***d*гМИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

### ЛАБОРАТОРНАЯ РАБОТА № 1

по дисциплине: «Введение в разработку программного обеспечения»

на тему: **«**Развитие представлений о разработке программ: от спагетти-кода к методологии структурного программирования. Объектно-ориентированное программирование в *Java***»**

Выполнил: студент гр. ИТИ-21

Павлович Д. Н.

Принял преподаватель:

Карась О. В.

Гомель 2025

**Цель работы:** изучить технологии проектных решений в *Java*.

**Задание**:

Спроектировать проектное решение для приложения, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы *equals*(), *hashCode*(), *toString*().

Вариант 5.

Создать объект класса Текст, используя класс Абзац. Методы: дополнить текст, вывести на консоль текст, заголовок текста.

**Ход работы:**

Класс *Paragraph*:  
Ответственность: хранение содержимого одного абзаца  
Методы: *getContent*(), *setContent*() - доступ к содержимому (управляют состоянием объекта); *toString*() - строковое представление (переопределен для удобства вывода); *equals*(), *hashCode*() - для корректного сравнения объектов (переопределены по требованию задания)

Класс *Text*:

Ответственность: композиция абзацев и управление текстом как целым

Методы: *addParagraph*() - добавление абзаца (принадлежит классу, так как изменяет состав текста); *getTitle*(), *setTitle*() - доступ к заголовку (управляют состоянием объекта); *toString*(), *equals*(), *hashCode*() - переопределены по требованию задания.

Для начала был создан класс *Paragraph*, представляющий отдельный абзац текста. В этом классе реализовано поле *content* для хранения текста абзаца, конструктор для инициализации содержимого, методы *getContent*() и *setContent*() для доступа к содержимому. Также были переопределены методы *toString*() для возврата содержимого абзаца, *equals*() и *hashCode*() для корректного сравнения объектов.

Далее был разработан класс *Text*, который представляет собой текст, состоящий из нескольких абзацев. В данном классе присутствуют поле *title* для хранения заголовка текста и поле *paragraphs* для хранения массива абзацев. Реализован конструктор для инициализации заголовка, метод *addParagraph*() для добавления новых абзацев, метод *displayText*() для вывода всего текста на консоль, а также методы *getTitle*() и *setTitle*() для работы с заголовком. Были переопределены методы *toString*(), *equals*() и *hashCode*().

Затем был создан демонстрационный класс *TextDemo* для показа работы программы. В нем выполнялось создание объекта *Text* с заголовком, добавление нескольких абзацев, вывод текста на консоль, демонстрация работы методов *getTitle*(), добавление нового абзаца и повторный вывод, а также показ работы *equals*() и *hashCode*().

Программа была протестирована на различных сценариях, включая создание текста с несколькими абзацами, корректный вывод заголовка и содержимого, добавление новых абзацев в существующий текст, сравнение различных текстовых объектов и проверку корректности *hashCode*().

В результате программа успешно реализует требуемую функциональность, обеспечивая создание текста из абзацев, добавление новых абзацев, вывод текста на консоль, работу с заголовком текста и корректно переопределенные методы *equals*(), *hashCode*() и *toString*().

На рисунке 1 показана структура папки, в которой и велось выполнение лабораторной работы.

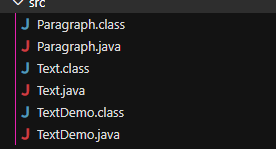


Рисунок 1 - Архитектура папки

Результат выполнения программы представлен на рисунке 2.

Листинг программы представлен в приложении А.



Рисунок 1 – Результат выполнения программы

**Вывод**: в ходе лабораторной работы были изучены основы объектно-ориентированного программирования в *Java*. Реализована программа для управления коллекцией книг с использованием ООП-подхода. Применены основные принципы ООП: создание классов, инкапсуляция данных через геттеры и сеттеры, организация взаимодействия между классами. Программа демонстрирует эффективное использование массивов объектов и методов для работы с ними.

***ПРИЛОЖЕНИЕ А***

***Листинг файла Studenk.java:***

*import java.time.LocalDate;*

*import java.time.format.DateTimeFormatter;*

*public class Student {*

*// Поля класса - информация о студенте*

*private int id;*

*private String lastName;*

*private String firstName;*

*private String middleName;*

*private LocalDate birthDate;*

*private String address;*

*private String phone;*

*private String faculty;*

*private int course;*

*private String group;*

*// Конструктор - создает нового студента со всеми данными*

*public Student(int id, String lastName, String firstName, String middleName,*

*LocalDate birthDate, String address, String phone,*

*String faculty, int course, String group) {*

*this.id = id;*

*this.lastName = lastName;*

*this.firstName = firstName;*

*this.middleName = middleName;*

*this.birthDate = birthDate;*

*this.address = address;*

*this.phone = phone;*

*this.faculty = faculty;*

*this.course = course;*

*this.group = group;*

*}*

*// Геттеры - методы для получения значений полей*

*public int getId() { return id; }*

*public String getLastName() { return lastName; }*

*public String getFirstName() { return firstName; }*

*public String getMiddleName() { return middleName; }*

*public LocalDate getBirthDate() { return birthDate; }*

*public String getAddress() { return address; }*

*public String getPhone() { return phone; }*

*public String getFaculty() { return faculty; }*

*public int getCourse() { return course; }*

*public String getGroup() { return group; }*

*// Сеттеры - методы для изменения значений полей*

*public void setId(int id) { this.id = id; }*

*public void setLastName(String lastName) { this.lastName = lastName; }*

*public void setFirstName(String firstName) { this.firstName = firstName; }*

*public void setMiddleName(String middleName) { this.middleName = middleName; }*

*public void setBirthDate(LocalDate birthDate) { this.birthDate = birthDate; }*

*public void setAddress(String address) { this.address = address; }*

*public void setPhone(String phone) { this.phone = phone; }*

*public void setFaculty(String faculty) { this.faculty = faculty; }*

*public void setCourse(int course) { this.course = course; }*

*public void setGroup(String group) { this.group = group; }*

*// Метод для красивого вывода информации о студенте*

*@Override*

*public String toString() {*

*DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");*

*return String.format(*

*"%-3d %-12s %-12s %-12s %-12s %-20s %-15s %-15s %-2d %-8s",*

*id,*

*lastName,*

*firstName,*

*middleName,*

*birthDate.format(formatter), // Форматируем дату в красивый вид*

*address.length() > 20 ? address.substring(0, 17) + "..." : address, // Обрезаем длинный адрес*

*phone,*

*faculty,*

*course,*

*group*

*);*

*}*

*}*

***Листинг файла StudentManager.java:***

*import java.util.Arrays;*

*import java.util.Comparator;*

*import java.util.HashMap;*

*import java.util.Map;*

*public class StudentManager {*

*private Student[] students;*

*// Конструктор - создает менеджер студентов*

*public StudentManager(Student[] students) {*

*this.students = students;*

*}*

*// Геттер для получения массива студентов*

*public Student[] getStudents() { return students; }*

*// Сеттер для установки массива студентов*

*public void setStudents(Student[] students) { this.students = students; }*

*// Получить студентов определенного факультета*

*public Student[] getStudentsByFaculty(String faculty) {*

*return Arrays.stream(students)*

*.filter(student -> student.getFaculty().equalsIgnoreCase(faculty))*

*.toArray(Student[]::new);*

*}*

*// Группировка студентов по факультетам и курсам*

*public Map<String, Map<Integer, Student[]>> getStudentsByFacultyAndCourse() {*

*Map<String, Map<Integer, Student[]>> result = new HashMap<>();*

*for (Student student : students) {*

*String faculty = student.getFaculty();*

*int course = student.getCourse();*

*// Создаем структуру если ее нет*

*result.putIfAbsent(faculty, new HashMap<>());*

*result.get(faculty).putIfAbsent(course, new Student[0]);*

*// Добавляем студента в массив*

*Student[] current = result.get(faculty).get(course);*

*Student[] updated = Arrays.copyOf(current, current.length + 1);*

*updated[current.length] = student;*

*result.get(faculty).put(course, updated);*

*}*

*return result;*

*}*

*// Получить студентов родившихся после указанного года*

*public Student[] getStudentsBornAfterYear(int year) {*

*return Arrays.stream(students)*

*.filter(student -> student.getBirthDate().getYear() > year)*

*.toArray(Student[]::new);*

*}*

*// Получить студентов определенной группы*

*public Student[] getStudentsByGroup(String group) {*

*return Arrays.stream(students)*

*.filter(student -> student.getGroup().equalsIgnoreCase(group))*

*.toArray(Student[]::new);*

*}*

*// Получить студентов в алфавитном порядке*

*public Student[] getStudentsInAlphabeticalOrder() {*

*Student[] sortedStudents = students.clone();*

*Arrays.sort(sortedStudents, Comparator.comparing(Student::getLastName)*

*.thenComparing(Student::getFirstName)*

*.thenComparing(Student::getMiddleName));*

*return sortedStudents;*

*}*

*// Вывод списка студентов в табличном формате*

*public static void printStudents(Student[] students, String title) {*

*System.out.println("\n" + title + ":");*

*System.out.println("=".repeat(120));*

*// Заголовок таблицы*

*System.out.printf("%-3s %-12s %-12s %-12s %-12s %-20s %-15s %-15s %-2s %-8s%n",*

*"ID", "Фамилия", "Имя", "Отчество", "Дата рожд.", "Адрес",*

*"Телефон", "Факультет", "К", "Группа");*

*System.out.println("-".repeat(120));*

*// Если студентов нет - сообщение*

*if (students.length == 0) {*

*System.out.println("Нет студентов, соответствующих критерию.");*

*} else {*

*// Вывод каждого студента*

*for (Student student : students) {*

*System.out.println(student);*

*}*

*}*

*System.out.println("=".repeat(120));*

*System.out.println("Найдено студентов: " + students.length);*

*}*

*// Вывод студентов с группировкой по факультетам и курсам*

*public void printStudentsByFacultyAndCourse() {*

*Map<String, Map<Integer, Student[]>> data = getStudentsByFacultyAndCourse();*

*System.out.println("\nСТУДЕНТЫ ПО ФАКУЛЬТЕТАМ И КУРСАМ:");*

*System.out.println("=".repeat(120));*

*// Перебор всех факультетов*

*for (String faculty : data.keySet()) {*

*System.out.println("\nФАКУЛЬТЕТ: " + faculty.toUpperCase());*

*System.out.println("-".repeat(60));*

*// Перебор всех курсов на факультете*

*Map<Integer, Student[]> courses = data.get(faculty);*

*for (Integer course : courses.keySet()) {*

*System.out.println("КУРС " + course + ":");*

*printStudents(courses.get(course), "");*

*}*

*}*

*}*

*}*

***Листинг файла Main.java:***

*import java.time.LocalDate;*

*public class Main {*

*public static void main(String[] args) {*

*// Создаем массив студентов с тестовыми данными*

*Student[] studentsArray = {*

*new Student(1, "Иванов", "Иван", "Иванович",*

*LocalDate.of(2000, 5, 15), "ул. Ленина, 15", "+79161234567",*

*"Информатика", 3, "ИВТ-301"),*

*new Student(2, "Петров", "Петр", "Петрович",*

*LocalDate.of(2001, 8, 22), "пр. Мира, 25", "+79169876543",*

*"Экономика", 2, "ЭК-202"),*

*new Student(3, "Сидорова", "Анна", "Алексеевна",*

*LocalDate.of(1999, 12, 3), "ул. Центральная, 10", "+79165554433",*

*"Информатика", 4, "ИВТ-401"),*

*new Student(4, "Кузнецов", "Дмитрий", "Сергеевич",*

*LocalDate.of(2002, 3, 18), "ул. Школьная, 5", "+79167778899",*

*"Экономика", 1, "ЭК-101"),*

*new Student(5, "Смирнов", "Андрей", "Владимирович",*

*LocalDate.of(2000, 7, 30), "ул. Садовая, 8", "+79163332211",*

*"Информатика", 3, "ИВТ-302"),*

*new Student(6, "Алексеев", "Сергей", "Михайлович",*

*LocalDate.of(2001, 1, 25), "пр. Победы, 12", "+79164445566",*

*"Юриспруденция", 2, "ЮР-201")*

*};*

*// Создаем менеджер для работы со студентами*

*StudentManager manager = new StudentManager(studentsArray);*

*// Выводим всех студентов*

*System.out.println("ВСЕ СТУДЕНТЫ:");*

*System.out.println("=".repeat(120));*

*System.out.printf("%-3s %-12s %-12s %-12s %-12s %-20s %-15s %-15s %-2s %-8s%n",*

*"ID", "Фамилия", "Имя", "Отчество", "Дата рожд.", "Адрес",*

*"Телефон", "Факультет", "К", "Группа");*

*System.out.println("-".repeat(120));*

*for (Student student : studentsArray) {*

*System.out.println(student);*

*}*

*System.out.println("=".repeat(120));*

*// Получаем и выводим студентов факультета Информатика*

*Student[] informaticsStudents = manager.getStudentsByFaculty("Информатика");*

*StudentManager.printStudents(informaticsStudents, "СТУДЕНТЫ ФАКУЛЬТЕТА ИНФОРМАТИКА");*

*// Выводим студентов с группировкой по факультетам и курсам*

*manager.printStudentsByFacultyAndCourse();*

*// Получаем и выводим студентов, родившихся после 2000 года*

*Student[] bornAfter2000 = manager.getStudentsBornAfterYear(2000);*

*StudentManager.printStudents(bornAfter2000, "СТУДЕНТЫ, РОДИВШИЕСЯ ПОСЛЕ 2000 ГОДА");*

*// Получаем и выводим студентов группы ИВТ-301*

*Student[] groupIVT301 = manager.getStudentsByGroup("ИВТ-301");*

*StudentManager.printStudents(groupIVT301, "СТУДЕНТЫ ГРУППЫ ИВТ-301");*

*// Получаем и выводим студентов в алфавитном порядке*

*Student[] alphabetical = manager.getStudentsInAlphabeticalOrder();*

*StudentManager.printStudents(alphabetical, "СТУДЕНТЫ В АЛФАВИТНОМ ПОРЯДКЕ");*

*}*

*}*