МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
Кафедра ИИТ

ОТЧЁТ

по лабораторной работе №3
за 1 семестр 3 курса

Выполнил:
студент группы ПО-9(1)
3 курса
Зейденс Никита
Вячеславович

Проверил:
Крощенко
А. А.

Брест, 2024

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

Вариант 5

Задание 1: -Создать другой класс с методом main, в котором будут находиться примеры использования пользовательского 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 целых чисел ограниченной мощности – Предусмотреть возможность объединения двух множеств, вывода на печать элементов множества, а также 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

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        MySet set = new MySet(1);
        Scanner in = new Scanner(System.in);
        while(true) {
            System.out.print("Choose option:\n" +
                    "\t1. add element\n" +
                    "\t2. remove element\n" +
                    "\t3. add elements (union)\n" +
                    "\t4. search for an element\n" +
                    "\t5. check set equality\n" +
                    "\t6. print set\n" +
                    "\t7. show number of elements\n" +
                    "\t... exit\n");
            int elem = Integer.parseInt(in.next());
            switch (elem) {
                case 1: {
                    System.out.print("Enter a number: ");
                    int temp = Integer.parseInt(in.next());
                    set.add(temp);
                    break;
                }
                case 2: {
                    System.out.print("Enter a number: ");
                    int temp = Integer.parseInt(in.next());
                    set.remove(temp);
                    break;
                }
                case 3: {
                    System.out.print("Enter amount of numbers: ");
```

```
                    int length = Integer.parseInt(in.next());
                    int[] array = new int[length];
                    System.out.print("Enter numbers: ");
                    for(int i = 0; i < length; ++i) {
                        int temp = Integer.parseInt(in.next());
                        array[i] = temp;
                    }
                    set.union(array);
                    break;
                }
                case 4: {
                    System.out.print("Enter a number: ");
                    int temp = Integer.parseInt(in.next());
                    set.print();
                    if(set.contains(temp)) {
                        System.out.print("Set contains " + Integer.toString(temp) +
"\n");
                    }
                    else {
                        System.out.print("Set doesn't contain " + Integer.toString(temp)
+ "\n");
                    }

                    break;
                }
                case 5: {
                    System.out.print("Enter amount of numbers: ");
                    int length = Integer.parseInt(in.next());
                    int[] array = new int[length];
                    System.out.print("Enter numbers: ");
                    for(int i = 0; i < length; ++i) {
                        int temp = Integer.parseInt(in.next());
                        array[i] = temp;
                    }
                    MySet temp = new MySet(array);
                    set.print();
                    temp.print();
                    if(set.equals(temp)) {
                        System.out.print("Sets are equal\n");
                    }
                    else {
                        System.out.print("Sets aren't equal\n");
                    }
                    break;
                }
                case 6: {
                    set.print();
                    break;
                }
                case 7: {
                    System.out.print(String.valueOf(set.getSetCapacity()) + '\n');
                    break;
                }
                default: {
                    return;
                }
            }
        }
    }
}
```

MySet

```
public class MySet {
    private int[] set;
```

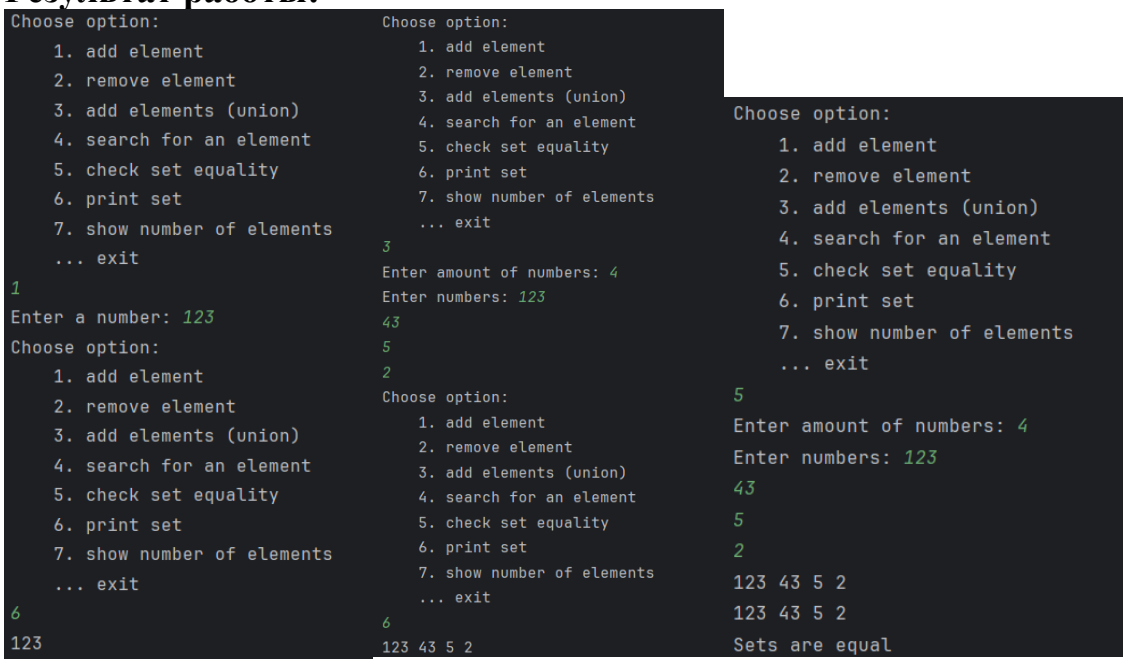
```
private int capacity;
MySet(int setCapacity) {
    this.capacity = 0;
    this.set = new int[setCapacity];
}
MySet(int[] set) {
    this.set = set;
    this.capacity = this.set.length;
}
public int[] getSet() {return this.set;}
public int getSetCapacity() {return this.capacity;}
public void union(int[] set) {
    if(this.isEmpty()) {
        this.set = set;
        this.capacity = this.set.length;
        return;
    }
    for(int i = 0; i < set.length; ++i) {
        this.add(set[i]);
    }
}
public void print() {
    if(this.isEmpty()) {
        return;
    }
    for(int i = 0; i < capacity; ++i) {
        System.out.print(String.valueOf(this.set[i]) + ' ');
    }
    System.out.print("\n");
}
public boolean contains(int elem) {
    if(this.isEmpty()) {
        return false;
    }
    for(int i = 0; i < capacity; ++i) {
        if(this.set[i] == elem) {
            return true;
        }
    }
    return false;
}
public void add(int elem) {
    if(this.contains(elem)) {
        return;
    }
    if(this.capacity == this.set.length) {
        int[] tempSet = new int[this.set.length * 2];
        for(int i = 0; i < this.capacity; ++i) {
            tempSet[i] = this.set[i];
        }
        tempSet[this.capacity] = elem;
        this.set = tempSet;
        ++this.capacity;
    }
    else {
        this.set[this.capacity] = elem;
        ++this.capacity;
    }
}
public void remove(int elem) {
    for (int i = 0; i < this.set.length; ++i) {
        if (this.set[i] == elem) {
            for (int j = i + 1; j < this.capacity; ++j) {
                this.set[j - 1] = this.set[j];
            }
```

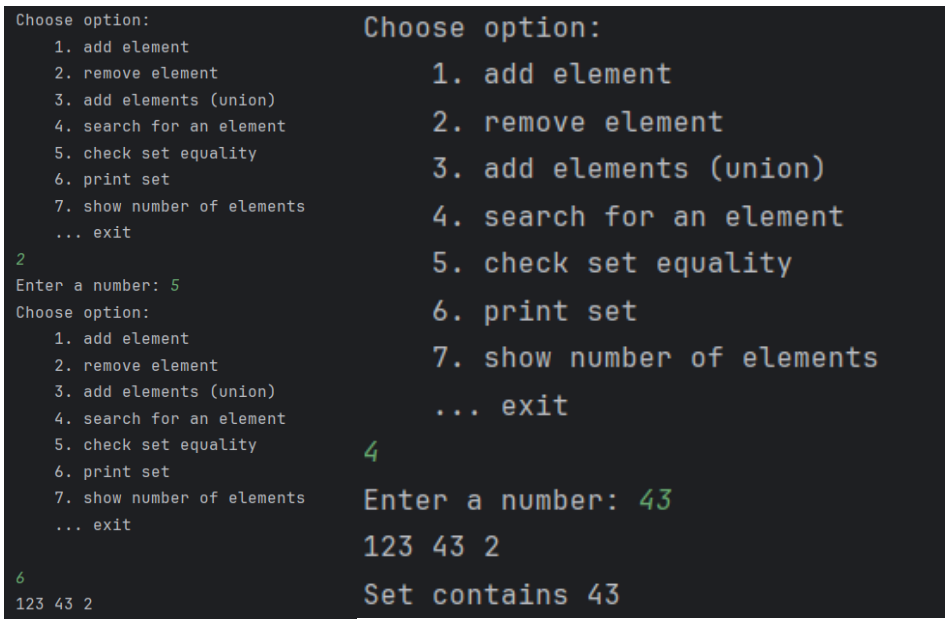
```
                --this.capacity;
                return;
            }
        }

    }
    public boolean equals(MySet set) {
        if(this.capacity != set.capacity) {
            return false;
        }
        for(int i = 0; i < this.capacity; ++i) {
            if(this.set[i] != set.set[i]) {
                return false;
            }
        }
        return true;
    }
    public boolean isEmpty() {
        return this.capacity == 0;
    }

}
```

Результат работы:





Задание 2: Разработать автоматизированную систему на основе некоторой структуры данных, манипулирующей объектами пользовательского класса. Реализовать требуемые функции обработки данных.

Составить программу, которая моделирует заполнение гибкого диска (1440 Кб). В процессе работы файлы могут записываться на диск и удаляться с него. С каждым файлом (File) ассоциированы следующие данные: • Размер • Расширение • Имя файла • Как файлы могут трактоваться и директории, которые в свою очередь содержат другие файлы и папки. Если при удалении образовался свободный участок, то вновь записываемый файл помещается на этом свободном участке, либо, если он не помещается на этом участке, то его следует разместить после последнего записанного файла. Если файл превосходит длину самого большого участка, выдается аварийное сообщение. Рекомендуется создать список свободных участков и список занятых участков памяти на диске.

Код программы:

File

```
public class File {
private final String diskName = "D:";
private double size;
private String extension;
private String fileName;
private String directoryName;
private String directoryPath;
private String filePath;
File() {
    this.size = 0;
    this.extension = "";
    this.fileName = "";
    this.directoryName = "";
    this.directoryPath = "";
    this.filePath = "";
```

```
    }
    File(double size, String extension, String fileName, String directoryName) {
        this.size = size;
        this.extension = extension;
        this.fileName = fileName;
        this.directoryName = directoryName;
        this.directoryPath = diskName + "/" + directoryName;
        this.filePath = this.directoryPath + "/" + this.fileName + "." + this.extension;
    }
    public void setSize(double size) {this.size = size;}
    public void setExtension(String extension) {this.extension = extension;}
    public void setFileName(String fileName) {this.fileName = fileName;}
    public void setDirectoryName(String directoryName) {this.directoryName =
directoryName;}
    public void setDirectoryPath(String directoryPath) {this.directoryPath =
directoryPath;}
    public void setFilePath(String filePath) {this.filePath = filePath;}
    public String getDiskName() {return diskName;}
    public double getSize() {return size;}
    public String getExtension() {return extension;}
    public String getFileName() {return fileName;}
    public String getDirectoryName() {return directoryName;}
    public String getDirectoryPath() {return directoryPath;}
    public String getFilePath() {return filePath;}
}
```

Main

```
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Random;

public class Main {
    final static int DISK_SPACE = 1440;//Килобайт
    final static int NUMBER_OF_LOCATIONS = 12;//Количество участков
    final static int LOCATION_SIZE = DISK_SPACE / NUMBER_OF_LOCATIONS;//Размер участка
    static int lastDeletedFileInLocation = 0;//Изначально модифицированный участок -
первый
    static int lastAddedFileInLocation = 0;//Изначально модифицированный участок - первый
    public static void deleteFile(ArrayList<ArrayList<File>> drive) {
        if(isDriveEmpty(drive)) {
            System.out.print("Drive is empty, delete drive instead (-. -)\n");
            return;
        }
        Scanner in = new Scanner(System.in);
        while(true) {
            printDrive(drive);
            System.out.print("Choose option:\n" +
                    "\t1. delete file\n" +
                    "\t... exit\n");
            int option = Integer.parseInt(in.next());
            switch (option) {
                case 1: {
                    System.out.print("Enter location number: ");
                    int location = Integer.parseInt(in.next()) - 1;
                    System.out.print("Enter file number: ");
                    int fileNumber = Integer.parseInt(in.next()) - 1;
                    File file = drive.get(location).remove(fileNumber);
                    if(file != null) {
                        System.out.print("File with path: " + file.getFilePath() + "\nWas
deleted successfully\n");
                        lastDeletedFileInLocation = location;
                        return;
                    }
                    System.out.print("File wasn't deleted\n");
                }
                default: {
```

```
                    return;
                }
            }
        }
    }
    public static void addFile(ArrayList<ArrayList<File>> drive, File file) {
        double[] freeSpaceOfLocations = new double[drive.size()];
        double maxFreeSpaceOfLocation = -1;
        for(int i = 0; i < drive.size(); ++i) {
            double usedSpaceOfLocation = 0;
            for(int j = 0; j < drive.get(i).size(); ++j) {
                usedSpaceOfLocation += drive.get(i).get(j).getSize();
            }
            freeSpaceOfLocations[i] = LOCATION_SIZE - usedSpaceOfLocation;
            if(maxFreeSpaceOfLocation < freeSpaceOfLocations[i]) maxFreeSpaceOfLocation =
freeSpaceOfLocations[i];
        }
        //Файл некуда добавить
        if(file.getSize() > maxFreeSpaceOfLocation) {
            System.out.print("File is too large for this drive. Clear the disk space\n");
            return;
        }
        //Добавление файла в свободный участок после последнего удалённого файла
        if(file.getSize() <= freeSpaceOfLocations[lastDeletedFileInLocation]) {
            System.out.print("Added via last deleted\n");
            drive.get(lastDeletedFileInLocation).add(file);
            lastAddedFileInLocation = lastDeletedFileInLocation;
            return;
        }
        //Добавление файла в свободный участок после последнего добавленного файла
        if(file.getSize() <= freeSpaceOfLocations[lastAddedFileInLocation]) {
            System.out.print("Added via last added\n");
            drive.get(lastAddedFileInLocation).add(file);
            return;
        }
        //Добавление в первый попавшийся свободный участок, если предыдущие не отработали
        for(int i = 0; i < freeSpaceOfLocations.length; ++i) {
            if(file.getSize() <= freeSpaceOfLocations[i]) {

                drive.get(i).add(file);
                lastAddedFileInLocation = i;
                //System.out.print("Added via via: " + lastAddedFileInLocation + "\n");
                return;
            }
        }
    }
    public static ArrayList<ArrayList<File>> createDrive() {
        ArrayList<ArrayList<File>> drive = new ArrayList<>();
        for(int i = 0; i < NUMBER_OF_LOCATIONS; ++i) {
            ArrayList<File> subDrive = new ArrayList<>();
            drive.add(subDrive);
        }
        return drive;
    }

    public static ArrayList<File> createFiles() {
        Scanner in = new Scanner(System.in);
        final String[] directoryNames = {"Проекты", "Финансы", "Маркетинг", "Кадры",
"Архив"};
        final String[] fileNames = {"Лев", "Собака", "Кошка", "Зебра", "Пингвин"};
        final String[] fileExtensions = {"docx", "jpg", "mp3", "pdf", "txt"};
        System.out.print("Введите количество файлов, которые вы хотите добавить в
файловую систему: ");
        int numberOfFiles = Integer.parseInt(in.next());
        ArrayList<File> files = new ArrayList<>();
```

```
        Random random = new Random();
        for(int i = 0; i < numberOfFiles; ++i) {
            double size = random.nextInt(LOCATION_SIZE - 1) + 1;//1-119
            String fileExtension = fileExtensions[random.nextInt(fileExtensions.length)];
            String fileName = fileNames[random.nextInt(fileNames.length)];
            String directoryName = directoryNames[random.nextInt(directoryNames.length)];
            files.add(new File(size, fileExtension, fileName, directoryName));
        }
        for(int i = 0; i < files.size(); ++i) {
            System.out.print(files.get(i).getFilePath() + "\t" + files.get(i).getSize() +
"\n");
        }
        return files;
    }

    public static void printDrive(ArrayList<ArrayList<File>> drive) {
        if(isDriveEmpty(drive)) {
            System.out.print("Drive is empty, check the FONT!\n");
            return;
        }
        for(int i = 0; i < drive.size(); ++i) {
            double freeSpaceOfLocations = LOCATION_SIZE;
            System.out.print("Location " + (i + 1) + ":");
            for(int j = 0; j < drive.get(i).size(); ++j) {
                if (j == 0) {
                    System.out.print("\n\tSize:\t\tPath to file:\n");
                }
                double size = drive.get(i).get(j).getSize();
                String filePath = drive.get(i).get(j).getFilePath();
                freeSpaceOfLocations -= size;
                System.out.print((j + 1) + ".\t" + size + "\t\t" + filePath + "\n");
            }
            System.out.print("Total size: " + freeSpaceOfLocations + "\n");
        }
    }

    public static void feelDrive(ArrayList<ArrayList<File>> drive, ArrayList<File> files)
{
        for(int i = 0; i < files.size(); ++i) {
            addFile(drive, files.get(i));
        }
    }

    public static boolean isDriveEmpty(ArrayList<ArrayList<File>> drive) {
        for(int i = 0; i < drive.size(); ++i) {
            if(!drive.get(i).isEmpty()) {
                return false;
            }
        }
        return true;
    }

    public static void simulateFileSystem() {
        ArrayList<ArrayList<File>> drive = createDrive();
        Scanner in = new Scanner(System.in);
        while(true) {
            System.out.print("Choose option: \n" +
                    "\t1. add files\n" +
                    "\t2. delete files\n" +
                    "\t3. print drive\n" +
                    "\t4. last modified location\n" +
                    "\t... exit\n");
            int option = Integer.parseInt(in.next());
            switch(option) {
                case 1: {
```

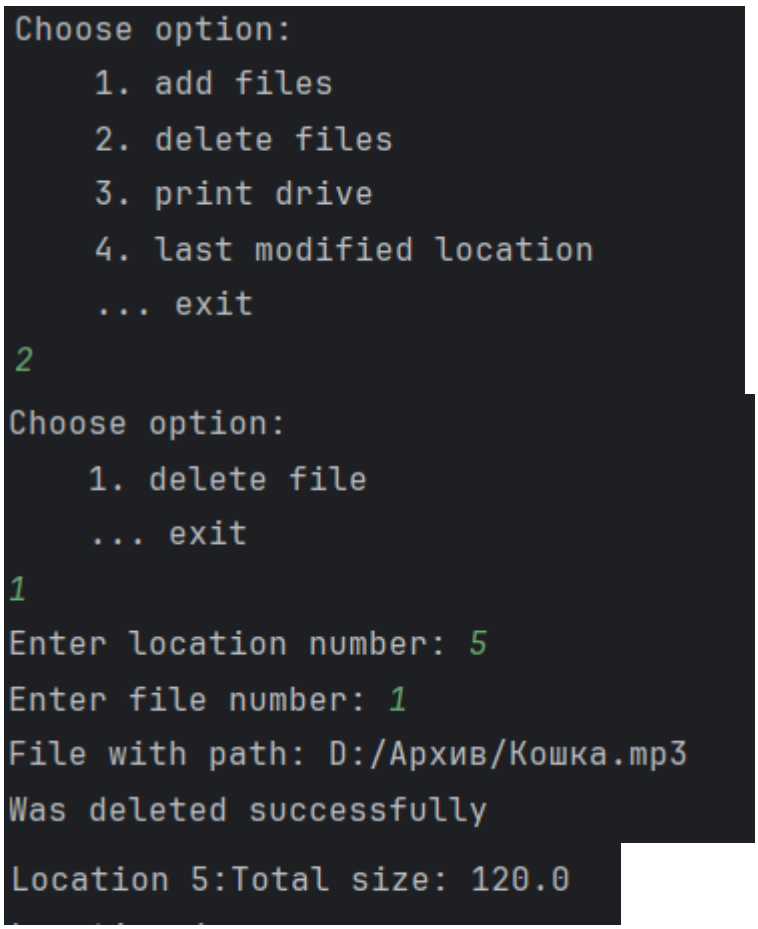
```
                    ArrayList<File> files = createFiles();
                    feelDrive(drive, files);
                    break;
                }
                case 2: {
                    deleteFile(drive);
                    break;
                }
                case 3: {
                    printDrive(drive);
                    break;
                }
                case 4: {
                    System.out.printf("Lastly added to location, number: " +
(lastAddedFileInLocation + 1) +"\n" +
                            "Lastly deleted in location, number: " +
(lastDeletedFileInLocation + 1) + "\n1");
                    break;
                }
                default: {
                    return;
                }
            }
        }
    }

    public static void main(String[] args) {
        simulateFileSystem();
    }
}
```

Результат работы:

```
Choose option:
    1. add files
    2. delete files
    3. print drive
    4. last modified location
    ... exit
1
Введите количество файлов, которые вы хотите добавить в файловую систему: 12
D:/Кадры/Лев.pdf    27.0
D:/Маркетинг/Зебра.pdf  71.0
D:/Кадры/Зебра.mp3  60.0
D:/Маркетинг/Зебра.jpg  20.0
D:/Кадры/Зебра.jpg  78.0
D:/Архив/Лев.pdf    18.0
D:/Кадры/Зебра.pdf  3.0
D:/Финансы/Собака.jpg   60.0
D:/Маркетинг/Кошка.txt  48.0
D:/Проекты/Лев.txt  69.0
D:/Архив/Кошка.mp3  117.0
D:/Проекты/Собака.docx  48.0
Added via last deleted
Added via last deleted
Added via last deleted
Added via last added
Added via last added
Added via last added
```

```
Choose option:
    1. add files
    2. delete files
    3. print drive
    4. last modified location
    ... exit
3
Location 1:
    Size:       Path to file:
1.  27.0        D:/Кадры/Лев.pdf
2.  71.0        D:/Маркетинг/Зебра.pdf
3.  20.0        D:/Маркетинг/Зебра.jpg
Total size: 2.0
Location 2:
    Size:       Path to file:
1.  60.0        D:/Кадры/Зебра.mp3
2.  60.0        D:/Финансы/Собака.jpg
Total size: 0.0
Location 3:
    Size:       Path to file:
1.  78.0        D:/Кадры/Зебра.jpg
2.  18.0        D:/Архив/Лев.pdf
3.  3.0     D:/Кадры/Зебра.pdf
Total size: 21.0
Location 4:
    Size:       Path to file:
1.  48.0        D:/Маркетинг/Кошка.txt
2.  69.0        D:/Проекты/Лев.txt
Total size: 3.0
Location 5:
    Size:       Path to file:
1.  117.0       D:/Архив/Кошка.mp3
Total size: 3.0
Location 6:
    Size:       Path to file:
1.  48.0        D:/Проекты/Собака.docx
Total size: 72.0
Location 7:Total size: 120.0
Location 8:Total size: 120.0
Location 9:Total size: 120.0
Location 10:Total size: 120.0
Location 11:Total size: 120.0
Location 12:Total size: 120.0
```



Вывод: Базовые навыки работы с файловой системой в Java были приобретены.