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

ОТЧЁТ

по лабораторной работе №3

Выполнил:

студент группы ПО-8
Вейгандт И.О.

Проверил:
Крощенко А.А.

Брест 2024

Вариант 5

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 находится 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 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.

Задание 1:

Main.java

```
package task01;

public class Main {
    public static void main(String[] args) {
        Task01 set1 = new Task01(5);
        set1.add('a');
        set1.add('b');
        set1.add('c');
        System.out.println("Set 1: " + set1);
```

```
        Task01 set2 = new Task01(5);
        set2.add('b');
        set2.add('c');
        set2.add('d');
        System.out.println("Set 2: " + set2);

        set1.remove('b');
        System.out.println("Set 1 after removing 'b': " +
set1);

        set2.remove('c');
        System.out.println("Set 2 after removing 'c': " +
set2);

        System.out.println("Is 'c' in Set 1? " +
set1.contains('c'));
        System.out.println("Is 'd' in Set 1? " +
set1.contains('d'));

        System.out.println("Are Set 1 and Set 2 equal? " +
set1.equals(set2));

    }
}
```

Task01.java

```
package task01;

import java.util.Arrays;

public class Task01 {
    private char[] symbols;
    private int size;

    public Task01(int capacity) {
        symbols = new char[capacity];
        size = 0;
    }

    public void add(char element) {
        if (size >= symbols.length) {
            System.out.println("Array is full. Cannot add
more elements.");
```

```
            return;
        }

        if (contains(element)) {
            System.out.println("Element " + element + "
already exists in the array.");
            return;
        }

        symbols[size++] = element;
        System.out.println("Added element " + element);
    }

    public void remove(char element) {
        for (int i = 0; i < size; i++) {
            if (symbols[i] == element) {
                symbols[i] = symbols[size - 1];
                symbols[size - 1] = 0;
                size--;
                System.out.println("Removed element " +
element);
                return;
            }
        }
        System.out.println("Element " + element + " not
found in the array.");
    }

    public boolean contains(char element) {
        for (int i = 0; i < size; i++) {
            if (symbols[i] == element) {
                return true;
            }
        }
        return false;
    }

    @Override
    public String toString() {
        return Arrays.toString(Arrays.copyOf(symbols,
size));
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == this) {
```

```
            return true;
        }
        if (!(obj instanceof Task01)) {
            return false;
        }
        Task01 other = (Task01) obj;
        return Arrays.equals(Arrays.copyOf(symbols, size),
Arrays.copyOf(other.symbols, other.size));
    }
}
```

```
C:\Users\veiga\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent
Added element a
Added element b
Added element c
Set 1: [a, b, c]
Added element b
Added element c
Added element d
Set 2: [b, c, d]
Removed element b
Set 1 after removing 'b': [a, c]
Removed element c
Set 2 after removing 'c': [b, d]
Is 'c' in Set 1? true
Is 'd' in Set 1? false
Are Set 1 and Set 2 equal? false
```

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.

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.Iterator;
import java.util.List;
import java.io.BufferedReader;
import java.io.FileReader;

class FileSystemObject {
    String name;

    public FileSystemObject(String name) {
        this.name = name;
    }
}

class File extends FileSystemObject {
    String extension;
    int size;

    public File(String name, String extension, int size) {
        super(name);
        this.extension = extension;
        this.size = size;
    }
```

```
}

class Directory extends FileSystemObject {
    List<FileSystemObject> contents;

    public Directory(String name) {
        super(name);
        this.contents = new ArrayList<>();
    }

    public void addFile(File file) {
        contents.add(file);
        System.out.println("Файл '" + file.name + "." +
file.extension + "' добавлен в директорию '" + name +
"'.");
    }

    public void addDirectory(Directory directory) {
        contents.add(directory);
        System.out.println("Директория '" + directory.name
+ "' добавлена в директорию '" + name + "'.");
    }

    public void printTree(String prefix) {
        System.out.println(prefix + name);
        for (FileSystemObject object : contents) {
            if (object instanceof File) {
                System.out.println(prefix + "  - " +
((File) object).name + "." + ((File) object).extension);
            }
        }
        for (FileSystemObject object : contents) {
```

```
            if (object instanceof Directory) {
                ((Directory) object).printTree(prefix + "
");
            }
        }
    }

}

class Segment {
    int start;
    int size;
    String fileName = "free";

    public Segment(int start, int size) {
        this.start = start;
        this.size = size;
    }
    public Segment(int start, int size, String fileName) {
        this.start = start;
        this.size = size;
        this.fileName = fileName;
    }
}

class Disk {
    int totalSize;
    int freeSpace;
    Directory rootDirectory;
    List<Segment> freeSegments;
```

```
    public Disk(int totalSize) {
        this.totalSize = totalSize;
        this.freeSpace = totalSize;
        this.rootDirectory = new Directory("root");
        this.freeSegments = new ArrayList<>();
        this.freeSegments.add(new Segment(0, totalSize));
    }

    public void addFile(File file, String path) {
        Directory currentDirectory = findDirectory(path);
        if (currentDirectory == null) {
            System.out.println("Ошибка: Директория '" +
path + "' не найдена. Файл не добавлен.");
            return;
        }

        if (file.size > totalSize) {
            System.out.println("Ошибка: Размер файла '" +
file.name + "' превышает размер диска. Файл не добавлен.");
            return;
        }

        if (file.size > freeSpace) {
            System.out.println("Ошибка: Недостаточно
свободного места на диске для файла '" + file.name + "'.
Файл не добавлен.");
            return;
        }

        currentDirectory.addFile(file);
        allocateSpace(file, true);
    }
```

```
    public void addDirectory(Directory directory, String
path) {
        Directory currentDirectory = findDirectory(path);
        if (currentDirectory == null) {
            System.out.println("Ошибка: Директория '" +
path + "' не найдена. Директория не добавлена.");
            return;
        }

        currentDirectory.addDirectory(directory);
    }

    public void removeFile(String fileName, String path) {
        Directory currentDirectory = findDirectory(path);
        if (currentDirectory == null) {
            return;
        }

        Iterator<FileSystemObject> iterator =
currentDirectory.contents.iterator();
        while (iterator.hasNext()) {
            FileSystemObject object = iterator.next();
            if (object instanceof File &&
object.name.equals(fileName)) {
                iterator.remove();
                freeSpace += ((File) object).size;
                allocateSpace((File) object,false);
                System.out.println("Файл '" + fileName + "'
удален из директории '" + path + "'.");
                return;
            }
```

```
        }

        System.out.println("Ошибка: Файл '" + fileName + "' не найден в директории '" + path + "'.");
    }

    public void removeDirectory(String directoryName, String path) {
        Directory currentDirectory = findDirectory(path);
        if (currentDirectory == null) {
            return;
        }

        Iterator<FileSystemObject> iterator = currentDirectory.contents.iterator();
        while (iterator.hasNext()) {
            FileSystemObject object = iterator.next();
            if (object instanceof Directory && object.name.equals(directoryName)) {
                iterator.remove();
                freeSpace += calculateDirectorySize((Directory) object);
                removeTree((Directory) object);
                System.out.println("Директория '" + directoryName + "' удалена из директории '" + path + "'.");
                return;
            }
        }

        System.out.println("Ошибка: Директория '" + directoryName + "' не найдена в директории '" + path + "'.");
    }
```

```
    public void removeTree(Directory directory) {
        for (FileSystemObject obj : directory.contents){
            if (obj instanceof File){
                removeFile(obj.name, directory.name);
                allocateSpace((File) obj, false);
            }
        }
        for (FileSystemObject object : directory.contents)
{
            if (object instanceof Directory) {
                removeTree((Directory) object);
            }
        }
    }

    private int calculateDirectorySize(Directory directory)
{
        int size = 0;
        for (FileSystemObject object : directory.contents)
{
            if (object instanceof File) {
                size += ((File) object).size;
            } else if (object instanceof Directory) {
                size += calculateDirectorySize((Directory)
object);
            }
        }
        return size;
    }

    private void allocateSpace(File file, boolean mode) {
```

```
        for (int i = 0; i < freeSegments.size(); i++) {
            Segment suitableSegment = freeSegments.get(i);
            if(mode) {
                if (suitableSegment.fileName.equals("free")
&& file.size <= suitableSegment.size) {
                    freeSpace -= file.size;
                    int newStart = suitableSegment.start +
file.size;
                    int newSize = suitableSegment.size -
file.size;
                    Segment freeSegment = new
Segment(newStart, newSize);
                    freeSegments.add(freeSegment);
                    suitableSegment.fileName = file.name;
                    suitableSegment.size = file.size;
                    return;
                }
            } else {

if(suitableSegment.fileName.equals(file.name)){
                    suitableSegment.fileName = "free";
                    mergeFreeSegments();
                }
            }
        }
    }


    private void mergeFreeSegments() {
        freeSegments.sort(Comparator.comparingInt(segment -
> segment.start));
        for (int i = 0; i < freeSegments.size() - 1; i++) {
            Segment currentSegment = freeSegments.get(i);
            Segment nextSegment = freeSegments.get(i + 1);
```

```
            if (currentSegment.fileName == "free" &&
nextSegment.fileName == "free") {
                currentSegment.size += nextSegment.size;
                freeSegments.remove(i + 1);
                i--;
            }
        }
    }

    private Directory findDirectory(String path) {
        if (path.equals("root")) {
            return rootDirectory;
        }

        String[] parts = path.split("/");
        Directory currentDirectory = rootDirectory;
        boolean directoryFound = false;
        for (String part : parts) {

            for (FileSystemObject object :
currentDirectory.contents) {
                if (object instanceof Directory &&
object.name.equals(part)) {
                    currentDirectory = (Directory) object;
                    directoryFound = true;
                    break;
                }
            }

        }

        if (!directoryFound) {
```

```
            return null;
        }


        return currentDirectory;
    }


    public void printDiskStatus() {
        System.out.println("Общий размер диска: " +
totalSize + " Кб");
        System.out.println("Свободное место на диске: " +
freeSpace + " Кб");
        for(Segment segment : freeSegments){
            System.out.println(segment.start + " - " +
(segment.size + segment.start) + "(" + segment.fileName +
")");
        }
        System.out.println("Структура файлов и
директорий:");
        rootDirectory.printTree("");
    }
}


public class Zad2 {
        public static void main(String[] args) {
            Disk disk = new Disk(1440);


            try (BufferedReader reader = new
BufferedReader(new
FileReader("C:\\Users\\Вадим\\IdeaProjects\\untitled2\\out\
\input.txt"))) {
                String line;
                while ((line = reader.readLine()) != null)
{
                    String[] tokens = line.split("\\s+");
```

```
                    switch (tokens[0]) {
                        case "ADD_FILE":
                            File file = new File(tokens[1],
tokens[2], Integer.parseInt(tokens[3]));
                            disk.addFile(file, tokens[4]);
                            break;

                        case "ADD_DIRECTORY":
                            Directory directory = new
Directory(tokens[1]);
                            disk.addDirectory(directory,
tokens[2]);
                            break;

                        case "REMOVE_FILE":
                            disk.removeFile(tokens[1],
tokens[2]);
                            break;

                        case "REMOVE_DIRECTORY":
                            disk.removeDirectory(tokens[1],
tokens[2]);
                            break;
                    }
                }
            } catch (IOException e) {
                e.printStackTrace();
            }

            disk.printDiskStatus();
        }
```

```
    }
```

input.txt – Блокнот

Файл Правка Формат Вид Справка

```
ADD_DIRECTORY subdir root
ADD_DIRECTORY subsubdir subdir
ADD_FILE document txt 200 root
ADD_FILE info jpg 100 root
ADD_FILE data csv 300 root/subdir/subsubdir
ADD_FILE code java 400 root/subdir
ADD_FILE image bmp 500 root
REMOVE_FILE document root
REMOVE_DIRECTORY subdir root
```

```
Директория 'subdir' добавлена в директорию 'root'.
Директория 'subsubdir' добавлена в директорию 'subdir'.
Файл 'document.txt' добавлен в директорию 'root'.
Файл 'info.jpg' добавлен в директорию 'root'.
Файл 'data.csv' добавлен в директорию 'subsubdir'.
Файл 'code.java' добавлен в директорию 'subdir'.
Ошибка: Недостаточно свободного места на диске для файла 'image'. Файл не добавлен.
Общий размер диска: 1440 Кб
Свободное место на диске: 440 Кб
0 - 200(document)
200 - 300(info)
300 - 600(data)
600 - 1000(code)
1000 - 1440(free)
Структура файлов и директорий:
root
  - document.txt
  - info.jpg
  subdir
    - code.java
    subsubdir
      - data.csv
```

Вывод: приобрёл практические навыки работы с классами в программах на языке программирования Java.