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

ОТЧЕТ
Лабораторная работа №3

Выполнила:
Студентка 4 курса
Группы АС-50
Клиницкая Р.П.
Проверил:
Крощенко А.А.

Брест 2020

Цель работы:

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

Задание. Вариант 5.

1. 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:

```
import  java.io.*;
import java.lang.*;
class CustomArray {
    int power, pos;
    int[] customArray;

    //зададим конструктору изначальную инициализацию множества
    CustomArray(int power) {
        this.power = power;
        customArray = new  int[power];
        for (int i = 0; i < power; i++)
            customArray[i] = 0;
    }
    CustomArray(int power, int numb) {
        this.power = power;
        customArray = new  int[power];
        for(int i = 0; i < power; i++)
            customArray[i] = numb;
    }

    //создадим метод вывода множества на экран
    public void see() {
        System.out.print("\nArray power is: "+power+"\nArray consists of: ");
        for(int i = 0; i < power - 1; i++)
            System.out.print("["+customArray[i]+"], ");
        System.out.print("["+customArray[power - 1]+"] elements");
    }

    //определим методы добавления, получения и удаления элемента в множество
и из него
    public void setEl(int position, int number) {
        this.pos = position;
        customArray[pos] = number;
    }
    public int getEl(int pos)
    {
        return customArray[pos];
    }
    public int getPos(int numb){
        for(int i = 0; i < power; i++){
            if(customArray[i] == numb) this.pos = i;
        }
```

```
        return pos;
    }

    //удаление элемента как по заданной позиции, так и по конкретному числу
    public void deleteElPos(int position){
        for(int i = position; i < power - 1; i++){
            customArray[position] = customArray[position + 1];
        }
        this.power--;
    }

    public void deleteElNumb(int numb){
        for(int i = 0; i < power; i++) {
            if (customArray[i] == numb) this.pos = i;
        }
        for(int i = pos; i < power - 1; i++){
            customArray[pos] = customArray[pos + 1];
        }
        this.power--;
    }

    //проверка наличия элемента в массиве
    public boolean isExist(int numb)
    {
        for(int i = 0; i < power; i++){
            if(customArray[i] == numb) return true;
        }
        return false;
    }

    //сравнение мощностей множеств. 1 - больше, 0 - равны, -1 - меньше
    public int compareArr(CustomArray customArray){
        if(this.power > customArray.power) return 1;
        if(this.power < customArray.power) return -1;
        return 0;
    }

    //сравнение элементов множеств.
    public int compareElArr(CustomArray customArray){
        if(this.power != customArray.power)
            return -1;
            for(int i = 0; i < power; i++){
                if(this.customArray[i] != customArray.getEl(i))
                    return -1;
            }
        return 1;
    }
}
//демонстрация работы пользовательского класса
public class task1{
    public static void main(String[] args)
    {
        CustomArray myArray1 = new CustomArray(5);
        myArray1.see();
        CustomArray myArr2 = new CustomArray(5,2);
        myArr2.see();
        myArr2.setEl(1, 4);
        System.out.println("\nDoes numb 4 exist in this array?
"+myArr2.isExist(4));
        myArr2.deleteElNumb(4);
        myArr2.see();
        System.out.println("\nDoes first array have more power than second
for now? " + myArray1.compareArr(myArr2));
        myArray1.setEl(4,3);
```

```
        System.out.println("Which pos number 3 is on? " +
myArray1.getPos(3));
        CustomArray testarr = new CustomArray(5);
        myArray1.see();
        testarr.see();
        System.out.println("\nCompare elemets of 2 arrays:
"+testarr.compareElArr(myArray1));
    }
}
```

Скриншоты:

```
G:\SSP>cd var5

G:\SSP\var5>cd Lab3_var5_task1

G:\SSP\var5\Lab3_var5_task1>java task1

Array power is: 5
Array consists of: [0], [0], [0], [0], [0] elements
Array power is: 5
Array consists of: [2], [2], [2], [2], [2] elements
Does numb 4 exist in this array? true

Array power is: 4
Array consists of: [2], [2], [2], [2] elements
Does first array have more power than second for now? 1
Which pos number 3 is on? 4

Array power is: 5
Array consists of: [0], [0], [0], [0], [3] elements
Array power is: 5
Array consists of: [0], [0], [0], [0], [0] elements
Compare elemets of 2 arrays: -1

G:\SSP\var5\Lab3_var5_task1>
```

2. Моделирование файловой системы. 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:

```
package com.company;
import java.io.*;
import java.util.*;
```

```
//сразу сделаем классы для отлова ошибок и исключений, навроде отсутствия
места на "диске"
//ошибка: файл не содержит данных (0 кб)
class FileEmptyException extends Exception {
}
//ошибка: файл не найден
class FileExistsException extends Exception {
}
//ошибка: директория не найдена
class NoDirectoryException extends Exception {
}
//ошибка: не хватает места на диске
class NoPlaceException extends Exception {
}
//создадим отдельный класс для создания хранилища.
class CustomDisk {
    //по условию хранилище в 1440 Кб
    int diskSize = 1440000;
    //создаем сам "диск"
    short[] storageData = new short[diskSize];
    //создаем списки для доступа к информации о сегментах и файлах
    ArrayList<FileInfo> files = new ArrayList<FileInfo>();

    ArrayList<MemorySegment> freeSegments = new ArrayList<MemorySegment>();
    ArrayList<MemorySegment> occupiedSegments = new
ArrayList<MemorySegment>();

    public CustomDisk()
    {
        //добавляем корневую папку
        FileInfo root = new FileInfo();
        root.setIsDirectory(true);
        root.setName("root");
        files.add(root);

        //при инициализации все сегменты свободны. добавляем в них 1440 Кб
        MemorySegment segment = new MemorySegment();
        segment.setStartIndex(0);
        segment.setEndIndex(diskSize -1);
        freeSegments.add(segment);
    }
    //метод для записи файла
    public void writeFile(String path, String directory) throws Exception {
        //проверяем куда проводится запись
        if (!directory.equals("root")) {
            //смотрим, существует ли уже эта директория
            boolean isFind = false;
            for (int i = 0; i < files.size(); i++)
                if (files.get(i).isDirectory() &&
files.get(i).getName().equals(directory)) {
                    isFind = true;
                    break;
                }
            //если директория не найдена выдаем исключение
            if (!isFind)
                throw new NoDirectoryException();
        }

        File file = new File(path);
        int sizeOfFile = (int) file.length();
        int segmentForWrite = -1;
        //проверяем: существование файла
        if (!file.exists())
            throw new FileNotFoundException();
```

```
        //помещается ли он на "диск"
        if (sizeOfFile > diskSize)
            throw new NoPlaceException();
        //если он пустой (0 Кб) выдаем исключение
        if (sizeOfFile == 0)
            throw new FileEmptyException();

        //просматриваем свободные сегменты и выбираем тот, в котором
достаточно места для записи
        for (int i = 0; i < freeSegments.size(); i++)
            if (freeSegments.get(i).getSize() >= sizeOfFile) {
                segmentForWrite = i;
                break;
            }
        //если не удалось найти подходящий сегмент для данного файла
        if (segmentForWrite == -1)
            throw new NoPlaceException();

        //иначе считываем информацию из файла в буффер
        short[] fileData = new short[sizeOfFile];

        Scanner scanner = new Scanner(file);

        for (int i = 0; scanner.hasNextByte(); i++)
            fileData[i] = scanner.nextByte();

        //записываем в класс FileInfo информацию по файлу
        FileInfo fileInfo = new FileInfo();
        fileInfo.setSize(sizeOfFile);
        //делим путь файла построчно на каждом символе "/"
        String[] dividedPath = path.split("/");

        //с последней строки таким же образом получаем расширение файла
        String[] resExtension = dividedPath[dividedPath.length -
1].split("\\.");

        String fileName, fileExtension;

        //если разбить не удалось, у файла нет расширения
        if (resExtension.length == 0) {
            fileName = path;
            fileExtension = null;
        }
        //иначе оформляем красивое имя файла "*.extension"
        else {
            fileExtension = resExtension[resExtension.length - 1];
            resExtension[resExtension.length - 1] = "";
            fileName = String.join(".", resExtension);
        }

        //проверяем, существует ли такой файл
        for (int i = 0; i < files.size(); i++){
            if (!files.get(i).isDirectory()
                    && files.get(i).getPath().equals(directory)
                    && files.get(i).getName().equals(fileName)
                    && files.get(i).getExtension().equals((fileExtension)))
            {
                throw new FileExistsException();
            }
        }
        //если же он существует, то мы дозаполняем информацию по файлу
        fileInfo.setName(fileName);
        fileInfo.setExtension(fileExtension);
        fileInfo.setPath(directory);
```

```
        //получаем записываемый сегмент
        MemorySegment memorySegment = freeSegments.get(segmentForWrite);
        //редактируем список свободных сегментов.
        freeSegments.remove(segmentForWrite);
        int startIndex = memorySegment.getStartIndex();
        //если выделенный сегмент = объему файла, то сразу добавляем его в
занятые сегменты
        if(memorySegment.getSize() == sizeOfFile)
            occupiedSegments.add(memorySegment);
            //если он больше длины файла, то происходит сегментация
        else
        {
            //отделяем свободную часть сегмента и вовзращаем в список
свободных
            MemorySegment occupiedSegment = memorySegment.clone();
            memorySegment.setStartIndex(startIndex + sizeOfFile);
            freeSegments.add(memorySegment);
            unionFreeNeighbors();
            //добавляем в список занятых сегментов часть, в которую будем
писать файл
            occupiedSegment.setEndIndex(startIndex + sizeOfFile - 1);
            occupiedSegments.add(occupiedSegment);
        }

        fileInfo.setStartIndex(startIndex);

        //добавляем в список файлов
        files.add(fileInfo);

        //записываем непосредственно данные
        for(int i = startIndex; i<startIndex+ sizeOfFile; i++)
            storageData[i] = fileData[i-startIndex];

    }

    public void createDirectory(String parentPath, String name) throws
Exception
    {
        //если такая директория существует
        for(int i=0; i<files.size(); i++)
            if(files.get(i).isDirectory() &&
files.get(i).getPath().equals(parentPath+"/"+name))
                throw new FileExistsException();

        FileInfo fileInfo = new FileInfo();
        fileInfo.setName(parentPath+"/"+name);
        fileInfo.setIsDirectory(true);

        files.add(fileInfo);
    }

    public void unionFreeNeighbors()
    {
        //начинаем поиск свободного сегмента с остальными свободными рядом
        for(int i=0; i<freeSegments.size(); i++)
        {
            for(int j=0; j<freeSegments.size(); j++)
            {
                //если нашлись, то объединяем их
                if(i!=j && freeSegments.get(i).getEndIndex() ==
freeSegments.get(j).getStartIndex()-1)
                {
                    //создаем объект класса сегменты памяти и задаем ему
```

```
индексы.
                    MemorySegment segment = new MemorySegment();

segment.setStartIndex(freeSegments.get(i).getStartIndex());
                    segment.setEndIndex(freeSegments.get(j).getEndIndex());
                    //т.к. индексы меняются после удаления первого элемента
                    int startIndexRemoved =
freeSegments.get(j).getStartIndex();
                    //и, соответственно, очищаемся от данного сегмента
                    freeSegments.remove(i);
                    for(int k=0; k<freeSegments.size(); k++)

if(freeSegments.get(k).getStartIndex()==startIndexRemoved) {
                            freeSegments.remove(k);
                            break;
                        }
                    freeSegments.add(segment);
                        // и создаем рекурсию, чтобы точно пройтись по
каждому сегменту.
                    unionFreeNeighbors();
                    return;
                }
            }
        }
    }
    //создаем метод для удаления файла
    public void removeFile(String path) throws Exception
    {
        //по такому же алгоритму получаем имя файла, его расширение и путь
        String[] splitted = path.split("/");
        String fileName = splitted[splitted.length-1];
        String fileExtension = "";
        String[] splitextension = fileName.split("\\.");

        if(splitextension.length!=0)
        {
            fileExtension = splitextension[splitextension.length-1];
            splitextension[splitextension.length-1] = "";
            fileName = String.join(".", splitextension);
        }

        splitted[splitted.length-1] = "";

        String filePath = String.join("/", splitted);
        filePath = filePath.substring(0, filePath.length()-1);

        boolean removed = false;

        //ищем файл и если нашли, удаляем. Т.е. меняем занятые в свободные
сегменты и
        //сразу же вызываем метод на объединение соседних областей
        for(int i=0; i<files.size(); i++)
            if(files.get(i).getName().equals(fileName) &&
files.get(i).getPath().equals(filePath)
                    && files.get(i).getExtension().equals(fileExtension)
&&!files.get(i).isDirectory())
            {
                for(int j=0; j<occupiedSegments.size(); j++)

if(occupiedSegments.get(j).getStartIndex()==files.get(i).getStartIndex())
                    {
                        MemorySegment segment =
occupiedSegments.get(j).clone();
                        occupiedSegments.remove(j);
```

```
                        freeSegments.add(segment);
                        unionFreeNeighbors();

                        break;
                    }
                //удаляем файл
                files.remove(i);
                removed = true;
                break;
            }
        //если вдруг удаление не удалось
        if(!removed)
            throw new FileNotFoundException();
    }
    //аналогично создаем метод удаления директории
    public void removeDirectory(String path){

        if(path.equals("root")||path.equals("root/"))
            return;

        for(int i=0; i<files.size(); i++)
        {
            if(!files.get(i).isDirectory())
            {
                //если файл, ищем совпадения по пути
                if(files.get(i).getPath().indexOf(path)==0)
                {
                    for(int j=0; j<occupiedSegments.size(); j++)

if(occupiedSegments.get(j).getStartIndex()==files.get(i).getStartIndex())
                        {
                            MemorySegment segment =
occupiedSegments.get(j).clone();
                            occupiedSegments.remove(j);
                            freeSegments.add(segment);
                            unionFreeNeighbors();
                            break;
                        }

                    files.remove(i);
                    i=0;
                }
            }
            else
            {
                //если папка, ищем совпадения по имени
                if(files.get(i).getName().indexOf(path)==0)
                {
                    files.remove(i);
                    i=0;
                }
            }
        }
    }

    public void showFiles(){
        for(int i=0; i<files.size(); i++)
            System.out.println(files.get(i).toString());
    }

    public void showFreeSegments(){
        for(int i=0; i<freeSegments.size(); i++)
            System.out.println(freeSegments.get(i).toString());
    }
```

```
    public void showOccupiedSegments(){
        for(int i=0; i<occupiedSegments.size(); i++)
            System.out.println(occupiedSegments.get(i).toString());
    }

}

//для удобства работы с сегментами диска выделим для них отдельный класс
class MemorySegment {
    //зададим область в памяти
    private int startIndex;
    private int endIndex;
    //определим методы получения индексов этой области и их задания
    public int getStartIndex() {
        return startIndex;
    }

    public void setStartIndex(int startIndex) {
        this.startIndex = startIndex;
    }

    public int getEndIndex() {
        return endIndex;
    }

    public void setEndIndex(int endIndex) {
        this.endIndex = endIndex;
    }
    //метод  для получения размера сегмента в Кб
    public int getSize(){
        return endIndex-startIndex;
    }
    //для объединения сегментов, перебрасывания их из свободных в занятые
    //создадим метод копирования
    @Override
    public MemorySegment clone(){
        MemorySegment segment = new MemorySegment();
        segment.setStartIndex(startIndex);
        segment.setEndIndex(endIndex);
        return segment;
    }
    @Override
    public String toString(){
        return "startIndex="+startIndex+"; endIndex="+endIndex;
    }
}

//для документирования, удобного вывода, удобства работы
class FileInfo {
    //определяем основные поля файла
    private String path = "";
    private String name = "";
    //т.к. должен содержать размер, то нет смысла задавать и endIndex
    private int startIndex = -1;
    private int size = -1;
    private String extension = "";
    private boolean isDirectory = false;
    //узнать или задать размер
    public int getSize() {
        return size;
    }

    public void setSize(int size) {
```

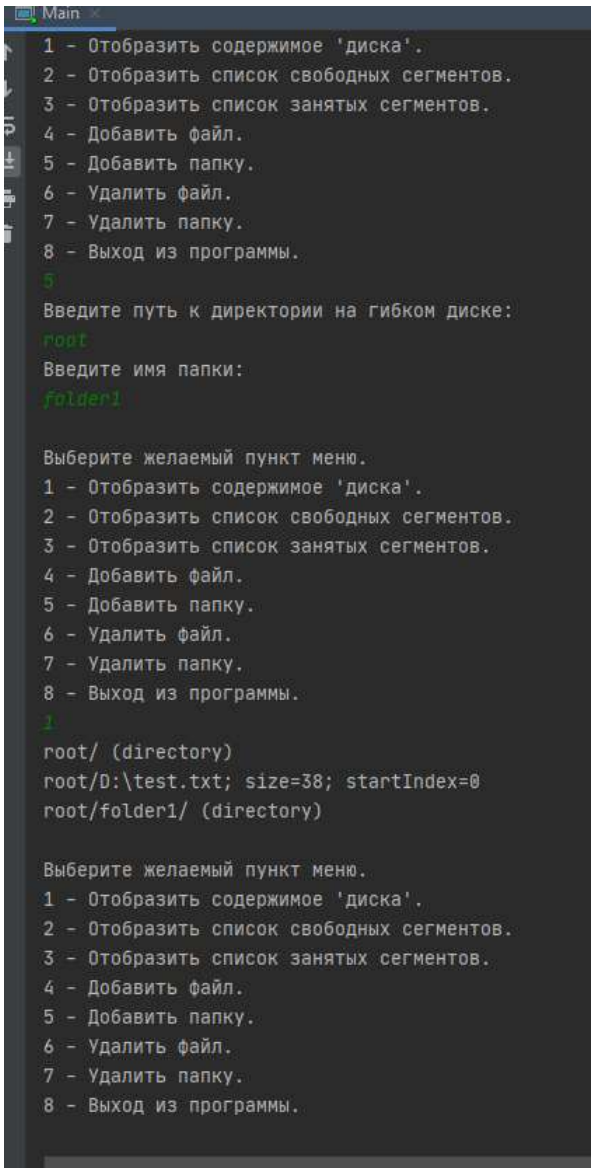
```
        this.size = size;
    }
    //узнать или задать имя
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
    //узнать или задать индекс начала файла
    public void setStartIndex(int startIndex) {
        this.startIndex = startIndex;
    }

    public int getStartIndex() {
        return startIndex;
    }
    //узнать или задать расширение файла
    public String getExtension() {
        return extension;
    }

    public void setExtension(String extension) {
        this.extension = extension;
    }
    //узнать или задать индентификатор, является ли это папкой или файлом
    public boolean isDirectory() {
        return isDirectory;
    }

    public void setIsDirectory(boolean isDirectory) {
        this.isDirectory = isDirectory;
    }
    //задать или получить путь к файлу
    public String getPath() {
        return path;
    }

    public void setPath(String path) {
        this.path = path;
    }
    //вывести имеющиеся данные на экран
    public String toString(){
        if(!isDirectory)
            return path+"/"+name+extension+"; size="+size+";
startIndex="+startIndex;
        else
            return name+"/"+ " (directory)";
    }
}

//пример работы "диской" системы
public class Main {

    public static void main(String[] args) {
        CustomDisk customDisk = new CustomDisk();

        boolean iswork = true;
        byte code;
        Scanner scanner = new Scanner(System.in);
        //программа продолжает работу, пока пользователь не захочет выйти
        while(iswork) {
            System.out.println();
```

```
            System.out.println("Выберите желаемый пункт меню.");
            System.out.println("1 - Отобразить содержимое 'диска'.");
            System.out.println("2 - Отобразить список свободных сегментов.");
            System.out.println("3 - Отобразить список занятых сегментов.");
            System.out.println("4 - Добавить файл.");
            System.out.println("5 - Добавить папку.");
            System.out.println("6 - Удалить файл.");
            System.out.println("7 - Удалить папку.");
            System.out.println("8 - Выход из программы.");

            code = Byte.parseByte(scanner.nextLine());
            switch (code) {
                //показать содержимое диска
                case 1: {
                    customDisk.showFiles();
                    break;
                }
                //
                case 2: {
                    customDisk.showFreeSegments();
                    break;
                }
                //
                case 3: {
                    customDisk.showOccupiedSegments();
                    break;
                }
                //
                case 4: {
                    String path, directory;
                    System.out.println("Введите путь к файлу:");
                    path = scanner.nextLine();
                    System.out.println("Введите путь к файлу на гибком
диске:");
                    directory = scanner.nextLine();

                    try {
                        customDisk.writeFile(path, directory);
                    } catch (FileEmptyException e) {
                        System.out.println("Ошибка: файл пуст");
                    } catch (FileExistsException e) {
                        System.out.println("Ошибка: файл существует на гибком
диске");
                    } catch (NoDirectoryException e) {
                        System.out.println("Ошибка: указанной директории не
существует на гибком диске");
                    } catch (NoPlaceException e) {
                        System.out.println("Ошибка: на гибком диске нет
места");
                    } catch (FileNotFoundException e) {
                        System.out.println("Ошибка: файл не найден");
                    } catch (Exception e) {
                        System.out.println("Ошибка");
                        e.printStackTrace();
                    }
                    break;
                }
                //
                case 5: {
                    String path, name;
                    System.out.println("Введите путь к директории на гибком
диске:");
                    path = scanner.nextLine();
                    System.out.println("Введите имя папки:");
```

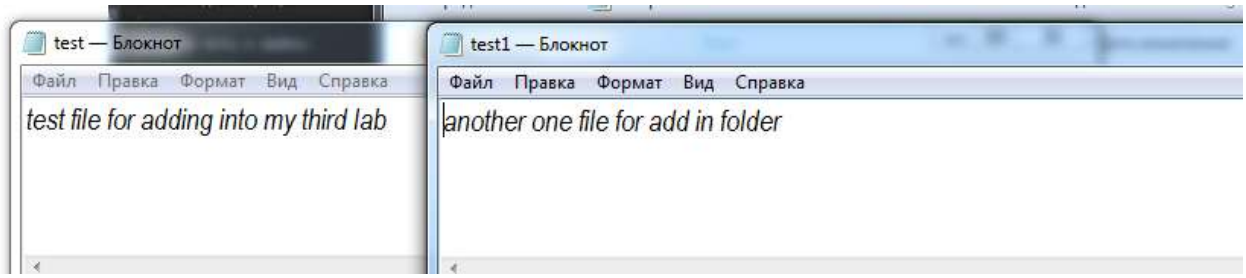
```
                    name = scanner.nextLine();

                    try {
                        customDisk.createDirectory(path, name);
                    } catch (FileExistsException e) {
                        System.out.println("Ошибка: папка уже существует на
гибком диске");
                    } catch (Exception e) {
                        System.out.println("Ошибка");
                        e.printStackTrace();
                    }
                    break;
                }
                //
                case 6: {
                    System.out.println("Введите путь к файлу на гибком
диске:");
                    String path = scanner.nextLine();

                    try {
                        customDisk.removeFile(path);
                    } catch (FileNotFoundException e) {
                        System.out.println("Ошибка: файл не существует на
гибком диске");
                    } catch (Exception e) {
                        System.out.println("Ошибка");
                        e.printStackTrace();
                    }
                    break;
                }
                //
                case 7: {
                    System.out.println("Введите путь к папке на гибком
диске:");
                    String path = scanner.nextLine();

                    try {
                        customDisk.removeDirectory(path);
                    } catch (Exception e) {
                        System.out.println("Ошибка");
                        e.printStackTrace();
                    }
                    break;
                }
                //выход
                case 8: {
                    iswork = false;
                    System.out.println("Нажмите Enter:");
                    scanner.nextLine();
                    break;
                }
            }
        }
    }
}
```

Скриншоты:


Main
3 - Отобразить список занятых сегментов.
4 - Добавить файл.
5 - Добавить папку.
6 - Удалить файл.
7 - Удалить папку.
8 - Выход из программы.
1
root/ (directory)
Выберите желаемый пункт меню.
1 - Отобразить содержимое 'диска'.
2 - Отобразить список свободных сегментов.
3 - Отобразить список занятых сегментов.
4 - Добавить файл.
5 - Добавить папку.
6 - Удалить файл.
7 - Удалить папку.
8 - Выход из программы.
4
Введите путь к файлу:
D:\test.txt
Введите путь к файлу на гибком диске:
root
Выберите желаемый пункт меню.
1 - Отобразить содержимое 'диска'.
2 - Отобразить список свободных сегментов.
3 - Отобразить список занятых сегментов.
4 - Добавить файл.
5 - Добавить папку.
6 - Удалить файл.
7 - Удалить папку.
8 - Выход из программы.
1
root/ (directory)
root/D:\test.txt; size=30; startIndex=0
Выберите желаемый пункт меню.


Main
1 - Отобразить содержимое 'диска'.
2 - Отобразить список свободных сегментов.
3 - Отобразить список занятых сегментов.
4 - Добавить файл.
5 - Добавить папку.
6 - Удалить файл.
7 - Удалить папку.
8 - Выход из программы.
5
Введите путь к директории на гибком диске:
root
Введите имя папки:
folder1
Выберите желаемый пункт меню.
1 - Отобразить содержимое 'диска'.
2 - Отобразить список свободных сегментов.
3 - Отобразить список занятых сегментов.
4 - Добавить файл.
5 - Добавить папку.
6 - Удалить файл.
7 - Удалить папку.
8 - Выход из программы.
1
root/ (directory)
root/D:\test.txt; size=38; startIndex=0
root/folder1/ (directory)
Выберите желаемый пункт меню.
1 - Отобразить содержимое 'диска'.
2 - Отобразить список свободных сегментов.
3 - Отобразить список занятых сегментов.
4 - Добавить файл.
5 - Добавить папку.
6 - Удалить файл.
7 - Удалить папку.
8 - Выход из программы.

```
5 - Добавить папку.
6 - Удалить файл.
7 - Удалить папку.
8 - Выход из программы.
4
Введите путь к файлу:
D:\test1.txt
Введите путь к файлу на гибком диске:
root/folder1

Выберите желаемый пункт меню.
1 - Отобразить содержимое 'диска'.
2 - Отобразить список свободных сегментов.
3 - Отобразить список занятых сегментов.
4 - Добавить файл.
5 - Добавить папку.
6 - Удалить файл.
7 - Удалить папку.
8 - Выход из программы.
2
startIndex=72; endIndex=1439999

Выберите желаемый пункт меню.
1 - Отобразить содержимое 'диска'.
2 - Отобразить список свободных сегментов.
3 - Отобразить список занятых сегментов.
4 - Добавить файл.
5 - Добавить папку.
6 - Удалить файл.
7 - Удалить папку.
8 - Выход из программы.
1
root/ (directory)
root/D:\test.txt; size=38; startIndex=0
root/folder1/ (directory)
root/folder1/D:\test1.txt; size=34; startIndex=38

Выберите желаемый пункт меню.
```



Вывод: в ходе лабораторной работы изучила работу с классами, обработку исключений.