

Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

ОТЧЕТ
По лабораторной работе №3

Выполнил:
Студент 4 курса
группы АС-50
Ольховик И.Ю.
Проверил:
Крощенко А.А.

Брест 2020

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

Задание 1

Реализовать простой класс.

Требования к выполнению

- Реализовать пользовательский класс по варианту.
- Создать другой класс с методом main, в котором будут находиться примеры использования пользовательского класса.

Для каждого 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:

```
package com.company;

public class Main {

    public static void main(String[] args) {
        CharMultitude firstSet = new CharMultitude(10, new
        Character[]{'a', 'a', 'b', 'c', 'a', 'a', 'b', 'c', 'a', 'a', 'a'});
        CharMultitude secondSet = new CharMultitude(10, new
        Character[]{'a', 'd', 'f'});
        System.out.println(firstSet);
        System.out.println(secondSet);
        System.out.println("Объединение множеств");
        System.out.println(CharMultitude.union(firstSet, secondSet));
        System.out.println("Добавление");
        secondSet.add('l');
        System.out.println(secondSet);
        System.out.println("Удаление");
        secondSet.remove('f');
        System.out.println(secondSet);
        CharMultitude firstEq = new CharMultitude(10, new
```

```

Character[]{'a','d','f'});
    CharMultitude secondEq = new CharMultitude(10, new
Character[]{'a','d','c'});
    CharMultitude thirdEq = new CharMultitude(1, new
Character[]{'a','d','f'});
    System.out.println(firstEq.toString() + " == " + secondEq.toString()
+ " " + firstEq.equals(secondEq));
    System.out.println(firstEq.toString() + " == " + thirdEq.toString() +
" " + firstEq.equals(thirdEq));
    }
}

```

```

package com.company;

import java.util.*;

public class CharMultitude {
    private int power;
    private Character [] list;

    public CharMultitude(int power, Character[] list) {
        this.power = power;
        if (power >= list.length)
            this.list = list;
        else {
            String str = String.format("Мощность указана меньше требуемой,
будут использованы символы только до указанной мощности %d, у множества %s",
this.power, Arrays.toString(list));
            System.out.println(str);
            List<Character> buff = new ArrayList<>(Arrays.asList(list));
            for (int i = power; i < list.length; i++)
                buff.remove(power);
            this.list = buff.toArray(new Character[this.power]);
        }
    }

    public void add(char a) {
        boolean full = true;
        Character [] buf = new Character[list.length+1];
        int i = 0;
        if (this.list.length == power) {
            System.out.println("Мощность вашего множества достигла
ограничения, добавить символ нельзя");
        } else {
            for (Character ch : list) {
                buf[i] = ch;
                i++;
            }
            int id = this.list.length;
            buf[id] = a;
            list = buf;
        }
    }

    public void remove(char a) {
        List<Character> buff = new ArrayList<>(Arrays.asList(this.list));
        if (buff.contains(a)) {
            int id = buff.indexOf(a);
            buff.remove(id);
            list = buff.toArray(new Character[this.list.length-1]);
        }
    }
}

```

```

public static String union(CharMultitude a, CharMultitude b) {
    Character[] union = new Character[a.list.length+b.list.length];
    int i=0;
    int j=a.list.length;
    for (Character ch : a.list) {
        union[i]=ch;
        i++;
    }
    for (Character ch : b.list) {
        union[j]=ch;
        j++;
    }
    return Arrays.toString(union);
}

@Override
public boolean equals(Object obj) {
    if (this == obj) return true;
    if (obj == null || getClass() != obj.getClass()) return false;
    CharMultitude that = (CharMultitude) obj;

    if(this.power != that.power)
        return false;
    if (this.list.length != that.list.length)
        return false;
    for (int i=0; i < Math.min(this.list.length,that.list.length); i++) {
        if(this.list[i] != that.list[i])
            return false;
    }
    return true;
}

@Override
public String toString() {
    StringBuilder stringBuilder = new StringBuilder("{}");
    for(Character character : this.list)
        stringBuilder.append(" "+character+ ", ");
    stringBuilder.append("}");
    return stringBuilder.toString();
}
}

```

Скриншоты с результатами работы программы

```

Мощность указана меньше требуемой, будут использованы символы только до указанной мощности 10, у множества [a, a, b, c, a, a, b, c, a, a, a]
{ a, a, b, c, a, a, b, c, a, a, }
{ a, d, f, }
Объединение множеств
[a, a, b, c, a, a, b, c, a, a, a, d, f]
Добавление
{ a, d, f, 1, }
Удаление
{ a, d, 1, }
Мощность указана меньше требуемой, будут использованы символы только до указанной мощности 1, у множества [a, d, f]
{ a, d, f, } == { a, d, c, } false
{ a, d, f, } == { a, } false

```

Задание 2

Разработать автоматизированную систему на основе некоторой структуры данных,

манипулирующей объектами пользовательского класса. Реализовать требуемые функции обработки данных

Требования к выполнению

- Задание посвящено написанию классов, решающих определенную задачу автоматизации;
- Данные для программы загружаются из файла (формат произволен). Файл создать и написать вручную.

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:

```
package com.company;

import java.io.FileNotFoundException;
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.Scanner;
import java.util.Timer;
import java.util.TimerTask;

public class Main {

    public static void main(String[] args) throws FileNotFoundException {
        TrainNotif controller = new TrainNotif();
        Scanner scanner = new Scanner(System.in);
        final SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");

        if(!controller.loadFromFile("timetable.txt"))
        {
            System.out.println("Ошибка чтения данных из файла.");
            return;
        }

        boolean menuKey = true;
```

```

int key;

//для печати информационного сообщения о скором отправлении поезда
Timer timer = new Timer();
timer.schedule(new TimerTask() {
    @Override
    public void run() {
        Timestamp timestamp = new
Timestamp(System.currentTimeMillis());
        String currTime = sdf.format(timestamp);
        controller.printtrainFromTime(currTime);
    }
}, 0, 60*1000);

String toStation, number, fromTime;
while(menuKey)
{
    for(int i=0; i<5; i++)
        System.out.println();
    System.out.println("1 - Вывести данные о всех поездах");
    System.out.println("2 - Вывести информацию о поезде по его
номеру");
    System.out.println("3 - По названию станции назначения выводит
данные обо всех поездах, которые следуют до этой станции");
    System.out.println("4 - Выводит список поездов, следующих до
заданного пункта назначения и отправляющихся после заданного часа");
    System.out.println("5 - Выводит список поездов, отправляющихся до
заданного пункта назначения и имеющих общие места");
    System.out.println("6 - завершить работу");

    key = scanner.nextInt();
    scanner.nextLine();

    if(key==1)
        controller.printTrains();
    else if(key==2)
    {
        System.out.print("Введите номер поезда:");
        number = scanner.nextLine();
        controller.printTrainByNumber(number);
    }
    else if(key==3)
    {
        System.out.print("Введите станцию назначения:");
        toStation = scanner.nextLine();
        controller.printTrainsByToStation(toStation);
    }
    else if(key==4)
    {
        System.out.print("Введите станцию назначения:");
        toStation = scanner.nextLine();
        System.out.print("Введите заданное время (в формате HH:mm,
например 07:10):");
        fromTime = scanner.nextLine();
        controller.printTrainsByToStationAndFromTime(toStation, fromTime);
    }
    else if(key == 5)
    {
        System.out.print("Введите станцию назначения:");
        toStation = scanner.nextLine();
        controller.printTrainsByToStationAndCP(toStation);
    }
}

```

```

        else if(key == 6)
            menuKey= false;
        else
            System.out.println("Неверный код");

        System.out.println("Нажмите Enter для продолжения...");
        scanner.nextLine();
    }
    scanner.close();
    timer.cancel();
}
}

```

```

package com.company;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

class Train {
    private String number;
    private String fromStantion;
    private String toStantion;
    private String fromTime;
    private String toTime;
    private boolean commonPlaces;

    public Train() {
    }

    public void print() {
        if (commonPlaces)
            System.out.println("Номер:" + number + ", от станции: " +
fromStantion + ", до станции: " + toStantion + ", отправление: " + fromTime +
", прибытие: " + toTime + ", наличие общих мест: есть");
        else
            System.out.println("Номер:" + number + ", от станции: " +
fromStantion + ", до станции: " + toStantion + ", отправление: " + fromTime +
", прибытие: " + toTime + ", наличие общих мест: нету");
    }

    public boolean haveCommonPlaces() {
        return commonPlaces;
    }

    public void setCommonPlaces(boolean commonPlaces) {
        this.commonPlaces = commonPlaces;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
        this.number = number;
    }

    public String getFromStantion() {
        return fromStantion;
    }

```

```

    }

    public void setFromStantion(String fromStantion) {
        this.fromStantion = fromStantion;
    }

    public String getToStantion() {
        return toStantion;
    }

    public void setToStantion(String toStantion) {
        this.toStantion = toStantion;
    }

    public String getFromTime() {
        return fromTime;
    }

    public void setFromTime(String fromTime) {
        this.fromTime = fromTime;
    }

    public String getToTime() {
        return toTime;
    }

    public void setToTime(String toTime) {
        this.toTime = toTime;
    }
}

public class TrainNotif {
    private ArrayList<Train> trains = new ArrayList<Train>();
    public boolean loadFromFile(String path) throws FileNotFoundException {
        ArrayList<Train> tempList = new ArrayList<Train>();
        File file = new File(path);

        Scanner scanner = new Scanner(file, "utf-8");

        while(scanner.hasNextLine()) {
            String[] values = scanner.nextLine().split(" ");

            if (values.length != 6)
                return false;

            Train train = new Train();

            try {
                train.setNumber(values[0]);
                train.setFromStantion(values[1]);
                train.setToStantion(values[2]);
                train.setFromTime(values[3]);
                train.setToTime(values[4]);
                train.setCommonPlaces(values[5].equals("Да") ||
values[5].equals("да"));
                tempList.add(train);
            }
            catch(Exception e) {
                return false;
            }
        }

        scanner.close();
    }
}

```



```

        trains = tempList;

        return true;
    }

    public void printTrains()
    {
        for (Train train : trains)
            train.print();
    }

    public void printTrainByNumber(String number) {
        for (Train train : trains)
            if (train.getNumber().equals(number))
                train.print();
    }

    public void printTrainsByToStation(String toStation) {
        for (Train train : trains)
            if (train.getToStantion().equals(toStation))
                train.print();
    }

    public void printTrainsByToStationAndFromTime(String toStation, String
fromTime) {
        String[] fT = fromTime.split(":");
        int fTHour = Integer.parseInt(fT[0]);
        int fTMinutes = Integer.parseInt(fT[1]);
        for (Train train : trains) {
            if (train.getToStantion().equals(toStation)) {
                String[] fTForCheck = train.getFromTime().split(":");
                int fTHourForCheck = Integer.parseInt(fTForCheck[0]);
                int fTMinutesForCheck = Integer.parseInt(fTForCheck[1]);
                if (fTHour < fTHourForCheck) {
                    train.print();
                }
                if (fTHour == fTHourForCheck) {
                    if (fTMinutes < fTMinutesForCheck) {
                        train.print();
                    }
                }
            }
        }
    }

    public void printTrainsByToStationAndCP(String toStation) {
        for (Train train : trains) {
            if (train.getToStantion().equals(toStation)) {
                if (train.haveCommonPlaces())
                    train.print();
            }
        }
    }

    public void printtrainFromTime(String fromTime) {
        String[] fT = fromTime.split(":");
        int fTHour = Integer.parseInt(fT[0]);
        int fTMinutes = Integer.parseInt(fT[1]);
        for (Train train : trains) {
            String[] fTForCheck = train.getFromTime().split(":");
            int fTHourForCheck = Integer.parseInt(fTForCheck[0]);
            int fTMinutesForCheck = Integer.parseInt(fTForCheck[1]);
            if (fTHour == fTHourForCheck) {

```

