Объектно-ориентированное программирование на Java

Практическая работа №4 (part 2)

Реализация отношений между объектами и классами. (8 неделя)

Задание 1: Реализация системы управления библиотекой

Описание

Создайте классы, представляющие библиотеку, книгу и читателя. Реализуйте отношения между этими классами, используя композицию и ассоциацию.

Задание для студентов

1. Реализуйте классы согласно описанию.
2. Добавьте возможность читателю возвращать книги в библиотеку и обновляйте статусы книг соответственно.
3. Создайте несколько читателей и книг, протестируйте созданные классы в главном методе.

Цели задания

- Понять и реализовать отношения между классами с помощью композиции и ассоциации.
- Практиковаться в работе со списками и методами классов.
- Научиться обрабатывать статусы объектов и взаимодействовать с ними.

Классы и их структура:

1. **Класс `Book` (Книга)**:
    - Поля:

    - `title` (название книги)
    - `author` (автор книги)
    - `isCheckedOut` (статус книги: доступна или на руках у читателя)
  - Методы:
    - Конструктор для инициализации полей.
    - Метод `checkout`, который изменяет статус книги на "на руках".
    - Метод `checkin`, который изменяет статус книги на "доступна".
    - Метод `displayInfo`, который выводит информацию о книге.
2. **Класс `Reader` (Читатель)**:
  - Поля:
    - `name` (имя читателя)
    - `checkedOutBooks` (список книг, которые находятся на руках у читателя)
  - Методы:
    - Конструктор для инициализации полей.
    - Метод `checkoutBook`, который добавляет книгу в список и вызывает метод `checkout` у книги.
    - Метод `checkinBook`, который удаляет книгу из списка и вызывает метод `checkin` у книги.
    - Метод `displayInfo`, который выводит информацию о читателе и списке книг.
3. **Класс `Library` (Библиотека)**:
  - Поля:
    - `books` (список книг в библиотеке)
  - Методы:
    - Конструктор для инициализации списка.
    - Метод `addBook`, который добавляет книгу в библиотеку.

- Метод `displayAvailableBooks`, который выводит информацию о всех доступных книгах.

Пример главного класса:

```
public class Main {
    public static void main(String[] args) {
        Library library = new Library();
        Book book1 = new Book("1984", "Джордж Оруэлл");
        Book book2 = new Book("Гарри Поттер", "Джоан Роулин
г");
        library.addBook(book1);
        library.addBook(book2);

        Reader reader = new Reader("Жанибек");

        library.displayAvailableBooks();

        reader.checkoutBook(book1);

        reader.displayInfo();

        library.displayAvailableBooks();
    }
}
```

Пример вывода программы:

```
Доступные книги:
Название: 1984
Автор: Джордж Оруэлл
Доступна: Да

Название: Гарри Поттер
```

```
Автор: Джоан Роулинг
Доступна: Да

Читатель: Жанибек
Книги на руках:
Название: 1984
Автор: Джордж Оруэлл
Доступна: Нет

Доступные книги:
Название: Гарри Поттер
Автор: Джоан Роулинг
Доступна: Да
```

Задание 2: Реализация системы учета автомобилей

Описание

Создайте классы, представляющие автомобиль, водителя и автопарк. Реализуйте отношения между этими классами.

Задание для студентов

1. Реализуйте классы согласно описанию.
2. Добавьте возможность водителю возвращать арендованный автомобиль и обновлять его статус.
3. Создайте несколько водителей и автомобилей, протестируйте созданные классы в главном методе.
4. Обсудите, как можно улучшить систему учета автомобилей, например, добавив функциональность для поиска автомобилей по модели или номерному знаку.

Цели задания

- Понять и реализовать отношения между классами с помощью композиции и ассоциации.

- Практиковаться в использовании списков и методов классов.
- Научиться обрабатывать статусы объектов и взаимодействовать с ними.

Классы и их структура:

1. **Класс `Car` (Автомобиль)**:
   - Поля:
     - `licensePlate` (номерной знак)
     - `model` (модель автомобиля)
     - `isRented` (статус аренды)
   - Методы:
     - Конструктор для инициализации полей.
     - Метод `rent`, который устанавливает статус аренды.
     - Метод `returnCar`, который сбрасывает статус аренды.
     - Метод `displayInfo`, который выводит информацию об автомобиле.
2. **Класс `Driver` (Водитель)**:
   - Поля:
     - `name` (имя водителя)
     - `rentedCar` (арендованный автомобиль)
   - Методы:
     - Конструктор для инициализации полей.
     - Метод `rentCar`, который устанавливает арендованный автомобиль и вызывает метод `rent`.
     - Метод `returnCar`, который сбрасывает статус аренды автомобиля и освобождает его.
     - Метод `displayInfo`, который выводит информацию о водителе и арендованном автомобиле.
3. **Класс `CarPark` (Автопарк)**:

- Поля:
  - `cars` (список автомобилей)
- Методы:
  - Конструктор для инициализации списка.
  - Метод `addCar`, который добавляет автомобиль в автопарк.
  - Метод `displayAvailableCars`, который выводит информацию о всех доступных автомобилях.

Пример главного класса

```
public class Main {
    public static void main(String[] args) {
        CarPark carPark = new CarPark();
        Car car1 = new Car("747ABC", "Toyota Camry");
        Car car2 = new Car("668AID", "Honda Accord");
        carPark.addCar(car1);
        carPark.addCar(car2);

        Driver driver = new Driver("Махмет");

        carPark.displayAvailableCars();

        driver.rentCar(car1);

        driver.displayInfo();

        carPark.displayAvailableCars();
    }
}
```

Пример вывода программы:

```
Доступные автомобили:
Модель: Toyota Camry
Номерной знак: 747ABC
Доступен: Да

Модель: Honda Accord
Номерной знак: 668AID
Доступен: Да

Водитель: Махмет
Арендованный автомобиль:
Модель: Toyota Camry
Номерной знак: 747ABC
Доступен: Нет

Доступные автомобили:
Модель: Honda Accord
Номерной знак: 668AID
Доступен: Да
```

Ресурсы по объектно-ориентированному программированию

- Object-Oriented Programming Concepts
- Java OOP Basics