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

ОТЧЁТ

по лабораторной работе №4

Выполнил:

студент группы ПО-8
Вейгандт И.О.

Проверил:
Крощенко А.А.

Брест 2024

Вариант 5

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

Задание 1:

Создать класс Department (отдел фирмы) с внутренним классом, с помощью объектов которого можно хранить информацию обо всех должностях отдела и обо всех сотрудниках, когда-либо занимавших конкретную должность.

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

public class Department {
    private String name;
    private List<Post> posts;

    public Department(String name) {
        this.name = name;
        this.posts = new ArrayList<>();
    }

    public void addPost(String name) {
        posts.add(new Post(name));
    }

    public int getTotalEmployees() {
        int totalEmployees = 0;
        for (Post post : posts) {
            totalEmployees += post.getEmployees().size();
        }
        return totalEmployees;
    }
    public void displayEmployees() {
        System.out.println("Employees in the " + name + "
department:");
        for (Post post : posts) {
            System.out.println("Position: " +
post.getName() + ", Employees: " + post.getEmployees());
        }
        System.out.println("Total employees in the " + name
+ " department: " + getTotalEmployees());
    }

    public class Post {
        private String name;
        private List<String> employees;
```

```
        public Post(String name) {
            this.name = name;
            this.employees = new ArrayList<>();
        }

        public Post(String name, List<String> employees) {
            this.name = name;
            this.employees = employees;
        }

        public String getName() {
            return name;
        }

        public List<String> getEmployees() {
            return employees;
        }

        public void addEmployee(String employeeName) {
            employees.add(employeeName);
        }


    }

    public static void main(String[] args) {
        Department itDepartment = new Department("IT");
        itDepartment.addPost("Software Developer");
        itDepartment.addPost("System Analyst");

        itDepartment.posts.get(0).addEmployee("Ilya");
        itDepartment.posts.get(0).addEmployee("Semyon");

        itDepartment.posts.get(1).addEmployee("Egor");

        itDepartment.displayEmployees();

    }
}
```

```
C:\Users\veiga\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent
Employees in the IT department:
Position: Software Developer, Employees: [Ilya, Semyon]
Position: System Analyst, Employees: [Egor]
Total employees in the IT department: 3
```

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. Создать класс Абзац, используя класс Слово.

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

public class Paragraph {
    private List<Word> words;

    public Paragraph() {
        this.words = new ArrayList<>();
    }

    public void addWord(String wordText) {
        Word word = new Word(wordText);
        words.add(word);
    }

    public void displayParagraph() {
        System.out.print("Paragraph: ");
        for (Word word : words) {
            System.out.print(word.getWordText() + " ");
        }
        System.out.println();
    }

    public int countWords() {
        return words.size();
    }

    public static void main(String[] args) {
        Paragraph paragraph = new Paragraph();
        paragraph.addWord("my");
        paragraph.addWord("name");
        paragraph.addWord("is");
        paragraph.addWord("Ilya");

        paragraph.displayParagraph();
        System.out.println("Number of words in the
paragraph: " + paragraph.countWords());
```

```
        Word word = paragraph.words.get(3);
        word.capitalizeFirstLetter();
        word.reverseWord();
        word.isPalindrome();

        paragraph.displayParagraph();
    }
}
class Word {
    private String wordText;

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

    public String getWordText() {
        return wordText;
    }

    public void setWordText(String wordText) {
        this.wordText = wordText;
    }

    public void capitalizeFirstLetter() {
        char firstChar =
Character.toUpperCase(wordText.charAt(0));
        wordText = firstChar + wordText.substring(1);
    }
    public void reverseWord(){
        StringBuilder stringBuilder = new
StringBuilder(wordText);
        stringBuilder.reverse();
        wordText = stringBuilder.toString();
    }
    public void isPalindrome(){
        StringBuilder stringBuilder = new
StringBuilder(wordText);
        String reverseWord =
stringBuilder.reverse().toString();
        if (wordText.equals(reverseWord)){
            System.out.println("The " + wordText + " is a
palindrome");
        } else {
            System.out.println("The " + wordText + " is not
a palindrome");
        }
```

```
    }
}
```

```
C:\Users\veiga\.jdks\openjdk-19.0.1\bin\java.exe
Paragraph: my name is Ilya
Number of words in the paragraph: 4
The aylI is not a palindrome
Paragraph: my name is aylI
```

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.

Система Библиотека. Читатель оформляет Заказ на Книгу. Система осуществляет поиск в Каталоге. Библиотекарь выдает Читателю Книгу на абонемент или в читальный зал. При невозвращении Книги Читателем он может быть занесен Администратором в «черный список».

```
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

public class LibrarySystem {

    public static void main(String[] args) {
        Library library = new Library();

        Book book1 = new Book( "Toyota. Путь к
совершенству", "Цунёси Нодзи");
        Book book2 = new Book("Цунёси Нодзи", "Дэвид
Килпатрик");
        Book book3 = new Book("Чистый код. Создание, анализ
и рефакторинг", "Роберт Мартин");
        Book book4 = new Book("Удивительный мир звука",
"Игорь Клюкин");

        Catalog catalog = new Catalog();

        catalog.addBook(book1);
        catalog.addBook(book2);
        catalog.addBook(book3);
        catalog.addBook(book4);
```

```
        Reader reader1 = new Reader("Илья");
        Reader reader2 = new Reader("Семён");
        Reader reader3 = new Reader("Егор");

        Order order1 =
reader1.placeOrderForReadingRoom(book1);
        Order order2 =
reader2.placeOrderForHome(book2,LocalDate.of(2024, 3,1));
        Order order3 = reader3.placeOrderForHome(book3,
LocalDate.of(2024, 5,5));
        Order order4 =
reader1.placeOrderForReadingRoom(book3);

        LibraryWorker libraryWorker = new
LibraryWorker(library);

        libraryWorker.addOrder(order1);
        libraryWorker.addOrder(order2);
        libraryWorker.addOrder(order3);
        libraryWorker.addOrder(order4);

        libraryWorker.issueBookToReader(order1);
        libraryWorker.issueBookToReader(order2);
        libraryWorker.issueBookToReader(order3);
        libraryWorker.issueBookToReader(order4);

        reader1.returnBook(book1,libraryWorker);
        reader2.returnBook(book2,libraryWorker);
        reader3.returnBook(book3,libraryWorker);

        Order order5 =
reader2.placeOrderForReadingRoom(book3);
        libraryWorker.issueBookToReader(order5);
        libraryWorker.issueBookToReader(order4);
    }
}
class Book {
    private String title;
    private String author;
    private boolean available =true;
    public Book(String title, String author) {
        this.title = title;
        this.author = author;
    }
```

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

    public String getAuthor() {
        return author;
    }

    public boolean isAvailable() {
        return available;
    }

    public void setAvailable(boolean available) {
        this.available = available;
    }
}
class Catalog {
    ArrayList<Book> books;

    public Catalog() {
        this.books = new ArrayList<>();
    }
    public void addBook(Book book){
        books.add(book);
    }
    public Book searchBook(String title){
        for (Book book: books){
            if(book.getTitle().equals(title)){
                return book;
            }
        }
        return null;
    }
}
class Order{
    private Reader reader;
    private Book book;
    boolean isForReadingRoom;

    private LocalDate returnDate = null;
    public Order(Reader reader, Book book, boolean
isForReadingRoom) {
        this.reader = reader;
        this.book = book;
        this.isForReadingRoom = isForReadingRoom;
    }
```

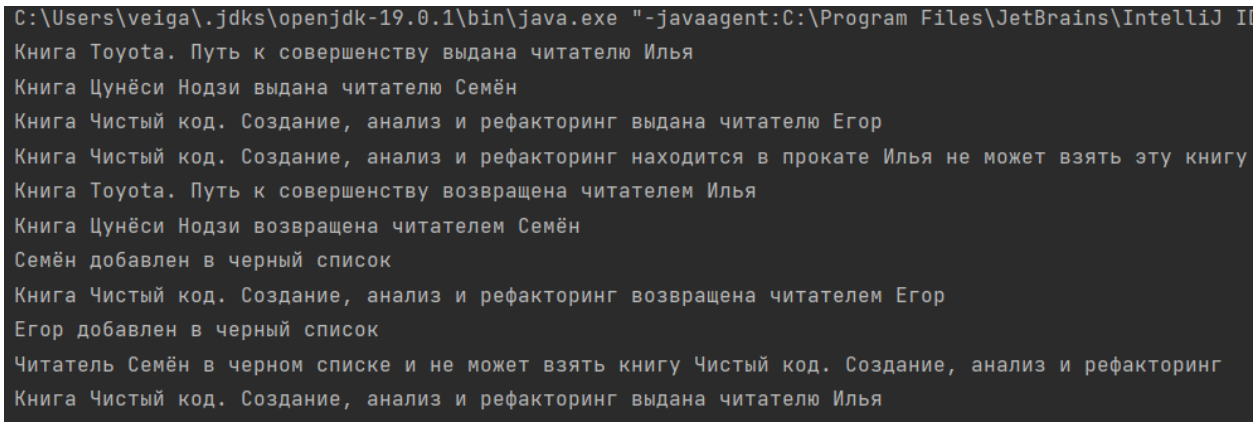
```
    public Order(Reader reader, Book book, boolean
isForReadingRoom, LocalDate date) {
        this.reader = reader;
        this.book = book;
        this.isForReadingRoom = isForReadingRoom;
        this.returnDate = date;
    }

    public Reader getReader() {
        return reader;
    }

    public void setReader(Reader reader) {
        this.reader = reader;
    }

    public Book getBook() {
        return book;
    }

    public void setBook(Book book) {
        this.book = book;
    }

    public boolean isForReadingRoom() {
        return isForReadingRoom;
    }

    public void setForReadingRoom(boolean forReadingRoom) {
        isForReadingRoom = forReadingRoom;
    }

    public LocalDate getReturnDate() {
        return returnDate;
    }

    public void setReturnDate(LocalDate returnDate) {
        this.returnDate = returnDate;
    }
}
class Library{
    Catalog catalog;
    protected List<Reader> blackList;
    public List<Order> orders;

    public Library(Catalog catalog) {
```

```
        this.catalog = catalog;
    }
    public Library() {
        this.blackList = new ArrayList<>();
        this.orders = new ArrayList<>();
    }

    public List<Reader> getBlackList() {
        return blackList;
    }

    public void setBlackList(List<Reader> blackList) {
        this.blackList = blackList;
    }

    public  List<Order> getOrders() {
        return orders;
    }
    public  void setOrders(List<Order> orders) {
        this.orders = orders;
    }
}
class Reader extends Library{
    List<Book> books;
    String name;
    boolean inBlackList = false;



    public Reader(String name) {
        this.name = name;
        this.books = new ArrayList<>();
    }

    Order placeOrderForReadingRoom(Book book){
        if(!inBlackList && book.isAvailable()) {
            Order newOrder = new Order(this, book, true);
            return newOrder;
        }else if(inBlackList) {
            System.out.println("Читатель " + name + " в
черном списке и не может взять книгу " + book.getTitle());
            return null;
        }else {
            System.out.println("Книга " + book.getTitle() +
" находится в прокате " + name+ "не может ее взять");
            return null;
```

```
        }
    }
    Order placeOrderForHome(Book book, LocalDate
returnDate){
        if(!inBlackList && book.isAvailable()) {
            Order newOrder = new Order(this, book, false,
returnDate);
            return newOrder;
        }else if(inBlackList) {
            System.out.println("Читатель " + name + " в
черном списке и не может взять книгу " + book.getTitle());
            return null;
        }else {
            System.out.println("Книга " + book.getTitle() +
" находится в прокате " + name+ "не может ее взять");
            return null;
        }
    }
    public void returnBook(Book book,LibraryWorker
libraryWorker) {
        if (books.contains(book)) {
            books.remove(book);
            System.out.println("Книга " + book.getTitle() +
" возвращена читателем " + name);
            libraryWorker.bookReception(book);
        }
    }


    public List<Book> getBooks() {
        return books;
    }

    public void setBooks(List<Book> books) {
        this.books = books;
    }

    public String getName() {
        return name;
    }

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

    public boolean isInBlackList() {
```

```
        return inBlackList;
    }

    public void setInBlackList(boolean inBlackList) {
        this.inBlackList = inBlackList;
    }
}
class LibraryWorker extends Library{
    Library library;

    public LibraryWorker(Library library) {
        this.library = library;
    }

    public void issueBookToReader(Order order) {
        if(order != null){
            Reader reader = order.getReader();
            Book book = order.getBook();
            if(!reader.inBlackList && book.isAvailable()) {
                reader.books.add(book);
                book.setAvailable(false);
                System.out.println("Книга " +
book.getTitle()+ " выдана читателю " + reader.name);
            }else if(reader.inBlackList) {
                System.out.println("Читатель " +
reader.name + " находится в черном списке и не может взять
книгу" + book.getTitle());
            }else {
                System.out.println("Книга " +
book.getTitle() + " находится в прокате " + reader.name +"
не может взять эту книгу");
            }
        }
    }
    public void bookReception(Book book) {
        for(Order order : library.orders){
            if(order.getBook().equals(book)){
                book.setAvailable(true);
                if(!order.isForReadingRoom() &&
!returnedOnTime(order.getReturnDate())){
                    Administrator administrator = new
Administrator();

administrator.addToBlackList(order.getReader());
                }
            }
```

```
        }


    }
    boolean returnedOnTime(LocalDate returnDate){
        LocalDate currentDate = LocalDate.now();
        if(currentDate.isBefore(returnDate) ||
currentDate.equals(returnDate)){
            return true;
        }
        return false;
    }

    void addOrder(Order order){
        library.orders.add(order);
    }
    class Administrator {
        public void addToBlackList(Reader reader) {
            reader.setInBlackList(true);
            blackList.add(reader);
            System.out.println(reader.name + " добавлен в
черный список");
        }
    }

}
```



Вывод: приобрёл практические в области объектно-ориентированного проектирования.