МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №3

Специальность ПО5

Выполнил:
А.А. Игнатюк,
студент группы ПО-5

Проверил:
А.А. Крощенко,
ст. преп. кафедры ИИТ,
«__» _____ 2021 г.

Брест 2021

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

Вариант 5.

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().

5) 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.

Спецификация ввода:
java Main

Спецификация вывода:
<параметры функций System.out.println() (содержимое коллекций, результаты сравнений)>
…

Структура проекта:

Код программы:

Java > Set > src > Set.java > ...

```java
public final class Set {
    private Integer[] base;

    private final void sort() {
        Boolean changed = false;

        for (Integer i = 0, size = this.base.length; i < size; ++i) {
            for (Integer j = 0, k = 1; k < size - i; ++j, ++k) {
                if (this.base[j] > this.base[k]) {
                    changed = true;

                    Integer temp = this.base[j];
                    this.base[j] = this.base[k];
                    this.base[k] = temp;
                }
            }

            if (!changed) {
                return;
            }
        }
    }

    public Set(Integer[] base) {
        this.base = new Integer[] {};

        for (final Integer item : base) {
            this.add(item);
        }

        this.sort();
    }

    public final Boolean equals(final Set other) {
        if (this.base.length != other.base.length) {
            return false;
        }

        for (Integer i = 0, size = this.base.length; i < size; ++i) {
            if (this.base[i] != other.base[i]) {
                return false;
            }
        }

        return true;
    }

    public final String toString() {
        if (this.base.length == 0) {
            return new String();
        }

        final String SEPARATOR = new String(", ");
        String result = new String();

        for (Integer i = 0, size = this.base.length - 1; i < size; ++i) {
            result += Integer.toString(this.base[i]) + SEPARATOR;
        }

        result += Integer.toString(this.base[this.base.length - 1]);
        return result;
    }
```

```
63
64      public final void join(final Set other) {
65          for (final Integer item : other.base) {
66              this.add(item);
67          }
68
69          this.sort();
70      }
71
72      public final void print() {
73          System.out.println(this.toString());
74      }
75
76      public final Boolean isMember(final Integer item) {
77          for (Integer i = 0, size = this.base.length; i < size; ++i) {
78              if (this.base[i] == item) {
79                  return true;
80              }
81          }
82
83          return false;
84      }
85
86      public final void add(final Integer item) {
87          if (this.isMember(item)) {
88              return;
89          }
90
91          Integer[] newBase = new Integer[this.base.length + 1];
92
93          for (Integer i = 0, size = this.base.length; i < size; ++i) {
94              newBase[i] = this.base[i];
95          }
96
97          newBase[this.base.length] = item;
98          this.base = newBase;
99          this.sort();
100     }
101
102     public final void remove(final Integer item) {
103         if (!this.isMember(item)) {
104             return;
105         }
106
107         Integer[] newBase = new Integer[this.base.length - 1];
108
109         for (Integer i = 0, size = this.base.length; i < size; ++i) {
110             if (this.base[i] != item) {
111                 newBase[i] = this.base[i];
112             }
113         }
114
115         this.base = newBase;
116         this.sort();
117     }
118 }
119
```

Java > Set > src > Main.java > ...

```java
public final class Main {
    Run | Debug
    public static final void main(String[] args) throws Exception {
        Set set_1 = new Set(new Integer[] {});
        set_1.print();

        set_1.add(5);
        set_1.add(5);
        set_1.add(6);
        set_1.add(7);
        set_1.print();

        set_1.remove(7);
        set_1.print();

        Set set_2 = new Set(new Integer[] { 3, 6, 7, 3, 9, 2 });
        set_2.print();

        Set set_3 = new Set(new Integer[] { 4, 3, 8, 1, 7 });
        set_3.print();

        if (set_2.equals(set_3)) {
            System.out.println("set_2 is equal to set_3");
        } else {
            System.out.println("set_2 is not equal to set_3");
        }

        Set set_4 = new Set(new Integer[] {});
        set_4.join(set_1);
        set_4.join(set_2);
        set_4.join(set_3);
        set_4.print();
    }
}
```

Результаты тестирования:

```
5, 6, 7
5, 6
2, 3, 6, 7, 9
1, 3, 4, 7, 8
set_2 is not equal to set_3
1, 2, 3, 4, 5, 6, 7, 8, 9
```

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.

5) 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.

Спецификация ввода:

./start

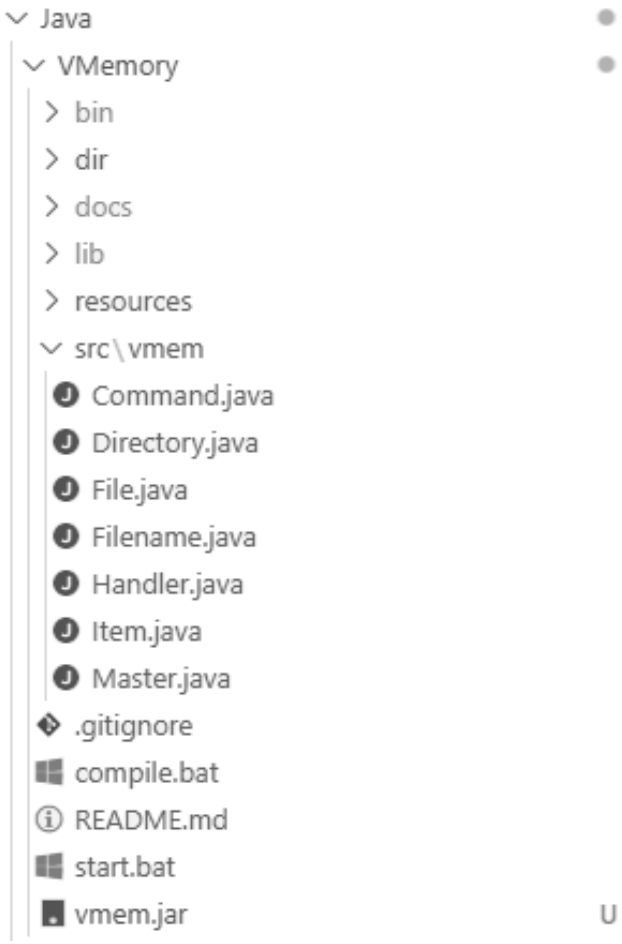
> <команда> [параметры]

Спецификация вывода:

<параметры функций System.out.println() (данные о системе, ошибки, предупреждения)>

…

Структура проекта:



Код программы и результаты тестирования:

Java > VMemory > src > vmem > Command.java > ...

```java
package vmem;

public final class Command implements java.io.Serializable {
    private String name = null;
    private Integer paramsCount = null;

    public Command(final String name, final Integer paramsCount) {
        this.name = new String(name);
        this.paramsCount = paramsCount;
    }

    public final String getName() {
        return new String(this.name);
    }

    public final void setName(final String name) {
        this.name = new String(name);
    }

    public final Integer getParamsCount() {
        return this.paramsCount;
    }

    public final void setParamsCount(final Integer paramsCount) {
        this.paramsCount = paramsCount;
    }
}

```

Java > VMemory > src > vmem > Directory.java > ...

```java
package vmem;

import java.util.Vector;

public final class Directory extends Item {
    private Vector<Item> content = null;

    public Directory(final String location, final String name) {
        super(location, name, Item.Type.DIRECTORY);
        this.content = new Vector<Item>();
    }

    public final Vector<Item> getContent() {
        return this.content;
    }

    @Override
    public final void printInfo() {
        System.out.println("d\t" + this.getLocation() + "\t" + this.getName() + "\t\t"
                + Integer.toString(this.content.size()) + " items");
    }
}

```

Java > VMemory > src > vmem > File.java > ...

```java
package vmem;

import net.sourceforge.sizeof.SizeOf;

public final class File extends Item {
    private String extension = null, content = null;

    public File(final String location, final String name, final String extension) {
        super(location, name, Item.Type.FILE);
        this.extension = new String(extension);
        this.content = new String();
    }

    public final String getExtension() {
        return new String(this.extension);
    }

    public final void setExtension(final String extension) {
        this.extension = new String(extension);
    }

    public final String getContent() {
        return new String(this.content);
    }

    public final void setContent(final String content) {
        this.content = new String(content);
    }

    public final void append(final String content) {
        this.content += new String(content);
    }

    @Override
    public final void printInfo() {
        System.out.println("f\t" + this.getLocation() + "\t" + this.getName() + "\t" + this.extension + "\t"
                + Long.toString(SizeOf.deepSizeOf(this)) + " bytes");
    }
}
```

Java > VMemory > src > vmem > Filename.java > ...

```java
package vmem;

public final class Filename implements java.io.Serializable {
    private String name = null, extension = null;

    public Filename(final String name, final String extension) {
        this.name = new String(name);
        this.extension = new String(extension);
    }

    public final String getName() {
        return new String(this.name);
    }

    public final void setName(final String name) {
        this.name = new String(name);
    }

    public final String getExtension() {
        return new String(this.extension);
    }

    public final void setExtension(final String extension) {
        this.extension = new String(extension);
    }
}

```

Java > VMemory > src > vmem > Item.java > ...

```java
package vmem;

public abstract class Item implements java.io.Serializable {
    public static enum Type {
        FILE, DIRECTORY
    }

    private String location = null, name = null;
    private Item.Type type = null;

    protected Item(final String location, final String name, final Item.Type type) {
        this.location = new String(location);
        this.name = new String(name);
        this.type = type;
    }

    protected final String getName() {
        return new String(this.name);
    }

    protected final void setName(final String name) {
        this.name = new String(name);
    }

    protected final Item.Type getType() {
        return this.type;
    }

    protected final String getLocation() {
        return new String(this.location);
    }

    protected abstract void printInfo();
}

```

Java > VMemory > src > vmem > Handler.java > ...

```java
package vmem;

import java.util.Scanner;
import java.util.Vector;

import net.sourceforge.sizeof.SizeOf;

public final class Handler implements java.io.Serializable {
    private Long packSize = null;
    private Directory root = null, workingDirectory = null;
    private static Vector<Command> commands = null;

    private enum CommandType {
        LIST_ALL, CHANGE_DIRECTORY, WORKING_DIRECTORY, DISK_USAGE, MAKE_FILE, REMOVE_FILE, MAKE_DIR, REMOVE_DIR,
        FILL_FILE, PRINT_FILE, HELP, QUIT
    }

    static {
        Handler.commands = new Vector<Command>();
        Handler.commands.add(new Command(new String("la"), 1));
        Handler.commands.add(new Command(new String("cd"), 2));
        Handler.commands.add(new Command(new String("wd"), 1));
        Handler.commands.add(new Command(new String("du"), 1));
        Handler.commands.add(new Command(new String("mf"), 2));
        Handler.commands.add(new Command(new String("rf"), 2));
        Handler.commands.add(new Command(new String("md"), 2));
        Handler.commands.add(new Command(new String("rd"), 2));
        Handler.commands.add(new Command(new String("ff"), 3));
        Handler.commands.add(new Command(new String("pf"), 2));
        Handler.commands.add(new Command(new String("h"), 1));
        Handler.commands.add(new Command(new String("q"), 1));
    }

    public Handler(final Long packSize) {
        final String HOME = new String("/");

        this.packSize = packSize;
        this.root = new Directory(new String(HOME), new String(""));
        this.workingDirectory = this.root;

        System.out.println("[INFO]: Pack created, size = " + Long.toString(this.packSize) + " bytes.");
    }

    public final void start() {
        String answer = new String();
        String[] details = null;

        final Scanner SCANNER = new Scanner(System.in);
        this.printHelp();

        while (true) {
            System.out.print("> ");
            answer = SCANNER.nextLine();

            if (answer.isBlank()) {
                continue;
            }

            answer = answer.trim();
            details = answer.split("\\s+");

            boolean isFound = false;
            final Integer COMMAND_INDEX = 0;

```

```java
for (int i = 0, size = Handler.commands.size(); i < size; ++i) {
    if (!details[COMMAND_INDEX].toLowerCase()
            .equals(Handler.commands.elementAt(i).getName().toLowerCase())) {
        continue;
    }

    isFound = true;

    if (details.length != Handler.commands.elementAt(i).getParamsCount()) {
        System.out.println(
                "[ERROR]: Invalid parameters with \"" + details[COMMAND_INDEX] + "\" command! Must be "
                        + Integer.toString(Handler.commands.elementAt(i).getParamsCount()) + '!');
        break;
    }

    switch (CommandType.values()[i]) {
        case LIST_ALL: {
            this.listAll();
            break;
        }
        case CHANGE_DIRECTORY: {
            final Integer NAME_INDEX = 1;
            final String UP_DIRECTORY = new String("../"), DIRNAME = new String(details[NAME_INDEX]);

            if (DIRNAME.equals(UP_DIRECTORY)) {
                final String SLASH = "/", HOME = new String(SLASH);

                if (!this.getCurrentPath().equals(HOME)) {
                    String currentPath = this.getCurrentPath();

                    currentPath = new String(currentPath.substring(0, currentPath.lastIndexOf(SLASH)));

                    if (currentPath.equals(new String(""))) {
                        currentPath = new String(HOME);
                    }

                    this.workingDirectory = this.getDirectoryByPath(currentPath);
                }

                break;
            }

            if (!this.changeDirectory(DIRNAME)) {
                System.out
                        .println("[ERROR]: The directory \'" + details[NAME_INDEX] + "\" does not exist!");
            }

            break;
        }
        case WORKING_DIRECTORY: {
            System.out.println(this.getCurrentPath());
            break;
        }
        case DISK_USAGE: {
            this.printDiskUsage();
            break;
        }
        case MAKE_FILE: {
            final Integer NAME_INDEX = 1;
            final Filename FILENAME = this.parseName(details[NAME_INDEX]);

            if (FILENAME.getExtension().equals(new String(""))) {
                System.out.println("[WARNING]: Creating a file without extension!");
            }

            if (!this.makeFile(FILENAME)) {
                System.out.println("[ERROR]: The file \'" + details[NAME_INDEX]
                        + "\" already exist or you do not have enough space!");
            }

            break;
        }
```

```
                case REMOVE_FILE: {
                    final Integer NAME_INDEX = 1;
                    final Filename FILENAME = this.parseName(details[NAME_INDEX]);

                    if (!this.removeFile(FILENAME)) {
                        System.out.println("[ERROR]: The file \'" + details[NAME_INDEX] + "\" does not exist!");
                    }

                    break;
                }
                case MAKE_DIR: {
                    final Integer NAME_INDEX = 1;

                    if (!this.makeDir(details[NAME_INDEX])) {
                        System.out.println("[ERROR]: The directory \'" + details[NAME_INDEX]
                                + "\" already exist or you don't have enough space!");
                    }

                    break;
                }
                case REMOVE_DIR: {
                    final Integer NAME_INDEX = 1;

                    if (!this.removeDir(details[NAME_INDEX])) {
                        System.out
                                .println("[ERROR]: The directory \'" + details[NAME_INDEX] + "\" does not exist!");
                        break;
                    }

                    break;
                }
                case FILL_FILE: {
                    final Integer NAME_INDEX = 1, CONTENT_INDEX = 2;
                    final Filename FILENAME = this.parseName(details[NAME_INDEX]);

                    if (!this.fillFile(FILENAME, details[CONTENT_INDEX])) {
                        System.out.println("[ERROR]: The file \'" + details[NAME_INDEX]
                                + "\" does not exist or you don't have enough space!");
                    }

                    break;
                }
                case PRINT_FILE: {
                    final Integer NAME_INDEX = 1;
                    final Filename FILENAME = this.parseName(details[NAME_INDEX]);

                    if (!this.printFile(FILENAME)) {
                        System.out.println("[ERROR]: The file \'" + details[NAME_INDEX] + "\" does not exist!");
                    }

                    break;
                }
                case HELP: {
                    this.printHelp();
                    break;
                }
                case QUIT: {
                    SCANNER.close();
                    return;
                }
            }

            break;
        }

        if (!isFound) {
            final Integer COMMANT_INDEX = 0;
            System.out.println("[ERROR]: Unknown command \"" + details[COMMANT_INDEX] + "\"!");
        }
    }
}
```

```
209     private final Long used() {
210         return SizeOf.deepSizeOf(this.root);
211     }
212 
213     private final Filename parseName(final String name) {
214         final Integer DOT = name.lastIndexOf('.', name.length());
215         String clearName = new String(name), extension = new String("");
216 
217         if (DOT != -1 && DOT != 0) {
218             extension = name.substring(DOT, name.length());
219             clearName = name.substring(0, DOT);
220         }
221 
222         return new Filename(clearName, extension);
223     }
224 
225     private final Boolean isFileExist(final Filename filename) {
226         for (final Item item : this.workingDirectory.getContent()) {
227             if (item.getType() == Item.Type.FILE && ((File) item).getName().equals(filename.getName())
228                     && ((File) item).getExtension().equals(filename.getExtension())) {
229                 return true;
230             }
231         }
232 
233         return false;
234     }
235 
236     private final Boolean isDirectoryExist(final String name) {
237         for (final Item item : this.workingDirectory.getContent()) {
238             if (item.getType() == Item.Type.DIRECTORY && ((Directory) item).getName().equals(name)) {
239                 return true;
240             }
241         }
242 
243         return false;
244     }
245 
246     private final String getCurrentPath() {
247         final String SLASH = new String("/");
248 
249         if (this.workingDirectory.getLocation().equals(SLASH)) {
250             return new String(this.workingDirectory.getLocation() + this.workingDirectory.getName());
251         }
252 
253         return new String(this.workingDirectory.getLocation() + SLASH + this.workingDirectory.getName());
254     }
255 
256     private final Directory getDirectoryByPath(final String path) {
257         final String SLASH = new String("/");
258         final String[] DIRECTORIES = path.split(SLASH);
259 
260         Directory currentDirectory = root;
261         Vector<Item> currentContent = this.root.getContent();
262 
263         for (int i = 1, size = DIRECTORIES.length; i < size; ++i) {
264             for (final Item item : currentContent) {
265                 if (item.getType() == Item.Type.DIRECTORY && item.getName().equals(DIRECTORIES[i])) {
266                     currentDirectory = (Directory) item;
267                     currentContent = ((Directory) item).getContent();
268                     break;
269                 }
270             }
271         }
272 
273         return currentDirectory;
274     }
275 
```

```java
    private final Boolean changeDirectory(final String name) {
        for (final Item item : this.workingDirectory.getContent()) {
            if (item.getType() == Item.Type.DIRECTORY && ((Directory) item).getName().equals(name)) {
                final String SLASH = new String("/");
                String currentPath = this.getCurrentPath();

                if (!currentPath.endsWith(SLASH)) {
                    currentPath += SLASH;
                }

                currentPath = new String(currentPath + name);
                this.workingDirectory = this.getDirectoryByPath(currentPath);

                return true;
            }
        }

        return false;
    }

    private final void listAll() {
        for (final Item item : this.workingDirectory.getContent()) {
            item.printInfo();
        }
    }

    private final void printDiskUsage() {
        final Long USED = this.used(), FREE = this.packSize - USED;

        final Double PERCENT = 100.0, PERCENTAGE_USED = (double) USED * PERCENT / this.packSize,
                PERCENTAGE_FREE = (double) FREE * PERCENT / this.packSize;

        System.out.println("[INFO]: Storage (bytes): " + Long.toString(this.packSize));

        System.out.print("Used (bytes): " + Long.toString(USED) + "; Used (%): ");
        System.out.printf("%.2f\n", PERCENTAGE_USED);

        System.out.print("Free (bytes): " + Long.toString(FREE) + "; Used (%): ");
        System.out.printf("%.2f\n", PERCENTAGE_FREE);
    }

    private final Boolean makeFile(final Filename filename) {
        if (this.isFileExist(filename)) {
            return false;
        }

        final File NEW_FILE = new File(this.getCurrentPath(), filename.getName(), filename.getExtension());

        if (this.used() + SizeOf.deepSizeOf(NEW_FILE) > this.packSize) {
            return false;
        }

        this.workingDirectory.getContent().add(NEW_FILE);
        return true;
    }

```

```java
    private final Boolean removeFile(final Filename filename) {
        for (int i = 0, size = this.workingDirectory.getContent().size(); i < size; ++i) {
            final Item ITEM = this.workingDirectory.getContent().elementAt(i);

            if (ITEM.getType() == Item.Type.FILE && ((File) ITEM).getName().equals(filename.getName())
                    && ((File) ITEM).getExtension().equals(filename.getExtension())) {
                this.workingDirectory.getContent().remove(i);
                return true;
            }
        }

        return false;
    }

    private final Boolean makeDir(final String name) {
        if (this.isDirectoryExist(name)) {
            return false;
        }

        final Directory NEW_DIRECTORY = new Directory(this.getCurrentPath(), name);

        if (this.used() + SizeOf.deepSizeOf(NEW_DIRECTORY) > this.packSize) {
            return false;
        }

        this.workingDirectory.getContent().add(NEW_DIRECTORY);
        return true;
    }

    private final Boolean removeDir(final String name) {
        for (int i = 0, size = this.workingDirectory.getContent().size(); i < size; ++i) {
            final Item ITEM = this.workingDirectory.getContent().elementAt(i);

            if (ITEM.getType() == Item.Type.DIRECTORY && ((Directory) ITEM).getName().equals(name)) {
                this.workingDirectory.getContent().remove(i);
                return true;
            }
        }

        return false;
    }

    private final Boolean fillFile(final Filename filename, final String content) {
        if (this.used() + SizeOf.deepSizeOf(content) > this.packSize) {
            return false;
        }

        for (final Item item : this.workingDirectory.getContent()) {
            if (item.getType() == Item.Type.FILE && ((File) item).getName().equals(filename.getName())
                    && ((File) item).getExtension().equals(filename.getExtension())) {
                ((File) item).append(content);
                return true;
            }
        }

        return false;
    }

```

```java
    private final Boolean printFile(final Filename filename) {
        for (final Item item : this.workingDirectory.getContent()) {
            if (item.getType() == Item.Type.FILE && ((File) item).getName().equals(filename.getName())
                    && ((File) item).getExtension().equals(filename.getExtension())) {
                System.out.println(((File) item).getContent());
                return true;
            }
        }

        return false;
    }

    private final void printHelp() {
        System.out.println("[INFO]: You can use one of the following commands:");
        System.out.println("\tla\t\t\t-\tPrint all items in the working directory");
        System.out.println("\tcd [name]\t\t-\tChange working directory");
        System.out.println("\twd\t\t\t-\tPrint working directory");
        System.out.println("\tdu\t\t\t-\tPrint disk usage");
        System.out.println("\tmf [name]\t\t-\tMake new file");
        System.out.println("\trf [name]\t\t-\tRemove file");
        System.out.println("\tmd [name]\t\t-\tMake new directory");
        System.out.println("\trd [name]\t\t-\tRemove directory");
        System.out.println("\tff [name] [content]\t-\tFill the file with additional content");
        System.out.println("\tpf [name]\t\t-\tPrint file content");
        System.out.println("\th\t\t\t-\tPrint help message");
        System.out.println("\tq\t\t\t-\tQuit program");
    }
}

```

Вспомогательные скрипты:

Java > VMemory > compile.bat

```bat
javac -cp ".\src\;.\lib\SizeOf.jar" -d ".\bin" .\src\vmem\*.java
@REM javadoc -cp ".\src\;.\lib\SizeOf.jar" -d ".\docs" .\src\vmem\*.java
jar -cvfm vmem.jar ".\resources\manifest.txt" -C ".\bin" .\
@REM jar -cvfm vmem.jar ".\resources\manifest.txt" -C ".\bin" .\ -C ".\docs" .\ -C ".\lib" .\

```

Java > VMemory > start.bat

```bat
java -javaagent:".\lib\SizeOf.jar" --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED -jar vmem.jar

```

Java > VMemory > src > vmem > Master.java > ...

```java
package vmem;

import java.io.IOException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public final class Master {
    Run | Debug
    public static final void main(final String[] args) throws Exception {
        final String FILENAME = new String(
                "C:\\Users\\User\\Desktop\\Handler.ser");
        final File FILE = new File(FILENAME);

        Handler handler = null;

        if (FILE.exists()) {
            try {
                FileInputStream fileIn = new FileInputStream(FILENAME);
                ObjectInputStream objectIn = new ObjectInputStream(fileIn);

                handler = (Handler) objectIn.readObject();
                System.out.println("Data deserialized from " + FILENAME + '.');

                objectIn.close();
                fileIn.close();

                handler.start();
            } catch (IOException exception) {
                exception.printStackTrace();
            } catch (Exception exception) {
                exception.printStackTrace();
            }
        } else {
            System.out.println("Getting ready your virtual memory pack...");

            final Long KB = 1024l, PACK_SIZE = KB * 1440l;

            handler = new Handler(PACK_SIZE);
            handler.start();
        }

        try {
            FileOutputStream fileOut = new FileOutputStream(FILENAME);
            ObjectOutputStream objectOut = new ObjectOutputStream(fileOut);

            objectOut.writeObject(handler);

            objectOut.close();
            fileOut.close();

            System.out.println("Data serialized in " + FILENAME + '.');
        } catch (IOException exception) {
            exception.printStackTrace();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
}
```

Результаты тестирования:

```
PS C:\Users\User\Documents\Visual Studio Code\Java\VMemory> .\compile.bat

C:\Users\User\Documents\Visual Studio Code\Java\VMemory>javac -cp ".\src\;.\lib\SizeOf.jar" -d ".\bin" .\src\vmem\*.java

C:\Users\User\Documents\Visual Studio Code\Java\VMemory>jar -cvfm vmem.jar ".\resources\manifest.txt" -C ".\bin" .\
added manifest
adding: vmem/(in = 0) (out= 0)(stored 0%)
adding: vmem/Command.class(in = 789) (out= 429)(deflated 45%)
adding: vmem/Directory.class(in = 1534) (out= 766)(deflated 50%)
adding: vmem/File.class(in = 1919) (out= 891)(deflated 53%)
adding: vmem/Filename.class(in = 730) (out= 383)(deflated 47%)
adding: vmem/Folder.class(in = 502) (out= 339)(deflated 32%)
adding: vmem/Handler$1.class(in = 1178) (out= 679)(deflated 42%)
adding: vmem/Handler$CommandType.class(in = 1498) (out= 810)(deflated 45%)
adding: vmem/Handler.class(in = 10237) (out= 5066)(deflated 50%)
adding: vmem/Item$Type.class(in = 875) (out= 508)(deflated 41%)
adding: vmem/Item.class(in = 904) (out= 490)(deflated 45%)
adding: vmem/Master.class(in = 2158) (out= 1205)(deflated 44%)
adding: vmem/Type.class(in = 770) (out= 461)(deflated 40%)
PS C:\Users\User\Documents\Visual Studio Code\Java\VMemory>
```

```
PS C:\Users\User\Documents\Visual Studio Code\Java\VMemory> .\start.bat

C:\Users\User\Documents\Visual Studio Code\Java\VMemory>java -javaagent:".\lib\SizeOf.jar"
--add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED -jar vmem.jar
JAVAGENT: call premain instrumentation for class SizeOf
Getting ready your virtual memory pack...
[INFO]: Pack created, size = 1474560 bytes.
[INFO]: You can use one of the following commands:
        la                      -       Print all items in the working directory
        cd [name]               -       Change working directory
        wd                      -       Print working directory
        du                      -       Print disk usage
        mf [name]               -       Make new file
        rf [name]               -       Remove file
        md [name]               -       Make new directory
        rd [name]               -       Remove directory
        ff [name] [content]     -       Fill the file with additional content
        pf [name]               -       Print file content
        h                       -       Print help message
        q                       -       Quit program
> la
> wd
/
> du
[INFO]: Storage (bytes): 1474560
Used (bytes): 120; Used (%): 0.01
Free (bytes): 1474440; Used (%): 99.99
> mf file
[WARNING]: Creating a file without extension!
> mf file.txt
> la
f       /       file            128 bytes
f       /       file    .txt    152 bytes
> rf file
> la
f       /       file    .txt    152 bytes
> du
[INFO]: Storage (bytes): 1474560
Used (bytes): 272; Used (%): 0.02
Free (bytes): 1474288; Used (%): 99.98
> md dir
> la
f       /       file    .txt    152 bytes
d       /       dir             0 items
> cd dir
> wd
/dir
> cd ..
> wd
/
> la
f       /       file    .txt    152 bytes
d       /       dir             0 items
> du
[INFO]: Storage (bytes): 1474560
Used (bytes): 392; Used (%): 0.03
Free (bytes): 1474168; Used (%): 99.97
```

```
> cd dir
> mf file.txt
> la
f       /dir    file    .txt    152 bytes
> ff file.txt content_content_content
> la
f       /dir    file    .txt    176 bytes
> cd ..
> la
f       /       file    .txt    152 bytes
d       /       dir             1 items
> du
[INFO]: Storage (bytes): 1474560
Used (bytes): 536; Used (%): 0.04
Free (bytes): 1474024; Used (%): 99.96
> rd dir
> la
f       /       file    .txt    152 bytes
> du
[INFO]: Storage (bytes): 1474560
Used (bytes): 272; Used (%): 0.02
Free (bytes): 1474288; Used (%): 99.98
> ff file.txt content_content_content
> pf file.txt
content_content_content
> ff file.txt _12345
> pf file.txt
content_content_content_12345
> la
f       /       file    .txt    184 bytes
> rf file.txt
> la
> cd dir
[ERROR]: The directory 'dir" does not exist!
> cd
[ERROR]: Invalid parameters with "cd" command! Must be 2!
> mf
[ERROR]: Invalid parameters with "mf" command! Must be 2!
> ff
[ERROR]: Invalid parameters with "ff" command! Must be 3!
> ff file
[ERROR]: Invalid parameters with "ff" command! Must be 3!
> ff file content
[ERROR]: The file 'file" does not exist or you don't have enough space!
> mf file
[WARNING]: Creating a file without extension!
> pf
[ERROR]: Invalid parameters with "pf" command! Must be 2!
> pf file

> la
f       /       file            128 bytes
> du
[INFO]: Storage (bytes): 1474560
Used (bytes): 248; Used (%): 0.02
Free (bytes): 1474312; Used (%): 99.98
> rf file
> du
[INFO]: Storage (bytes): 1474560
Used (bytes): 120; Used (%): 0.01
Free (bytes): 1474440; Used (%): 99.99
> wd
/
> la
> h
[INFO]: You can use one of the following commands:
        la                      -       Print all items in the working directory
        cd [name]               -       Change working directory
        wd                      -       Print working directory
        du                      -       Print disk usage
        mf [name]               -       Make new file
        rf [name]               -       Remove file
        md [name]               -       Make new directory
        rd [name]               -       Remove directory
        ff [name] [content]     -       Fill the file with additional content
        pf [name]               -       Print file content
        h                       -       Print help message
        q                       -       Quit program
> q
Data serialized in C:\Users\User\Desktop\Handler.ser.
PS C:\Users\User\Documents\Visual Studio Code\Java\VMemory>
```

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