МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Лабораторная работа 4
По дисциплине “СПП”

Выполнил: Вальчук А.А.
Проверил: Крощенко А.А.

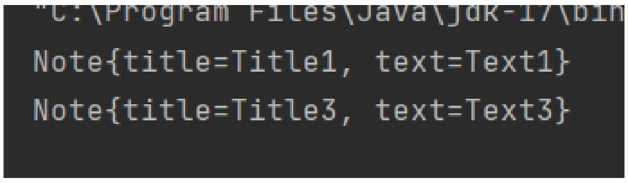
Брест 2021

Цель работы: приобрести практические навыки в области объектно-ориентированного проектирования

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

```
public class Notepad {
   private String owner;
   private final HashMap<LocalDate, List<Note>> notes = new HashMap<>();

   public Notepad() {}

   public Notepad(String owner) {
      this.owner = owner;
   }

   public String getOwner() {
      return owner;
   }

   public void setOwner(String owner) {
      this.owner = owner;
   }

   public void addNote(LocalDate date, Note note) {
      List<Note> existingNotes = notes.get(date);

      if (existingNotes == null) {
         notes.put(date, new ArrayList<>(List.of(note)));

         return;
      }

      existingNotes.add(note);
   }

   public List<Note> getNotesByDate(LocalDate date) {
      return notes.get(date);
   }

   public static class Note {
      private String title;
      private String text;

      public Note() {}

      public Note(String title, String text) {
         this.title = title;
         this.text = text;
      }
```

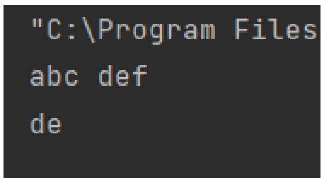
```
        public String getTitle() {
            return title;
        }

        public String getText() {
            return text;
        }

        public void setTitle(String title) {
            this.title = title;
        }

        public void setText(String text) {
            this.text = text;
        }

        @Override
        public String toString() {
            return String.format("Note{title=%s, text=%s}", title, text);
        }
    }
}

public class Task1 {

    public static void main(String[] args) {
        Notepad notepad = new Notepad("Mr. Potato");

        notepad.addNote(LocalDate.parse("2021-12-14"), new Notepad.Note("Title1", "Text1"));
        notepad.addNote(LocalDate.parse("2021-12-15"), new Notepad.Note("Title2", "Text2"));
        notepad.addNote(LocalDate.parse("2021-12-14"), new Notepad.Note("Title3", "Text3"));

        for (Notepad.Note note: notepad.getNotesByDate(LocalDate.parse("2021-12-14"))) {
            System.out.println(note);
        }
    }
}
```



```
"C:\Program Files\Java\jdk-17\bin
Note{title=Title1, text=Text1}
Note{title=Title3, text=Text3}
```

Задание 2. Создать класс Строка, используя классы Слово, Символ.

```
public class Line {
    private final List<Word> words;

    public Line() {
        words = new ArrayList<>();
    }

    public Line(List<Word> words) {
        this.words = words;
    }

    public void addWord(Word word) {
        words.add(word);
    }

    public void removeWord() {
        if (words.size() == 0) {
            throw new IllegalStateException();
        }

        words.remove(words.size() - 1);
    }

    @Override
    public String toString() {
        return words.stream().map(Word::toString).collect(Collectors.joining(" "));
    }
}

public class Symbol {
    private final Character value;

    public Symbol(char value) {
        this.value = value;
    }

    @Override
    public String toString() {
        return value.toString();
    }
}


public class Word {
    private final List<Symbol> symbols;

    public Word() {
        symbols = new ArrayList<>();
    }

    public Word(List<Symbol> symbols) {
        this.symbols = symbols;
    }

    public void addSymbol(Symbol symbol) {
        symbols.add(symbol);
    }

    public void removeSymbol() {
        if (symbols.size() == 0) {
            throw new IllegalStateException();
        }

        symbols.remove(symbols.size() - 1);
    }

    @Override
    public String toString() {
        return symbols.stream().map(Symbol::toString).collect(Collectors.joining());
    }
}

public class Task2 {

    public static void main(String[] args) {
            Word word1 = new Word();
        Word word2 = new Word();

        word1.addSymbol(new Symbol('a'));
        word1.addSymbol(new Symbol('b'));
        word1.addSymbol(new Symbol('c'));

        word2.addSymbol(new Symbol('d'));
        word2.addSymbol(new Symbol('e'));
        word2.addSymbol(new Symbol('f'));

        System.out.println(new Line(List.of(word1, word2)));

        word2.removeSymbol();

        System.out.println(word2);
    }
}
```



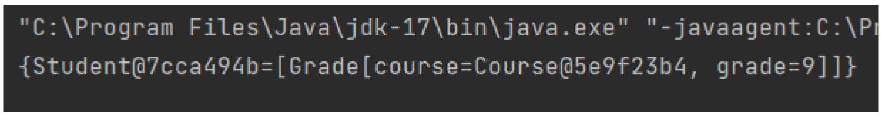
```
"C:\Program Files
abc def
de
```

Задание 3. 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.

```
public class Archive extends HashMap<Student, List<Grade>> {}

public class Course {
    private final Teacher teacher;
    private final List<Student> students = new ArrayList<>();

    public Course(Teacher teacher) {
        this.teacher = teacher;
    }

    public Teacher getTeacher() {
        return teacher;
    }

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

public class ElectiveSystem {
    private final List<Teacher> teachers = new ArrayList<>();
    private final List<Student> students = new ArrayList<>();
    private final List<Course> courses = new ArrayList<>();
    private final Archive archive = new Archive();

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

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

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

    public Archive getArchive() {
        return archive;
    }
}

public record Grade(Course course, int grade) {

    public Course getCourse() {
        return course;
    }

    public int getGrade() {
        return grade;
    }
}

public interface ICourseCreator {
    Course createCourse();
```

```
}

public class Person {
   private String name;

   public String getName() {
      return name;
   }

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

public class Student extends Person {
   public void enroll(Course course) {
      course.getStudents().add(this);
   }
}

public class Teacher extends Person implements ICourseCreator {
   @Override
   public Course createCourse() {
      return new Course(this);
   }

   public void rate(Student student, Course course, Archive archive, int gradeValue) throws
Exception {
      if (course.getStudents().stream().noneMatch(s -> s == student)) {
         throw new Exception();
      }

      Grade grade = new Grade(course, gradeValue);

      List<Grade> grades = archive.get(student);

      if (grades == null) {
         archive.put(student, new ArrayList<>(List.of(grade)));

         return;
      }

      grades.add(grade);
   }
}

public class Task3 {

   public static void main(String[] args) throws Exception {
            ElectiveSystem electives = new ElectiveSystem();

      electives.getTeachers().add(new Teacher());
```

```
        electives.getStudents().add(new Student());
        electives.getCourses().add(electives.getTeachers().get(0).createCourse());
        electives.getStudents().get(0).enroll(electives.getCourses().get(0));
        electives.getTeachers().get(0).rate(electives.getStudents().get(0), electives.getCourses().get(0),
electives.getArchive(), 9);

        System.out.println(electives.getArchive());
    }
}
```



```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Pr
{Student@7cca494b=[Grade[course=Course@5e9f23b4, grade=9]]}
```