

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

ОТЧЕТ  
Лабораторная работа №4

Выполнил:  
Студент 4 курса  
Группа АС-50  
Барболин М.О.  
Проверил:  
Крощенко А.А.

Брест 2020

## Вариант 1

Цель работы: приобрести базовые навыки работы с файловой системой в Java

### Задание 1

Реализовать указанный класс, включив в него вспомогательный внутренний класс или классы. Реализовать 2-3 метода (на выбор). Продемонстрировать использование реализованных классов.

- 1) Создать класс Notepad (записная книжка) с внутренним классом или классами, с помощью объектов которого могут храниться несколько записей на одну дату.

Код:

```
package com.company;
import java.io.BufferedReader; import java.io.IOException;
import java.io.InputStreamReader;
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Notepad notepad = new Notepad();
        notepad.show();
        try {
            notepad.add();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        try {
            notepad.add();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        notepad.show();
        try {
            notepad.deleteDate();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        notepad.show();
    }

    public static class Notepad{
        List<Date> dates;
        public Notepad() {
            this.dates = new ArrayList<>();
        }

        public class Date{
            List<String> note = new ArrayList<>();
            int dd;
            int mm;
            int year;
            public Date(int dd, int mm, int year) {
                this.dd = dd;
                this.mm = mm;
                this.year = year;
            }
        }
    }
}
```

```

    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass())
            return false;
        Date date = (Date) o;
        return dd == date.dd &&
            mm == date.mm &&
            year == date.year;
    }
    @Override
    public int hashCode() {
        return Objects.hash(dd, mm, year);
    }
}

void add() throws IOException {
    Date date = dateInitialization("Добавления");
    String note = noteInitialization();
    Boolean addition = false;
    for (int i = 0; i < this.dates.size() && !addition ; i++) {
        if(date.equals(dates.get(i))) {
            dates.get(i).note.add(note);
            addition = true;
        }
    }
    if (!addition){
        date.note.add(note);
        this.dates.add(date);
    }
}

public void deleteDate() throws IOException{
    Date delete = dateInitialization("Удаления");
    Iterator<Date> iterator = dates.iterator();
    while(iterator.hasNext()){
        Date date = iterator.next();
        if(delete.equals(date))
            iterator.remove();
    }
}

public void show(){
    for (int i = 0; i < this.dates.size(); i++) {
        System.out.printf("Дата
%d/%d/%d\n",dates.get(i).dd,dates.get(i).mm,dates.get(i).year);
        for (int j = 0; j < dates.get(i).note.size(); j++) {
            System.out.println((j+1)+" ". +dates.get(i).note.get(j));
        }
    }
}

private Date dateInitialization(String log) throws IOException{
    BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
    int dd, mm, year;
    System.out.println("Операция : " + log);
    System.out.println("Введите день ");
    dd = Integer.parseInt(reader.readLine());
    System.out.println("Введите месяц ");
    mm = Integer.parseInt(reader.readLine());
    System.out.println("Введите год");
    year = Integer.parseInt(reader.readLine());
    Date date = new Date(dd,mm,year);
    return date;
}

private String noteInitialization() throws IOException {

```

```

        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
        System.out.println("Введите запись");
        String note = reader.readLine();
        return note;
    }
}

```

Скриншот:

The screenshot shows an IDE window with the following components:

- Project Explorer:** Shows the project structure with 'lab4\_var1\_task1' and 'External Libraries'.
- Run Console:** Displays the execution output of the 'Main' class. The output shows the program prompting for date and record, and then displaying the entered values and the resulting date and record entries.

The output in the Run Console is as follows:

```

C:\Users\Максим\.jdk\openjdk-15.0.1\bin\java.exe
Операция : Добавления
Введите день
18
Введите месяц
04
Введите год
2000
Введите запись
Я пришел в этот мир
Операция : Добавления
Введите день
31
Введите месяц
12
Введите год
2019
Введите запись
Корона вирус пришел во всех нас(
Дата 18/4/2000
1. Я пришел в этот мир
Дата 31/12/2019
1. Корона вирус пришел во всех нас(
Операция : Удаления
Введите день
31
Введите месяц
12
Введите год
2019
Дата 18/4/2000
1. Я пришел в этот мир

Process finished with exit code 0

```

## Задание 2

Реализовать агрегирование. При создании класса агрегируемый класс объявляется как атрибут(локальная переменная, параметр метода). Включить в каждый класс 2-3 метода на выбор. Продемонстрировать использование разработанных классов.

1) Создать класс Строка, используя классы Слово, Символ.

Код:

```
package com.company;
import java.util.ArrayList;
public class Main {

    public static void main(String[] args) {
        Str str=new Str();
        str.addStr("Test start first try");
        str.addStr("Start when it's work");
        str.showString();
    }
    //вложенный класс строка. формируется из массива слов. слова разделяются
    пробелами
    public static class Str
    {
        public ArrayList<Word> str=new ArrayList<Word>();
        public void addStr(String words)
        {
            Word temp=new Word();
            temp.addWords(words);
            str.add(temp);
        }
        public void showString()
        {
            for(Word item:str)
            {
                for(Word.Symbol item2:item.word)
                {
                    System.out.print(item2.symbol);
                }
                System.out.println();
            }
        }
    }
    //класс слово формируется на основе добавления новых символов до момента
    встречи " " - пробел
    //который будет обозначать начало нового слова
    public static class Word {
        //public String words;

        public ArrayList<Symbol> word = new ArrayList<Symbol>();

        public Word() {
        }

        public void addWords(String str) {
            for (String words:str.split(" ")) {
                for(String symbol:words.split(""))
                {
                    word.add(new Symbol(symbol));
                }
                word.add(new Symbol(" "));
            }
        }
    }
}
```

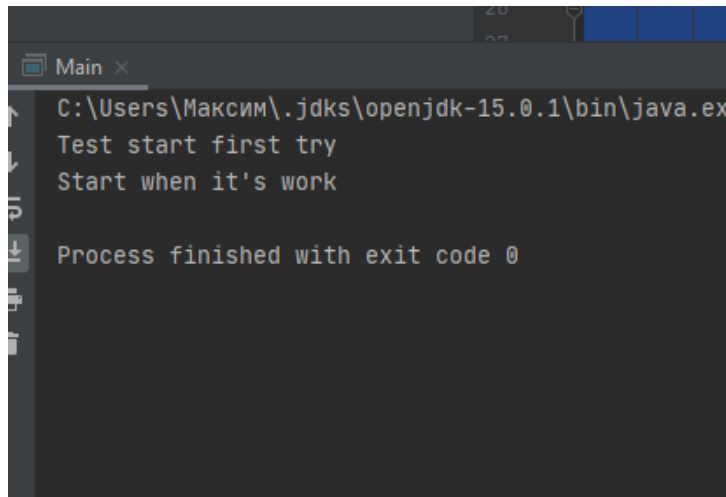
```

//есть класс символа на основе которого будет формироваться строка
//как совокупность символом до пробела
public class Symbol {
    String symbol;

    public Symbol(String symbol) {
        this.symbol = symbol;
    }
}
}
}

```

## Скриншоты:



## Задание 3

Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы.

1) Система Факультатив. Преподаватель объявляет запись на Курс. Студент записывается на Курс, обучается и по окончании Преподаватель выставляет Оценку, которая сохраняется в Архиве. Студентов, Преподавателей и Курсов при обучении может быть несколько.

## Код:

```

package com.company;
public abstract class Human {
    protected String name;
    protected Integer age;
    //класс, для расширения студентом и преподавателем. содержит ФИО и
    возраст
    protected Human(String name, Integer age)
    {
        if (name.isBlank() || age < 0)
            throw new RuntimeException("You must declare both name and age");
        this.name = name;
        this.age = age;
    }
}

```

```

package com.company;
import java.util.ArrayList;
import java.util.List;

public class Student extends Human {

    public Student(String name, Integer age, String group) {
        super(name, age);
        this.group = group;
    }

    private String group;
    private final List<Course> courses = new ArrayList<>();

    public void signForCourse(String courseName) {
        if (courseName.isBlank())
            throw new RuntimeException("You must declare course name");
        Course.courses.forEach(course -> {
            if (course.getCourseName().equals(courseName)) {
                this.courses.add(course);
                course.getStudents().add(this);
            }
        });
    }

    public List<Course> getCourses() {
        return courses;
    }

    @Override
    public String toString() {
        return "Student{" +
            "name='" + name + '\'' +
            ", age=" + age +
            ", group=" + group +
            '}';
    }
}

```

```

package com.company;
import java.util.ArrayList;
import java.util.List;

public class Teacher extends Human {
    protected Teacher(String name, Integer age, String degree) {
        super(name, age);
        this.degree = degree;
    }

    private String degree;
    private final List<Course> courses = new ArrayList<>();

    public Course createCourse(String courseName) {
        if (courseName.isBlank())
            throw new RuntimeException("You must declare course name");
        else {
            Course course = new Course(courseName, this);
            courses.add(course);
            return course;
        }
    }
}

```

```

    public void addTeacher(Teacher teacher, Course course) {
        if(course.getTeachers().contains(this))
            course.getTeachers().add(teacher);
    }

    public void finishCourseAndGiveMarks(Course course) {
        course.getStudents().forEach(student -> {
            course.getArchive().put(student, 7); // 7 - it can be
substituted for anything
        });
    }

    public List<Course> getCourses() {
        return courses;
    }

    @Override
    public String toString() {
        return "Teacher{" +
            "name='" + name + '\'' +
            ", degree='" + degree + '\'' +
            ", age='" + age +
            '\'';
    }
}

```

```

package com.company;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
//создаем класс курса. содержит название курса и привязанного преподавателя
public class Course {
    public Course(String courseName, Teacher teacher) {
        this.courseName = courseName;
        teachers = new ArrayList<>();
        teachers.add(teacher);
        courses.add(this);
    }

    public static List<Course> courses = new ArrayList<>();
    private Map<Student, Integer> archive = new HashMap<>();
    private List<Teacher> teachers;

    private final String courseName;
    private List<Student> students = new ArrayList<>();
    private List<Integer> marks = new ArrayList<>();

    //методы для получения информации "с курса"
    public String getCourseName() {
        return courseName;
    }

    public static List<Course> getCourses() {
        return courses;
    }

    public List<Teacher> getTeachers() {
        return teachers;
    }
}

```



```

    public List<Student> getStudents() {
        return students;
    }

    public Map<Student, Integer> getArchive() {
        return archive;
    }

    @Override
    public String toString() {
        return "Course{" +
            "courseName='" + courseName + '\'' +
            '}';
    }
}

```

```

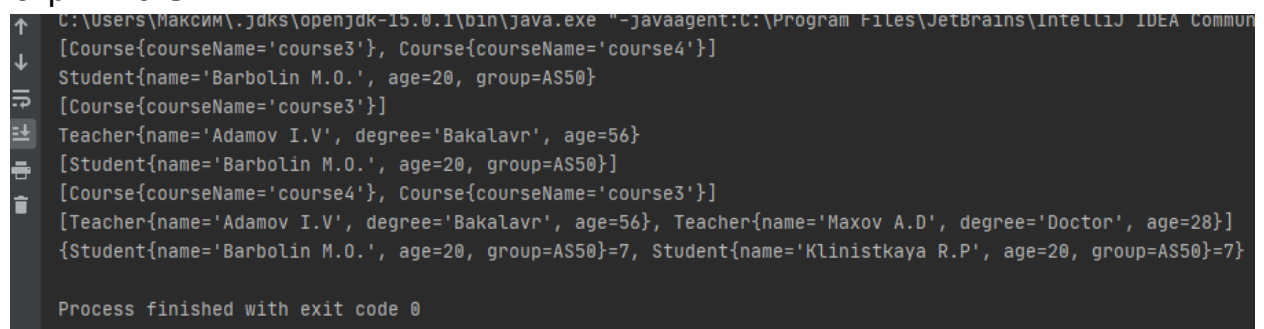
package com.company;
//изображение результатов
public class Main {

    public static void main(String[] args) {
        Teacher teacher = new Teacher("Adamov I.V", 56, "Bakalavr");
        Teacher teacher1 = new Teacher("Maxov A.D", 28, "Doctor");
        Course course = teacher.createCourse("course3");
        Course course1 = teacher1.createCourse("course4");
        Student student = new Student("Barbolin M.O.", 20, "AS50");
        Student student1 = new Student("Klinistkaya R.P", 20, "AS50");
        student.signForCourse("course3");
        System.out.println(Course.getCourses());
        System.out.println(student);
        System.out.println(teacher.getCourses());
        System.out.println(teacher);
        student1.signForCourse("course4");
        System.out.println(course.getStudents());
        student1.signForCourse("course3");
        System.out.println(student1.getCourses());
        teacher.addTeacher(teacher1, course);
        System.out.println(course.getTeachers());
        teacher.finishCourseAndGiveMarks(course);
        System.out.println(course.getArchive());

    }
}

```

### Скриншоты:



```

C:\Users\Максим\jdk\openjdk-15.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Commu
[Course{courseName='course3'}, Course{courseName='course4'}]
Student{name='Barbolin M.O.', age=20, group=AS50}
[Course{courseName='course3'}]
Teacher{name='Adamov I.V', degree='Bakalavr', age=56}
[Student{name='Barbolin M.O.', age=20, group=AS50}]
[Course{courseName='course4'}, Course{courseName='course3'}]
[Teacher{name='Adamov I.V', degree='Bakalavr', age=56}, Teacher{name='Maxov A.D', degree='Doctor', age=28}]
{Student{name='Barbolin M.O.', age=20, group=AS50}=7, Student{name='Klinistkaya R.P', age=20, group=AS50}=7}

Process finished with exit code 0

```

Вывод: в ходе лабораторной ознакомился с принципами ООП в java.