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

ОТЧЕТ Лабораторная работа №4

Выполнил:
Студент 4 курса
Группа АС-50
Бойченко А. Д.
Проверил:
Крощенко А.А.

Брест 2020

Лабораторная работа №4

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.

```
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;
    }
  }
}
```

```
Введите месяц
1
Введите год
2021
Введите запись
New Year
Операция : Добавления
Введите день
2
Введите месяц
1
Введите год
2021
Введите запись
В 2020 было лучше
Дата 1/1/2021
1. New Year
Дата 2/1/2021
1. В 2020 было лучше
Операция : Удаления
Введите день
1
Введите месяц
1
Введите год
2021
Дата 2/1/2021
1. В 2020 было лучше

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.

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

  public static void main(String[] args) {
    Str str=new Str();
    str.addStr("Hello World");
    str.addStr("How are you?");
    str.showString();
  }
  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();
      }
    }
  }
  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(" "));
```

```
        }
    }


    public class Symbol {
        String symbol;

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

```
Hello World
How are you?

Process finished with exit code 0
```

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.

Main.java

```
public class Main {
  public Main() {
  }

  public static void main(String[] args) {
    Teacher teacher = new Teacher("t1", 25, "b");
    Teacher teacher1 = new Teacher("t2", 28, "m");
    Course course = teacher.createCourse("course1");
    Course course1 = teacher1.createCourse("course2");
    Student student = new Student("s1", 20, "AS51");
    Student student1 = new Student("s2", 21, "AS50");
    student.signForCourse("course1");
    System.out.println(Course.getCourses());
    System.out.println(student);
    System.out.println(teacher.getCourses());
    System.out.println(teacher);
    student1.signForCourse("course1");
    System.out.println(course.getStudents());
    student1.signForCourse("course2");
    System.out.println(student1.getCourses());
    teacher.addTeacher(teacher1, course);
```

```
        System.out.println(course.getTeachers());
        teacher.finishCourseAndGiveMarks(course);
        System.out.println(course.getArchive());
    }
}
```

Course.java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

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<>();

    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 + '\'' +
                '}';
    }
}
```

Human.java

```
public abstract class Human {
    protected String name;
    protected Integer age;

    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;
    }
}
```

Student.java

```
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 +
                '}';
    }
}
```

Teacher.java

```
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 +
                '}';
    }
}
```

```
[Course{courseName='course1'}, Course{courseName='course2'}]
Student{name='s1', age=20, group=AS51}
[Course{courseName='course1'}]
Teacher{name='t1', degree='b', age=25}
[Student{name='s1', age=20, group=AS51}, Student{name='s2', age=21, group=AS50}]
[Course{courseName='course1'}, Course{courseName='course2'}]
[Teacher{name='t1', degree='b', age=25}, Teacher{name='t2', degree='m', age=28}]
{Student{name='s1', age=20, group=AS51}=7, Student{name='s2', age=21, group=AS50}=7}
```