МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №3

По дисциплине «Современные платформы программирования»
Специальность ПО-8

Выполнил:
Липовик И.С.
студент группы ПО-8
Проверил:
ст. преп. кафедры ИИТ,
«__»_____2024 г.

Брест 2024

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

Вариант 15

Задание 1. Множество целых чисел 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.

Выполнение

Код программы

Set.java

```
import java.util.Arrays;

public class Set {
    private int size;
    private int[] elements;

    Set(int capacity){
        size=0;
        elements=new int[capacity];
    }
    Set(int[] set){
        elements=set;
        size=set.length-1;
    }
    Set(){
        size=0;
        elements=new int[5];
    }
    public void merge(Set set){
        int[] newElements = new int[elements.length+set.elements.length];
        size+=set.size;
        System.arraycopy(elements, 0, newElements, 0, elements.length);
        System.arraycopy(set.elements, 0, newElements, elements.length,
set.elements.length);
        elements=new int[newElements.length];

elements=Arrays.stream(Arrays.stream(newElements).sorted().toArray()).distinc
t().toArray();
    }
    @Override
    public String toString(){
        StringBuilder str= new StringBuilder();
        for(int i=elements.length-size;i< elements.length;i++){
            str.append(elements[i]).append("\n");
        }
        return str.toString();
    }
    public int contains(int value){
        for(int i=elements.length-size;i< elements.length;i++){
            if(elements[i]==value){
                return i;
            }
```

```
        }
        return -1;
    }
    public void add(int value){
        if(elements.length==size){
            System.out.println("the set is completely filled!");
        }
        else{
            if(contains(value)==-1){
                elements[0]=value;
                elements=Arrays.stream(elements).sorted().toArray();
                size++;
            }
        }
    }
    public void erase(int value){
        int index=contains(value);
        if(index!=-1){
            elements[index]=0;
            size--;
            elements=Arrays.stream(elements).sorted().toArray();
        }
        else{
            System.out.println("the set does not contain this element!");
        }
    }
    @Override
    public boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }
        if (obj == null || obj.getClass() != this.getClass()) {
            System.out.println("not same class!");
            return false;
        }

        Set set = (Set) obj;
        return size==set.size && Arrays.equals(elements, set.elements);
    }
}
```

task1.java

```
public class task1 {
    public static void main(String[] args) {
        Set set= new Set(3);
        set.add(1);
        set.add(5);
        set.add(3);
        set.add(3);
        Set set1= new Set();
        set1.add(2);
        set1.add(4);
        set.merge(set1);
        System.out.println("set:\n"+set);
        System.out.println(set1.equals(set));
        Set set2= new Set();
        set2.add(2);
        set2.add(4);
        System.out.println(set1.equals(set2));
        set.erase(1);
        set.erase(2);
        System.out.println("set:\n"+set);
    }
}
```

Пример

```
D:\СПП\lab3\out\production\lab3>java task1
the set is completely filled!
set:
1
2
3
4
5

false
true
set:
3
4
5
```

Задание 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

File.java

```
import java.util.Arrays;

public class File {
    private String name;
    private int size;
    private String type;
    private String extension;

    private String parentDirectoryPath;
    File(String _name,int _size){
        String[] words = _name.split("/");
        name=words[words.length-1];
        words = name.split("\\.");
        extension = words[words.length-1];
        size=_size;
        parentDirectoryPath = _name.substring(0, _name.length()-
```

```
name.length());
        type="F";
    }
    File(String _name){
        name=_name;
        parentDirectoryPath = _name.substring(0, _name.length()-
name.length());
        type="D";
    }
    @Override
    public String toString(){
        String result="";
        if(type.equals("D")){
            result+="name: "+name;
        }
        else{
            String[] words = name.split("\\.");
            result+="name: "+words[0]+" size: "+size+" extension:
"+extension;
        }
        return result;
    }
    public void setName(String _name){
        name=_name;
    }
    public void setParentDirectoryPath(String _parentDirectoryPath){
        parentDirectoryPath=_parentDirectoryPath;
    }

    public String getName() {
        return name;
    }
    public int getSize() {
        return size;
    }

    public String getExtension() {
        return extension;
    }

    public String getParentDirectoryPath() {
        return parentDirectoryPath;
    }

    public String getType() {
        return type;
    }
}
```

Space.java

```
public class Space {
    private int spaceStart;
    private int spaceEnd;
    private File file;
    Space(int _spaceStart, int _spaceEnd){
        spaceStart=_spaceStart;
        spaceEnd=_spaceEnd;
    }

    public void setSpaceStart(int spaceStart) {
        this.spaceStart = spaceStart;
    }

    public void setSpaceEnd(int spaceEnd) {
```

```
        this.spaceEnd = spaceEnd;
    }

    public void setFile(File file) {
        this.file = file;
    }

    public int getSpaceStart() {
        return spaceStart;
    }

    public int getSpaceEnd() {
        return spaceEnd;
    }

    public File getFile() {
        return file;
    }
    @Override
    public String toString() {
        String result="";
        result +="begin: "+spaceStart+" end: "+spaceEnd;
        return result;
    }
}
```

Disk.java

```
import java.util.ArrayList;

public class Disk {
    private ArrayList<Space> freeSpace = new ArrayList<>();
    private ArrayList<Space> occupiedSpace = new ArrayList<>();
    Disk(){
        freeSpace.add(new Space(1, 1440));
    }
    public void addFile(File file){
        if(file.getType().equals("D")){
            Space directorySpace = new Space(0, 0);
            directorySpace.setFile(file);
            occupiedSpace.add(directorySpace);
            return;
        }
        for(Space space:freeSpace){
            if((space.getSpaceEnd()- space.getSpaceStart())> file.getSize()){
                Space newSpace= new Space(space.getSpaceStart(),
space.getSpaceStart()+file.getSize()-1);
                newSpace.setFile(file);
                occupiedSpace.add(newSpace);
                space.setSpaceStart(space.getSpaceStart()+file.getSize());
                return;
            }
        }
        System.out.println("not enough free space!");
    }
    public void deleteFile(File file){
        if(file.getType()=="D"){
            Space directorySpace=null;
            String directoryPath="";
            if(!file.getParentDirectoryPath().isEmpty()){
                directoryPath+=file.getParentDirectoryPath()+"/";
            }
            directoryPath+=file.getName()+"/";
            Space space;
            for(int i =0;i<occupiedSpace.size();i++){
```

```
                space=occupiedSpace.get(i);

if(space.getFile().getParentDirectoryPath().equals(directoryPath)){
                    deleteFile(space.getFile());
                    i--;
                }
                if(space.getFile()==file){
                    directorySpace=space;
                }
            }
            occupiedSpace.remove(directorySpace);
            return;
        }
        for(Space space:occupiedSpace){
            if(space.getFile() == file){

                Space rightSpace = null, leftSpace=null;
                for(Space fSpace:freeSpace){
                    if(fSpace.getSpaceStart() == space.getSpaceEnd()+1){
                        rightSpace=fSpace;
                        continue;
                    }
                    if(fSpace.getSpaceEnd() == space.getSpaceStart()-1){
                        leftSpace=fSpace;
                    }
                }
                if(rightSpace != null && leftSpace!=null){
                    leftSpace.setSpaceEnd(rightSpace.getSpaceEnd());
                    occupiedSpace.remove(space);
                    freeSpace.remove(rightSpace);
                    return;
                }
                else if(rightSpace != null && leftSpace==null){
                    rightSpace.setSpaceStart(space.getSpaceStart());
                    occupiedSpace.remove(space);
                    return;
                }
                else if(rightSpace == null && leftSpace!=null){
                    leftSpace.setSpaceEnd(space.getSpaceEnd());
                    occupiedSpace.remove(space);
                    return;
                }
                else if(rightSpace == null && leftSpace==null){
                    occupiedSpace.remove(space);
                    space.setFile(null);
                    freeSpace.add(space);
                    return;
                }
            }
        }
    }
    public void printFreeSpace(){
        System.out.println("free spaces:");
        for(Space space:freeSpace){
            System.out.println(space);
        }
    }
    public void printOccupiedSpace(){
        System.out.println("occupied spaces:");
        for(Space space:occupiedSpace){
            if(!space.getFile().getType().equals("D")) {
                System.out.println(space);
                System.out.println(space.getFile());
            }
```

```
        }
    }
}
```

Task2.java

```
public class task2 {
    public static void main(String[] args){
        Disk disk = new Disk();
        File file1 = new File("a.txt", 700);
        File directory = new File ("a");
        File file3 = new File("a/c.txt", 300);
        File file2 = new File("a/b.txt", 300);
        File file4 = new File("d.txt", 400);
        disk.addFile(file1);
        disk.addFile(directory);
        disk.addFile(file3);
        disk.addFile(file2);
        disk.printOccupiedSpace();
        disk.deleteFile(directory);
        disk.printFreeSpace();
        disk.printOccupiedSpace();
        disk.addFile(file3);
        disk.addFile(file2);
        disk.deleteFile(file3);
        disk.addFile(file4);
    }
}
```

Пример

```
D:\СПП\lab3\out\production\lab3>java task2
Add file1, directory, file2 and file3
occupied spaces:
begin: 1 end: 700
name: a size: 700 extension: txt
begin: 701 end: 1000
name: c size: 300 extension: txt
begin: 1001 end: 1300
name: b size: 300 extension: txt
delete directory with file2 and file3
free spaces:
begin: 701 end: 1440
occupied spaces:
begin: 1 end: 700
name: a size: 700 extension: txt
Add file2 and file3
delete file3
Add file4
not enough free space!
```

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