Лабораторная работа №1

Создание базовых классов

Предмет: Объектно-ориентированное программирование

Группа: 203

Вариант: [Выбирается по списку группы]

Цель работы

Изучить основы создания классов в Java, освоить работу с конструкторами, методами и модификаторами доступа.

Теоретические сведения

Класс в Java

Класс — это шаблон для создания объектов, который определяет их структуру и поведение.

Структура класса:

```
[модификаторы] class ИмяКласса {
    // Поля (переменные экземпляра)
    [модификаторы] тип имяПоля;

    // Конструкторы
    [модификаторы] ИмяКласса(параметры) { }

    // Методы
    [модификаторы] возвращаемыйТип имяМетода(параметры) { }
}
```

Задание

Вариант 1: Система юнитов

Создать классы для представления различных типов юнитов в игре.

Вариант 2: Система ресурсов

Создать классы для управления игровыми ресурсами (золото, дерево, камень).

Вариант 3: Система зданий

Создать классы для различных типов зданий в игре.

Требования к реализации

Обязательные классы:

1. Класс `Position`

```
public class Position {
    private int x;
    private int y;

    // Конструкторы
    // Геттеры и сеттеры
    // Методы для работы с координатами
}
```

2. Базовый класс `Unit`

```
public class Unit {
    private String name;
    private int health;
```

Детальные требования

Класс `Position`:

- Конструктор по умолчанию (0, 0)
- Параметризованный конструктор (x, y)
- Геттеры и сеттеры для x и y
- Метод `getDistanceTo(Position other)` — вычисление расстояния
- Метод `equals(Object obj)` — сравнение позиций
- Метод `toString()` — строковое представление

Класс `Unit`:

- Конструктор по умолчанию
- Параметризованный конструктор
- Конструктор копирования

Дополнительные задания

Для получения "хорошо":

- Добавить валидацию входных данных в конструкторах
- Реализовать метод `clone()` для класса `Unit`
- Добавить логирование действий юнитов

Для получения "отлично":

- Создать интерфейс `Movable` с методом `moveTo(Position position)`
- Создать интерфейс `Combatable` с методами `attack(Unit target)` и `takeDamage(int damage)`
- Реализовать паттерн "Фабрика" для создания юнитов
- Добавить систему опыта и уровней для юнитов

Пример использования

```
public class Main {
    public static void main(String[] args) {
        // Создание игрового поля
        GameBoard board = new GameBoard(10, 10);

        // Создание юнитов
        Unit warrior = new Unit("Warrior", 150, 25, 15);
        Unit archer = new Unit("Archer", 100, 30, 10);

        // Размещение на поле
        Position pos1 = new Position(1, 1);
        Position pos2 = new Position(2, 2);

        board.placeUnit(warrior, pos1);
        board.placeUnit(archer, pos2);

        // Игровая логика
        if (warrior.canAttack(archer)) {
            warrior.attack(archer);
        }

        // Движение
        Position newPos = new Position(3, 3);
        warrior.moveTo(newPos);
        board.placeUnit(warrior, newPos);
        board.removeUnit(pos1);
    }
}
```

Критерии оценки

- **3 балла** — создание всех обязательных классов с базовой функциональностью
- **4 балла** — корректная реализация всех методов и конструкторов
- **5 баллов** — выполнение дополнительных заданий для "хорошо"
- **6 баллов** — выполнение дополнительных заданий для "отлично"

Вопросы для самопроверки

1. В чем разница между `public`, `private` и `protected`?
2. Зачем нужен конструктор по умолчанию?
3. Когда используется ключевое слово `this`?
4. Как правильно реализовать метод `equals()`?
5. Что такое инкапсуляция и как она реализуется в Java?

Сроки выполнения

- **Начало:** [Дата занятия]
- **Сдача:** [Дата следующего занятия]
- **Защита:** [Дата защиты]

Контакты преподавателя

- **Email:** [ваш.email@university.edu]
- **Telegram:** [@username]