Классы и объекты в Java

Лекция 2: Основы ООП

Преподаватель: [Ваше имя]

Группа: 203

Семестр: Осенний 2024

План лекции

1. **Синтаксис Java**
2. **Структура класса**
3. **Конструкторы**
4. **Модификаторы доступа**
5. **Практический пример: класс Unit**

Синтаксис Java

Основные правила:

- **Регистрозависимость** — `Unit` ≠ `unit`
- **Точка с запятой** обязательна в конце выражений
- **Фигурные скобки** для блоков кода
- **Имена классов** начинаются с заглавной буквы
- **Имена методов** начинаются с маленькой буквы

Структура класса

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

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

    // Методы
    [модификаторы] возвращаемыйТип имяМетода(параметры) {
        // тело метода
        return значение;
    }
}
```

Пример простого класса

```
public class Unit {
    // Поля класса
    private String name;
    private int health;
    private int attack;

    // Конструктор
    public Unit(String name, int health, int attack) {
        this.name = name;
        this.health = health;
        this.attack = attack;
    }

    // Методы
    public void takeDamage(int damage) {
        this.health -= damage;
        if (this.health < 0) {
            this.health = 0;
        }
    }

    public boolean isAlive() {
        return this.health > 0;
    }
}
```

Конструкторы

Назначение:

- **Инициализация объекта** при создании
- **Установка начальных значений** полей
- **Выполнение подготовительных операций**

Особенности:

- Имя конструктора = имя класса
- Нет возвращаемого значения
- Может быть несколько конструкторов (перегрузка)
- Автоматически вызывается при `new`

Типы конструкторов

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

    // Конструктор по умолчанию
    public Unit() {
        this.name = "Unknown";
        this.health = 100;
        this.attack = 10;
    }

    // Параметризованный конструктор
    public Unit(String name, int health, int attack) {
        this.name = name;
        this.health = health;
        this.attack = attack;
    }

    // Конструктор копирования
    public Unit(Unit other) {
        this.name = other.name;
        this.health = other.health;
        this.attack = other.attack;
    }
}
```

Модификаторы доступа

`public` — доступ везде

```
public String name;          // Поле доступно из любого места
public void attack() { }     // Метод доступен из любого места
```

`private` — доступ только внутри класса

```
private int health;          // Поле доступно только внутри класса
private void heal() { }      // Метод доступен только внутри класса
```

`protected` — доступ в пакете и наследниках

```
protected int experience;  // Доступ в пакете и наследниках
```

Ключевое слово `this`

Использование:

- **Обращение к полям** текущего объекта
- **Вызов конструкторов** текущего класса
- **Передача ссылки** на текущий объект

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

    public Unit(String name, int health) {
        this.name = name;       // this.name – поле класса
        this.health = health;  // name – параметр метода
    }

    public Unit copy() {
```

Создание объектов

```
public class Main {
    public static void main(String[] args) {
        // Создание объекта с помощью new
        Unit warrior = new Unit("Warrior", 150, 25);
        Unit archer = new Unit("Archer", 100, 30);

        // Вызов методов
        warrior.takeDamage(50);
        System.out.println("Warrior alive: " + warrior.isAlive());

        // Создание копии
        Unit warriorCopy = new Unit(warrior);
    }
}
```

Практический пример: Иерархия юнитов

```
// Базовый класс для всех юнитов
public abstract class Unit {
    protected String name;
    protected int health;
    protected int maxHealth;
    protected int attack;
    protected int defense;
    protected Position position;

    public abstract void performAction();
    public abstract String getUnitType();
}

// Конкретные типы юнитов
public class Warrior extends Unit { }
public class Archer extends Unit { }
public class Mage extends Unit { }
```

Домашнее задание

Задача 1:

Создать класс `Position` с полями `x` и `y` (координаты на игровом поле)

Задача 2:

Расширить класс `Unit` методами:

- `moveTo(Position newPosition)`
- `getDistanceTo(Unit other)`
- `canAttack(Unit target)`

Задача 3:

Создать класс `GameBoard` для представления игрового поля

Что дальше?

На следующей лекции:

- **Наследование и полиморфизм**
- **Абстрактные классы**
- **Переопределение методов**

Подготовка:

- Изучить главу 3-4 из учебника
- Выполнить домашнее задание
- Подготовить вопросы по текущей теме

Вопросы?

Контакты:

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

Следующая лекция: Наследование и полиморфизм