

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

ОТЧЁТ
по лабораторной работе №4

Выполнил
студент 3 курса
группы ПО-9
Ничингер Кирилл Александрович

Проверил:
Крощенко А. А.

Брест 2024

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

Вариант 4

Задание 1: создать класс Зачетная Книжка с внутренним классом, с помощью объектов которого можно хранить информацию о сессиях, зачетах, экзаменах.

Код программы:

```
package Task1;

import java.time.Year;
import java.util.ArrayList;

public class ZachetnayaKnizhka {
    private ArrayList<Session> sessions;
    public ZachetnayaKnizhka() {
        sessions = new ArrayList<>();
    }
    public void addSession(Session session) {
        if (session != null) {
            for (Session ses : sessions) {
                if (ses.getYear() == session.getYear()
                    && ses.getSemester() == session.getSemester())
                    break;
            }
            sessions.add(session);
        }
    }
    public enum Semester {
        FIRST("первый семестр"), SECOND("второй семестр");
        private final String name;
        Semester(String name) {
            this.name = name;
        }
        public String getName() {
            return name;
        }
    }
    public enum Subject {
        MATHEMATICS("математика"), PHYSICS("физика"), HISTORY("история"),
        MODERN_PROGRAMMING_PLATFORMS("современные платформы программирования");
        private final String name;
        Subject(String name) {
            this.name = name;
        }
        public String getName() {
            return name;
        }
    }
}

public static class Session {
    private ArrayList<Exam> exams;
    private ArrayList<Zachet> zachets;
    private Year year;
    private Semester semester;
```

```

    {
        this.exams = new ArrayList<>();
        this.zachets = new ArrayList<>();
        year = Year.now();
        semester = Semester.FIRST;
    }
    public Session(Year year, Semester semester){
        this.year = year;
        this.semester = semester;
    }
    public void addExam(Exam exam){
        if (exam != null){
            for (Exam ex : exams){
                if (ex.getSubject() == exam.getSubject())
                    break;
            }
            exams.add(exam);
        }
    }
    public void addZachet(Zachet zachet){
        if (zachet != null){
            for (Zachet zach : zachets){
                if (zach.getSubject() == zachet.getSubject())
                    break;
            }
            zachets.add(zachet);
        }
    }
}

public static class KnowledgeAssessment {
    private Subject subject;
    private int mark;
    private KnowledgeAssessment(){}
    public KnowledgeAssessment(Subject subject, int mark){
        this.subject = subject;
        this.mark = (mark >= 1 && mark <= 10) ? mark : 1;
    }
    public void setMark(int mark) {
        if (mark >= 1 && mark <= 10)
            this.mark = mark;
    }
}

public static class Exam extends KnowledgeAssessment {
    private Exam(){
        super();
    }
    public Exam(Subject subject, int mark){
        super(subject, mark);
    }
}

public static class Zachet extends KnowledgeAssessment {
    private Zachet(){
        super();
    }
    public Zachet(Subject subject, int mark){
        super(subject, mark);
    }
}
}

```

Входные данные:

```
public static void main(String[] args) {
    ZachetnayaKnizhka zk = new ZachetnayaKnizhka();

    Session session = new Session(Year.now(), Semester.SECOND);

    session.addExam(new Exam(Subject.PHYSICS, 10));
    session.addExam(new Exam(Subject.MATHEMATICS, 9));
    session.addZachet(new Zachet(Subject.HISTORY, 8));

    zk.addSession(session);

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

Результат работы программы:

D:\SDK\JDK\bin\java.exe "-javaagent

сессии:

сессия 2024 второй семестр:

экзамены:

физика: оценка 10

математика: оценка 9

зачёты:

история: оценка 8

Process finished with exit code 0

Задание 2: создать класс Текст, используя классы Страница, Слово.

Код программы:

```
public class Word {
    private String word;

    public Word(String word) {
        this.word = word;
    }
    ...
}

public class Page {
    private ArrayList<Word> words;

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

    public Page addWord(Word word) {
        words.add(word);
        return this;
    }

    public void removeWord(Word word) {
        words.remove(word);
    }
    ...
}
```

```

public class Text {
    ArrayList<Page> pages;

    public Text() {
        pages = new ArrayList<>();
    }

    public Text addPage(Page page) {
        pages.add(page);
        return this;
    }

    public void removePage(Page page) {
        pages.remove(page);
    }
    ...
}

```

Входные данные:

```

public static void main(String[] args) {
    Page firstPage = new Page();
    firstPage.addWord(new Word("Lorem "))
        .addWord(new Word("ipsum, \n"))
        .addWord(new Word("dolor "))
        .addWord(new Word("consectetur.));

    Page secondPage = new Page();
    secondPage.addWord(new Word("Excepteur "))
        .addWord(new Word("sint, \n"))
        .addWord(new Word("occaecat "))
        .addWord(new Word("cupidatat.));

    Text text = new Text();
    text.addPage(firstPage).addPage(secondPage);

    System.out.println(text);
}

```

Результат работы программы:

D:\SDK\JDK\bin\java.exe

Page 1

Lorem ipsum,
dolor consectetur.

Page 2

Excepteur sint,
occaecat cupidatat.

Задание 3: создать систему Вступительные экзамены. Абитуриент регистрируется на Факультет, сдает Экзамены. Преподаватель выставляет Оценку. Система подсчитывает средний балл и определяет Абитуриентов, зачисленных в учебное заведение.

Код программы:

```
abstract public class Person {
    private String name;
    private final int id;
    private static int nextPersonId = 1;

    public Person(String name){
        this.name = name;
        id = nextPersonId++;
    }
}

public class Enrollee extends Person{
    public Enrollee(String name) {
        super(name);
    }
}

public class Teacher extends Person{
    public Teacher(String name) {
        super(name);
    }
    public int getMark(){
        return (int) (Math.random() * 100);
    }
}

public class Faculty {
    private String name;
    private List<Subject> requiredExams = new ArrayList<>();
    private List<EnrolleeData> registeredEnrolles = new ArrayList<>();
    private List<EnrolleeData> evolvedEnrolles = new ArrayList<>();

    public static class EnrolleeData {
        private Enrollee enrollee;
        private Map<Subject, Integer> examMarks = new HashMap<>();

        EnrolleeData(Enrollee enrollee){
            this.enrollee = enrollee;
        }
        public void setExamMark(Subject subject, int mark){
            if (!examMarks.containsKey(subject))
                examMarks.put(subject, mark);
        }
        public int getExamScore(){
            int examScore = 0;
            for (int examMark : examMarks.values()){
                examScore += examMark;
            }
            return examScore / examMarks.size();
        }
    }
}
```

```

Faculty(String name, List<Subject> requiredExams){
    this.name = name;
    this.requiredExams.addAll(requiredExams.stream().distinct().toList());
}

public Faculty registerEnrollee(Enrollee enrollee){
    if (!registeredEnrolles.contains(enrollee)){
        EnrolleeData enrolleeData = new EnrolleeData(enrollee);
        registeredEnrolles.add(enrolleeData);
    }
    return this;
}

public Faculty conductExam(Subject subject, Teacher teacher){
    if (requiredExams.contains(subject))
        for (EnrolleeData enrolleeData : registeredEnrolles)
            if (!enrolleeData.havePassedExam(subject))
                enrolleeData.setExamMark(subject, teacher.getMark());
    return this;
}

public void evolveFromEnrolleeToStudent(int passingScore){
    for (int i = 0; i < registeredEnrolles.size(); i++) {
        EnrolleeData enrolleeData = registeredEnrolles.get(i);
        if (enrolleeData.getExamMarks().size() == requiredExams.size()
            && enrolleeData.getExamScore() >= passingScore){
            registeredEnrolles.remove(i--);
            evolvedEnrolles.add(enrolleeData);
        }
    }
}
}

```

Входные данные:

```

public static void main(String[] args) {
    String facultyName = "CreativnoyeImya";
    List<Subject> requiredExams = new ArrayList<>(List.of(Subject.HISTORY,
Subject.MATHEMATICS));
    Faculty faculty = new Faculty(facultyName, requiredExams);

    faculty.registerEnrollee(new Enrollee("Petya"))
        .registerEnrollee(new Enrollee("Kirill"));

    Teacher teacher = new Teacher("Prepodavatel");

    faculty.conductExam(Subject.HISTORY, teacher)
        .conductExam(Subject.MATHEMATICS, teacher);

    System.out.println("All enrolles:");
    for (Faculty.EnrolleeData enrolleeData : faculty.getRegisteredEnrolles()) {
        System.out.println(enrolleeData);
    }

    faculty.evolveFromEnrolleeToStudent(70);

    System.out.println("\nEnrolled people:");
    for (Faculty.EnrolleeData enrolleeData : faculty.getEvolvedEnrolles()) {
        System.out.println(enrolleeData);
    }
}

```

Результат работы программы:

```
D:\SDK\JDK\bin\java.exe "-javaagent
```

```
All enrolles:
```

```
enrollee:
```

```
    id=1
```

```
    name='Petya'
```

```
    exams:
```

```
        история - 59
```

```
        математика - 5
```

```
enrollee:
```

```
    id=2
```

```
    name='Kirill'
```

```
    exams:
```

```
        история - 60
```

```
        математика - 82
```

```
Enrolled people:
```

```
enrollee:
```

```
    id=2
```

```
    name='Kirill'
```

```
    exams:
```

```
        история - 60
```

```
        математика - 82
```

```
Process finished with exit code 0
```

Вывод: я приобрёл практические навыки в области объектно-ориентированного проектирования.