МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”  **КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Отчёт

по лабораторной работе №4

Выполнил:

студент группы ПО-9

Солышко Дмитрий Андреевич

Проверил:

Крощенко А. А.

Брест 2024

**Вариант 6**

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

# Задание 1

Реализовать указанный класс, включив в него вспомогательный внутренний класс или классы.

Реализовать 2-3 метода (на выбор). Продемонстрировать использование реализованных классов.

6) Создать класс Catalog (каталог) с внутренним классом, с помощью объектов которого можно

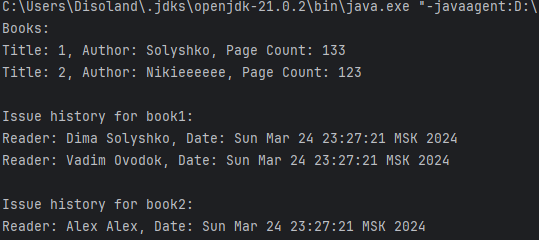
хранить информацию об истории выдач книги читателям.

**Код программы:**

**Task\_01.java:**

import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
class Catalog {  
 private List<Book> books;  
  
 public Catalog() {  
 books = new ArrayList<>();  
 }  
  
 public void addBook(Book book) {  
 books.add(book);  
 }  
  
 public void displayBooks() {  
 for (Book book : books) {  
 System.*out*.println(book);  
 }  
 }  
  
 public class IssueHistory {  
 private Book book;  
 private List<String> readers;  
 private List<Date> dates;  
  
 public IssueHistory(Book book) {  
 this.book = book;  
 readers = new ArrayList<>();  
 dates = new ArrayList<>();  
 }  
  
 public void issueBook(String reader, Date date) {  
 readers.add(reader);  
 dates.add(date);  
 }  
  
 public void displayIssueHistory() {  
 for (int i = 0; i < readers.size(); i++) {  
 System.*out*.println("Reader: " + readers.get(i) + ", Date: " + dates.get(i));  
 }  
 }  
 }  
}  
  
class Book {  
 private String title;  
 private String author;  
 private int pageCount;  
  
 public Book(String title, String author, int pageCount) {  
 this.title = title;  
 this.author = author;  
 this.pageCount = pageCount;  
 }  
  
 @Override  
 public String toString() {  
 return "Title: " + title + ", Author: " + author + ", Page Count: " + pageCount;  
 }  
}  
  
public class Task\_01 {  
 public static void main(String[] args) {  
 Catalog catalog = new Catalog();  
  
 Book book1 = new Book("1", "Solyshko", 133);  
 Book book2 = new Book("2", "Nikieeeeee", 123);  
  
 catalog.addBook(book1);  
 catalog.addBook(book2);  
  
 Catalog.IssueHistory issueHistory1 = catalog.new IssueHistory(book1);  
 issueHistory1.issueBook("Dima Solyshko", new Date());  
 issueHistory1.issueBook("Vadim Ovodok", new Date());  
  
 Catalog.IssueHistory issueHistory2 = catalog.new IssueHistory(book2);  
 issueHistory2.issueBook("Alex Alex", new Date());  
  
 System.*out*.println("Books:");  
 catalog.displayBooks();  
  
 System.*out*.println("\nIssue history for book1:");  
 issueHistory1.displayIssueHistory();  
  
 System.*out*.println("\nIssue history for book2:");  
 issueHistory2.displayIssueHistory();  
 }  
}

# Результат работы программы:



# 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.

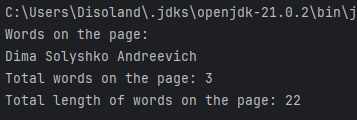
6) Создать класс Страница, используя класс Слово.

**Код программы**

**Task\_02.java:**

import java.util.ArrayList;  
  
class Word {  
 private String word;  
  
 public Word(String word) {  
 this.word = word;  
 }  
  
 public String getWord() {  
 return word;  
 }  
  
 public int length() {  
 return word.length();  
 }  
}  
  
class Page {  
 private ArrayList<Word> words;  
  
 public Page(ArrayList<Word> words) {  
 this.words = words;  
 }  
  
 public void addWord(Word word) {  
 words.add(word);  
 }  
  
 public int wordCount() {  
 return words.size();  
 }  
  
 public void displayWords() {  
 System.*out*.println("Words on the page:");  
 for (Word word : words) {  
 System.*out*.print(word.getWord() + " ");  
 }  
 System.*out*.println();  
 }  
 public int totalLength() {  
 int totalLengthOfWords = 0;  
 for (Word word : words) {  
 totalLengthOfWords += word.length();  
 }  
 return totalLengthOfWords;  
 }  
}  
  
public class Task\_02 {  
 public static void main(String[] args) {  
 Word word1 = new Word("Dima");  
 Word word2 = new Word("Solyshko");  
  
 ArrayList<Word> pageWords = new ArrayList<>();  
 pageWords.add(word1);  
 pageWords.add(word2);  
 Page page = new Page(pageWords);  
  
 Word word3 = new Word("Andreevich");  
 page.addWord(word3);  
  
 page.displayWords();  
  
 System.*out*.println("Total words on the page: " + page.wordCount());  
 System.*out*.println("Total length of words on the page: " + page.totalLength());  
 }  
}

**Результат работы программы:**

****

# 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.

6) Система **Телефонная станция. Абонент** оплачивает **Счет** за разговоры и **Услуги**, может

попросить **Администратора** сменить номер и отказаться от услуг. **Администратор** изменяет

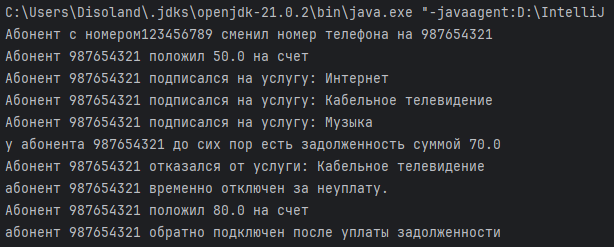
номер, **Услуги** и временно отключает **Абонента** за неуплату.

**Код программы**

**Task\_03.java:**

import java.sql.SQLOutput;  
import java.util.ArrayList;  
import java.util.List;  
  
class Bill {  
 private double amount;  
  
 public Bill(double amount) {  
 this.amount = amount;  
 }  
  
 public double getAmount() {  
 return amount;  
 }  
  
 public void setAmount(double amount) {  
 this.amount = amount;  
 }  
}  
  
class Service {  
 private String name;  
 private double price;  
  
 public Service(String name, double price) {  
 this.name = name;  
 this.price = price;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
}  
  
class Subscriber {  
 public String phoneNumber;  
 public List<Service> services;  
 public Bill bill;  
 public boolean isActive;  
  
 public Subscriber(String phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 this.services = new ArrayList<>();  
 this.bill = new Bill(0);  
 this.isActive = true;  
 }  
  
 public void requestPhoneNumberChange(Administrator administrator, String newNumber) {  
 administrator.changePhoneNumber(this, newNumber);  
 }  
  
 public void requestService(Administrator administrator, Service service) {  
 administrator.requestService(this, service);  
 }  
  
 public void cancelService(Administrator administrator, Service service) {  
 administrator.cancelService(this, service);  
 }  
  
 public void payBill(double amount) {  
 this.bill.setAmount(this.bill.getAmount() - amount);  
 System.*out*.println("Абонент " + this.phoneNumber + " положил " + amount + " на счет");  
 if(!this.isActive && !checkUnpaidBill())  
 {  
 this.isActive = true;  
 System.*out*.println  
 ("абонент " + this.phoneNumber + " обратно подключен после уплаты задолженности");  
 }  
 }  
  
 public void accountAmount()  
 {  
 if(checkUnpaidBill())  
 {  
 System.*out*.println  
 ("у абонента " + this.phoneNumber + " до сих пор есть задолженность суммой " + this.bill.getAmount());  
 }  
 else  
 {  
 System.*out*.println  
 ("у абонента " + this.phoneNumber + " имеется остаток на счете суммой " + (-this.bill.getAmount()));  
 }  
 }  
  
 public boolean checkUnpaidBill() {  
 return this.bill.getAmount() > 0;  
 }  
}  
  
class Administrator {  
 public void changePhoneNumber(Subscriber subscriber, String newNumber) {  
 System.*out*.println("Абонент с номером" + subscriber.phoneNumber + " сменил номер телефона на " + newNumber);  
 subscriber.phoneNumber = newNumber;  
 }  
  
 public void requestService(Subscriber subscriber, Service service) {  
 subscriber.services.add(service);  
 subscriber.bill.setAmount(subscriber.bill.getAmount() + service.getPrice());  
 System.*out*.println("Абонент " + subscriber.phoneNumber + " подписался на услугу: " + service.getName());  
 }  
  
 public void cancelService(Subscriber subscriber, Service service) {  
 if (subscriber.services.contains(service)) {  
 subscriber.services.remove(service);  
 subscriber.bill.setAmount(subscriber.bill.getAmount() - service.getPrice());  
 System.*out*.println("Абонент " + subscriber.phoneNumber + " отказался от услуги: " + service.getName());  
 }  
 }  
  
 public void temporarilyDisableSubscriber(Subscriber subscriber) {  
 if (subscriber.checkUnpaidBill()) {  
 subscriber.isActive = false;  
 System.*out*.println("абонент " + subscriber.phoneNumber + " временно отключен за неуплату.");  
 }  
 }  
}  
  
class TelephoneStation {  
 private List<Subscriber> subscribers;  
  
 public TelephoneStation() {  
 this.subscribers = new ArrayList<>();  
 }  
  
 public void addSubscriber(Subscriber subscriber) {  
 this.subscribers.add(subscriber);  
 }  
  
 public List<Subscriber> getSubscribers() {  
 return subscribers;  
 }  
}  
  
public class Task\_03 {  
 public static void main(String[] args) {  
 TelephoneStation telephoneStation = new TelephoneStation();  
  
 Service service1 = new Service("Интернет", 30.0);  
 Service service2 = new Service("Кабельное телевидение", 50.0);  
 Service service3 = new Service("Музыка", 40.0);  
 Subscriber subscriber1 = new Subscriber("123456789");  
  
 telephoneStation.addSubscriber(subscriber1);  
  
 Administrator administrator = new Administrator();  
  
 subscriber1.requestPhoneNumberChange(administrator, "987654321");  
  
 subscriber1.payBill(50);  
  
 subscriber1.requestService(administrator, service1);  
 subscriber1.requestService(administrator, service2);  
 subscriber1.requestService(administrator, service3);  
  
 subscriber1.accountAmount();  
  
 subscriber1.cancelService(administrator, service2);  
  
 administrator.temporarilyDisableSubscriber(subscriber1);  
  
 subscriber1.payBill(80);  
 }  
}

**Результат работы программы:**



**Вывод:** яприобрёл практические навыки в области объектно-ориентированного проектирования