void printTrainsToDestinationWithAvailableSeats(Node root, String destination) {
        if (root != null) {
            this.printTrainsToDestinationWithAvailableSeats(root.left, destination);
            if (root.train.destination.equals(destination) && root.train.availableSeats > 0) {
                System.out.println(root.train);
            }

            this.printTrainsToDestinationWithAvailableSeats(root.right, destination);
        }

    }

    public void notifyTrainDeparture(String time) {
        this.notifyTrainDeparture(this.root, time);
    }

    private void notifyTrainDeparture(Node root, String time) {
        if (root != null) {
            this.notifyTrainDeparture(root.left, time);
            int minutesBeforeDeparture = Integer.parseInt(root.train.departureTime.split(":")[0]) * 60 +
Integer.parseInt(root.train.departureTime.split(":")[1]) - (Integer.parseInt(time.split(":")[0]) * 60 +
Integer.parseInt(time.split(":")[1]));
            if (minutesBeforeDeparture == 3 || minutesBeforeDeparture == 5 || minutesBeforeDeparture ==
10) {
                System.out.println("Оповещение: Поезд " + root.train.trainNumber + " отправляется через " +
minutesBeforeDeparture + " минут");
            }

            this.notifyTrainDeparture(root.right, time);
        }

    }

    public void loadScheduleFromFile(String filename) {
        try {
            BufferedReader reader = new BufferedReader(new FileReader(filename));

            String line;
            while((line = reader.readLine()) != null) {
                String[] parts = line.split(",");
                if (parts.length == 4) {
                    int trainNumber = Integer.parseInt(parts[0]);
                    String destination = parts[1];
                    String departureTime = parts[2];
                    int availableSeats = Integer.parseInt(parts[3]);
                    this.insert(new Train(trainNumber, destination, departureTime, availableSeats));
                }
```

```
        }

        reader.close();
    } catch (IOException var9) {
        System.out.println("Ошибка при загрузке расписания из файла: " + var9.getMessage());
    }

  }
}
```

Код программы (Lab3_2.java) package

```
Lab3_2;

public class Lab3_2 {
    public static void main(String[] args) {
TrainSchedule schedule = new TrainSchedule();
        schedule.loadScheduleFromFile("src/Lab3_2/schedule.txt");
        System.out.println("Вывод всех поездов");
schedule.printSchedule();
        System.out.println("Поиск по номеру");
         schedule.findTrainByNumber(1);
        System.out.println("Вывод по месту прибытия");
schedule.printTrainsToDestination("Krakov ");
        System.out.println("Вывод по месту прибытия после указанного времени");
        schedule.printTrainsToDestinationAfterTime("Krakov ", "15:00");
System.out.println("Вывод по месту прибытия со свободными местами");
schedule.printTrainsToDestinationWithAvailableSeats("Krakov ");
schedule.notifyTrainDeparture("11:20");
    } }
```

Содержимое файла *schedule.txt*

```
1,Brest,10:40,3
5,Krakov,15:40,0
2,Krakov,11:30,5
4,Moscow,11:25,10
```

Вывод

```
Вывод всех поездов
Train{trainNumber=1, destination='Brest', departureTime='10:40', availableSeats=3}
Train{trainNumber=4, destination='Minsk', departureTime='11:25', availableSeats=10}
Train{trainNumber=2, destination='Pakestan', departureTime='11:30', availableSeats=5}
Train{trainNumber=5, destination='Pakestan', departureTime='15:40', availableSeats=0}
Поиск по номеру
Найден поезд: Train{trainNumber=1, destination='Brest', departureTime='10:40', availableSeats=3}
Вывод по месту прибытия
Train{trainNumber=2, destination='Pakestan', departureTime='11:30', availableSeats=5}
Train{trainNumber=5, destination='Pakestan', departureTime='15:40', availableSeats=0}
Вывод по месту прибытия после указанного времени
Train{trainNumber=5, destination='Pakestan', departureTime='15:40', availableSeats=0}
Вывод по месту прибытия со свободными местами
Train{trainNumber=2, destination='Pakestan', departureTime='11:30', availableSeats=5}
Оповещение: Поезд 4 отправляется через 5 минут
Оповещение: Поезд 2 отправляется через 10 минут

Process finished with exit code 0
```