Задание к лабораторной работе №1

Создать консольное приложение, удовлетворяющее следующим требованиям:

* Использовать возможности ООП: классы, наследование, полиморфизм, инкапсуляция.
* Определить иерархию классов (абстрактных классов, интерфейсов) соответствующую вашей предметной области (от 5 классов/интерфейсов).
* Каждый класс должен иметь отражающее смысл название и информативный состав.
* Определить конструкторы и методы setТип(), getТип(), toString(), equals(), hashcode().
* Наследование должно применяться только тогда, когда это имеет смысл.

Рекомендации при проектировании классов

При создании класса следует придерживаться некоторых правил. **Принцип единственной ответственности**. Каждый класс должен иметь простое назначение. Решать в идеале единственную задачу. Классу следует давать такое имя, чтобы его пользователю была понятна роль класса в пакете или приложении.

Если класс отвечает за хранение информации, то функциональность работы с этой информацией должна быть базовой. Манипулированием информацией через объект должны заниматься другие классы, которых может оказаться достаточно много.

Класс должен быть разработан так, чтобы внесение в него изменений было относительно простой задачей.

Код конструктора должен заниматься только инициализацией объекта.

Следует избегать вызовов из конструктора других методов, за исключением final, static, private. Иначе такой метод может быть переопределен в подкласс и исказить процесс инициализации объекта.

Использовать инкапсуляцию нестатических и неконстантных полей.

Классы-сущности должны применять для доступа к полям классов, хранящих информацию, корректные методы типа get, set, is, а также желательно реализовать методы equals(), hashCode(), clone(), toString() и имплементировать интерфейсы Comparable и Serializable при необходимости.

Если разрабатываемый класс кажется сложным, следует разбить его на несколько простых.

Класс Автор

import java.util.Objects;  
  
public class Author {  
 private String name;//Фамилия  
 private int birthday;// Год рождения  
 private String email;// почта  
  
 public Author() {  
 }  
  
 public Author(String name, int birthday, String email) {  
 this.name = name;  
 this.birthday = birthday;  
 this.email = email;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getBirthday() {  
 return birthday;  
 }  
  
 public void setBirthday(int birthday) {  
 this.birthday = birthday;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Author)) return false;  
 Author author = (Author) o;  
 return birthday == author.birthday && name.equals(author.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, birthday);  
 }  
  
 @Override  
 public String toString() {  
 return "Author{" +  
 "name='" + name + '\'' +  
 ", birthday=" + birthday +  
 ", email='" + email + '\'' +  
 '}';  
 }  
}

Класс книга

import java.util.Objects;  
  
public class Book {  
 private String name;// название  
 private int quantity;// количество  
 private Author author;// автор  
  
 public Book() {  
 }  
  
 public Book(String name, int quantity, Author author) {  
 this.name = name;  
 this.quantity = quantity;  
 this.author = author;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getQuantity() {  
 return quantity;  
 }  
  
 public void setQuantity(int quantity) {  
 this.quantity = quantity;  
 }  
  
 public Author getAuthor() {  
 return author;  
 }  
  
 public void setAuthor(Author author) {  
 this.author = author;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Book)) return false;  
 Book book = (Book) o;  
 return name.equals(book.name) && Objects.*equals*(author, book.author);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, author);  
 }  
  
 @Override  
 public String toString() {  
 return "Book{" +  
 "name='" + name + '\'' +  
 ", quantity=" + quantity +  
 ", author=" + author +  
 '}';  
 }  
}

Класс Читатель

import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class Reader {  
 private int id;//идентификатор  
 private String name;//фамилия  
 private List<Book> books;// список взятых книг  
  
 public Reader() {  
  
 }  
  
 public Reader(int id, String name) {  
 this.id = id;  
 this.name = name;  
 books = new ArrayList<>();  
 }  
  
 public Reader(int id, String name, List<Book> books) {  
 this.id = id;  
 this.name = name;  
 this.books = books;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public List<Book> getBooks() {  
 return books;  
 }  
  
 public void setBooks(List<Book> books) {  
 this.books = books;  
 }  
  
 public void addBookToReader(Book book){ // добавить книгу в список взятых  
 books.add(book);  
 }  
  
 public void removeBookFromReader(String name){ // удалить взятую книгу  
 int index = -1;  
 for(Book b:books){  
 if(b.getName().equals(name)) index = books.indexOf(b);  
 }  
 books.remove(index);  
 }  
  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Reader)) return false;  
 Reader reader = (Reader) o;  
 return id == reader.id && name.equals(reader.name);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(id, name);  
 }  
  
 @Override  
 public String toString() {  
 return "Reader{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", books=" + books +  
 '}';  
 }  
}

Класс библиотека

mport java.util.ArrayList;  
import java.util.List;  
  
public class Library {  
 private List<Book> bookList; // список книг в библиотеке  
 private List<Reader> readerList;// список читателей библиотеки  
  
 public Library() {  
 bookList = new ArrayList<>();  
 }  
  
  
  
 public Library(List<Book> bookList, List<Reader> readerList) {  
 this.bookList = bookList;  
 this.readerList = readerList;  
 }  
  
 public List<Book> getBookList() {  
 return bookList;  
 }  
  
 public void setBookList(List<Book> bookList) {  
 this.bookList = bookList;  
 }  
  
 public List<Reader> getReaderList() {  
 return readerList;  
 }  
  
 public void setReaderList(List<Reader> readerList) {  
 this.readerList = readerList;  
 }  
  
 public void addBook(Book book){  
  
 bookList.add(book);  
 }  
  
 public void addReader(Reader reader){  
 readerList.add(reader);  
 }  
  
}

Класс Main

import java.util.Scanner;  
  
import static java.lang.System.*exit*;  
  
public class Main {  
  
 static Library *library*;  
  
 public static void printMenu(String[] options){  
 for (String option : options){  
 System.*out*.println(option);  
 }  
 System.*out*.print("Choose your option : ");  
 }  
  
 public static void main(String[] args) {  
  
 *library* = new Library();  
  
 String[] options = {"1- Add Book ",  
 "2- Print all books from library ",  
 "3- Add Reader",  
 "4 -Print all readers",  
 "5- Exit",  
 };  
 Scanner scanner = new Scanner(System.*in*);// класс Scanner для ввода данных из консоли  
 int option = 1;  
  
 while (option != 5){  
 *printMenu*(options);  
 try {  
 option = scanner.nextInt();  
 scanner.nextLine();  
 switch (option){  
 case 1: *addBook*(scanner); break;  
 case 2: *printAllBooks*(); break;  
 case 3: *addReader*(scanner); break;  
 case 4: *printAllReaders*(); break;  
 case 5: *exit*(0);  
 }  
 }  
 catch (Exception ex){  
 System.*out*.println("Please enter an integer value between 1 and " + options.length);  
 scanner.nextInt();  
 }  
 }  
 scanner.close();// закрытие потока  
  
  
  
 }  
  
 // Options  
 private static void addBook(Scanner scanner) {  
 System.*out*.println(" Enter Author ");  
 System.*out*.println(" Enter name ");  
  
 String authorName = scanner.nextLine();  
  
 System.*out*.println(" Enter birthday ");  
 int birthday = scanner.nextInt();  
  
 System.*out*.println(" Enter email ");  
 scanner.nextLine();  
 String email = scanner.nextLine();  
  
 Author author = new Author(authorName,birthday,email);  
  
 System.*out*.println(" Enter Book ");  
 System.*out*.println(" Enter name ");  
 String bookName = scanner.nextLine();  
 System.*out*.println(" Enter quantity ");  
 int quantity = scanner.nextInt();  
 scanner.nextLine();  
  
 Book book = new Book(bookName,quantity,author);  
 *library*.addBook(book);  
  
  
  
 System.*out*.println(book.toString());  
  
 }  
 private static void printAllBooks() {  
 System.*out*.println("Books in library");  
 for(Book book: *library*.getBookList()){  
  
 System.*out*.println(book.getName());  
 System.*out*.println(book.getQuantity());  
 System.*out*.println(book.getAuthor());  
 System.*out*.println();  
 }  
 }  
 private static void addReader(Scanner scanner) {  
 System.*out*.println("Enter reader");  
 System.*out*.println(" Enter id ");  
 int id = scanner.nextInt();  
 scanner.nextLine();  
 System.*out*.println(" Enter name ");  
 String name = scanner.nextLine();  
  
 Reader reader = new Reader(id,name);  
 *library*.addReader(reader);  
  
 }  
 private static void printAllReaders() {  
 System.*out*.println("Readers in library");  
 for(Reader reader: *library*.getReaderList()){  
  
 System.*out*.println(reader.getId());  
 System.*out*.println(reader.getName());  
 System.*out*.println(reader.getBooks().toString());  
 System.*out*.println();  
 }  
 }  
  
}

Варианты заданий

1. Цветочница. Определить иерархию цветов. Создать несколько объектов-цветов. Собрать букет (используя аксессуары) с определением его стоимости. Провести сортировку цветов в букете на основе уровня свежести. Найти цветок в букете, соответствующий заданному диапазону длин стеблей.

2. Новогодний подарок. Определить иерархию конфет и прочих сладостей.

Создать несколько объектов-конфет. Собрать детский подарок с определением его веса. Провести сортировку конфет в подарке на основе одного из параметров. Найти конфету в подарке, соответствующую заданному диапазону содержания сахара.

3. Домашние электроприборы. Определить иерархию электроприборов.

Включить некоторые в розетку. Подсчитать потребляемую мощность. Провести сортировку приборов в квартире на основе мощности. Найти прибор в квартире, соответствующий заданному диапазону параметров.

4. Шеф-повар. Определить иерархию овощей. Сделать салат. Подсчитать калорийность. Провести сортировку овощей для салата на основе одного из параметров. Найти овощи в салате, соответствующие заданному диапазону калорийности.

5. Звукозапись. Определить иерархию музыкальных композиций. Записать на диск сборку. Подсчитать продолжительность. Провести перестановку композиций диска на основе принадлежности к стилю. Найти композицию, соответствующую заданному диапазону длины треков.

6. Камни. Определить иерархию драгоценных и полудрагоценных камней. Отобрать камни для ожерелья. Подсчитать общий вес (в каратах) и стоимость. Провести сортировку камней ожерелья на основе ценности. Найти камни в ожерелье, соответствующие заданному диапазону параметров прозрачности.

7. Мотоциклист. Определить иерархию амуниции. Экипировать мотоциклиста. Подсчитать стоимость. Провести сортировку амуниции на основе веса. Найти элементы амуниции, соответствующие заданному диапазону параметров цены.

8. Транспорт. Определить иерархию подвижного состава железнодорожного транспорта. Создать пассажирский поезд. Подсчитать общую численность пассажиров и багажа. Провести сортировку вагонов поезда на основе уровня комфортности. Найти в поезде вагоны, соответствующие заданному диапазону параметров числа пассажиров.

9. Авиакомпания. Определить иерархию самолетов. Создать авиакомпанию.

Посчитать общую вместимость и грузоподъемность. Провести сортировку самолетов компании по дальности полета. Найти самолет в компании, соответствующий заданному диапазону параметров потребления горючего.

10. Таксопарк. Определить иерархию легковых автомобилей. Создать таксопарк. Подсчитать стоимость автопарка. Провести сортировку автомобилей парка по расходу топлива. Найти автомобиль в компании, соответствующий заданному диапазону параметров скорости.

11. Страхование. Определить иерархию страховых обязательств. Собрать из обязательств дериватив. Подсчитать стоимость. Провести сортировку обязательств в деривативе на основе уменьшения степени риска. Найти обязательство в деривативе, соответствующее заданному диапазону параметров.

12. Мобильная связь. Определить иерархию тарифов мобильной компании.

Создать список тарифов компании. Подсчитать общую численность клиентов. Провести сортировку тарифов на основе размера абонентской платы. Найти тариф в компании, соответствующий заданному диапазону параметров.

13. Фургон кофе. Загрузить фургон определенного объема грузом на определенную сумму из различных сортов кофе, находящихся, к тому же, в разных физических состояниях (зерно, молотый, растворимый в банках и пакетиках). Учитывать объем кофе вместе с упаковкой. Провести сортировку товаров на основе соотношения цены и веса. Найти в фургоне товар, соответствующий заданному диапазону параметров качества.

14. Игровая комната. Подготовить игровую комнату для детей разных возрастных групп. Игрушек должно быть фиксированное количество в пределах выделенной суммы денег. Должны встречаться игрушки родственных групп: маленькие, средние и большие машины, куклы, мячи, кубики. Провести сортировку игрушек в комнате по одному из параметров. Найти игрушки в комнате, соответствующие заданному диапазону параметров.

15. Налоги. Определить множество и сумму налоговых выплат физического лица за год с учетом доходов с основного и дополнительного мест работы, авторских вознаграждений, продажи имущества, получения в подарок денежных сумм и имущества, переводов из-за границы, льгот на детей и материальной помощи. Провести сортировку налогов по сумме.

16. Счета. Клиент может иметь несколько счетов в банке. Учитывать возможность блокировки/разблокировки счета. Реализовать поиск и сортировку счетов. Вычисление общей суммы по счетам. Вычисление суммы по всем счетам, имеющим положительный и отрицательный балансы отдельно.

17. Туристические путевки. Сформировать набор предложений клиенту по выбору туристической путевки различного типа (отдых, экскурсии, лечение, шопинг, круиз и т. д.) для оптимального выбора. Учитывать возможность выбора транспорта, питания и числа дней. Реализовать выбор и сортировку путевок.

18. Кредиты. Сформировать набор предложений клиенту по целевым кредитам различных банков для оптимального выбора. Учитывать возможность досрочного погашения кредита и/или увеличения кредитной линии. Реализовать выбор и поиск кредита.

19. 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.

20. Система Платежи. Клиент имеет Счет в банке и Кредитную Карту (КК). Клиент может оплатить Заказ, сделать платеж на другой Счет, заблокировать КК и аннулировать Счет. Администратор может заблокировать КК за превышение кредита.

21. Система Больница. Пациенту назначается лечащий Врач. Врач может сделать назначение Пациенту (процедуры, лекарства, операции). Медсестра или другой Врач выполняют назначение. Пациент может быть выписан из Больницы по окончании лечения, при нарушении режима или при иных обстоятельствах.

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

23. 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».

24. Система Конструкторское бюро. Заказчик представляет Техническое Задание (ТЗ) на проектирование многоэтажного Дома. Конструктор регистрирует ТЗ, определяет стоимость проектирования и строительства, выставляет Заказчику Счет за проектирование и создает Бригаду Конструкторов для выполнения Проекта.

25. 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.

26. Система Автобаза. Диспетчер распределяет заявки на Рейсы между Водителями и назначает для этого Автомобиль. Водитель может сделать заявку на ремонт. Диспетчер может отстранить Водителя от работы. Водитель делает отметку о выполнении Рейса и состоянии Автомобиля.

27. Система Интернет-магазин. Администратор добавляет информацию о Товаре. Клиент делает и оплачивает Заказ на Товары. Администратор регистрирует Продажу и может занести неплательщиков в «черный список».

28. Система Железнодорожная касса. Пассажир делает Заявку на станцию назначения, время и дату поездки. Система регистрирует Заявку и осуществляет поиск подходящего Поезда. Пассажир делает выбор Поезда и получает Счет на оплату. Администратор вводит номера Поездов, промежуточные и конечные станции, цены.

29. Система Городской транспорт. На Маршрут назначаются Автобус, Троллейбус или Трамвай. Транспортные средства должны двигаться с определенным для каждого Маршрута интервалом. При поломке на Маршрут должен выходить резервный транспорт или увеличиваться интервал движения.

30. Система Аэрофлот. Администратор формирует летную Бригаду (пилоты, штурман, радист, стюардессы) на Рейс. Каждый Рейс выполняется Самолетом с определенной вместимостью и дальностью полета. Рейс может быть отменен из-за погодных условий в Аэропорту отлета или назначения. Аэропорт назначения может быть изменен в полете из-за технических неисправностей, о которых сообщил командир.

31. Система Периодические издания. Читатель может сделать Заявку, предварительно выбрав периодические Издания из списка. Система подсчитывает сумму для оплаты. Читатель оплачивает заявку. Администратор добавляет Заявку в «черный список», если Клиент не оплачивает её в определённый срок.

32. Система Заказ гостиницы. Клиент оставляет Заявку на Номер, указав количество мест в номере, класс апартаментов и время пребывания. Администратор рассматривает Заявку, подтверждает или отклоняет её. Результат просматривает Клиент. В случае подтверждения Заявки Клиент оплачивает услуги.

33. Система Жилищно-коммунальные услуги. Квартиросъемщик отправляет Заявку, в которой указывает род работ, масштаб и желаемое время выполнения. Диспетчер формирует соответствующую Бригаду и регистрирует её в Плане работ. Диспетчер может отклонить Заявку в случае занятости всех Бригад.

34. Система Прокат автомобилей. Клиент выбирает Автомобиль из списка доступных, заполняет форму Заказа, указывая паспортные данные, срок аренды. Администратор может отклонить Заявку, указав причины отказа. При подтверждении Заявки Клиент оплачивает Заказ. Система выписывает сумму. В случае повреждения Автомобиля Клиентом Администратор вносит соответствующие пометки.

Вопросы

1. Что такое ООП?

2. Что такое объект?

3. Что такое класс?

4. Назовите основные принципы объектно-ориентированного программирования.

5. Что такое абстракция?

6. Что такое инкапсуляция?

7. Что такое наследование?

8. Что такое полиморфизм?

9. Что такое поле/атрибут/переменная-член класса?

10. Как правильно организовать доступ к полям класса?

11. Что такое модификаторы уровня доступа?

12. Что такое конструктор?

13. Чем различаются конструктор по умолчанию, конструктор без параметров и конструктор с параметрами?

14. Что означает ключевые слова this и super и как их можно использовать?

15. Что такое геттеры? Что такое сеттеры?