Урок 4

Наследование

Продолжаем знакомиться с основополагающими принципами ООП.

Основополагающие принципы ООП

Продолжаем знакомиться с основными принципами ООП: рассмотрим наследование.

Используя **наследование**, можно создать класс, который определяет общие характеристики для семейства, расширяющегося классами-наследниками. Это более специализированные классы, каждый из которых обладает особыми характеристиками, наращивая сущность класса-родителя. Подкласс наследует все члены, определенные в родителе, добавляя к ним собственные. Для реализации наследования используется ключевое слово extends в следующей форме:

```
class имя_подкласса extends имя_суперкласса
```

Можно говорить о суперклассе «Животные» (**Animal**), от которого будут наследоваться более специализированные классы: «Звери» (**Beast**) и «Птицы» (**Birds**). Они будут родителями для классов «Кот» (**Cat**), «Пес» (**Dog**) и «Утка» (**Duck**).

```
public class Animal {
    protected String name;
    public Animal() {
    }
    public Animal(String name) {
        this.name = name;
    }
    public void animalInfo() {
        System.out.println("Animal: " + name);
    }
}

public class Beast extends Animal {
    protected int pawsNumber;

    public Beast(String name, int pawsNumber) {
        this.name = name;
        this.pawsNumber = pawsNumber;
    }
    public void beastInfo() {
        System.out.println("Beast: " + name + " with " + pawsNumber + " paws");
    }
}

public class Cat extends Beast {
    protected String color;

    public Cat(String name, int pawsNumber, String color) {
        this.name = name;
        this.pawsNumber = pawsNumber;
        this.color = color;
    }
    public void catInfo() {
        System.out.println("Cat: " + name + " Color " + color);
    }
}

public class MainClass {
    public static void main(String[] args) {
        Animal animal = new Animal("Animal");
```

```
        Cat cat = new Cat("Barsik", "White");
        animal.animalInfo();
        cat.animalInfo();
        cat.catInfo();
    }
}
Результат:
Animal: animal
Animal: Barsik
Cat: Barsik White
```

Подкласс **Cat** включает все члены суперкласса **Animal**. Поэтому объект **cat** имеет доступ к методу **animalInfo()**, и в методе **catInfo()** возможна непосредственная ссылка на переменную **name**, как если бы она была частью класса **Cat**.

Несмотря на то, что класс **Animal** является суперклассом для **Cat**, он остается независимым и самостоятельным. Более того, один подкласс может быть суперклассом для другого. Для каждого создаваемого подкласса можно указать только один суперкласс: в Java не поддерживается множественное наследование.

Ключевое слово super

Java позволяет не только наследовать методы родительских классов, но и расширять их функциональность через перегрузку в дочерних. Чтобы в перегруженном дочернем методе сначала вызвать родительский функционал, а затем расширить его, применяют ключевое слово **super**.

Класс **Container** рассчитывает объем грузового контейнера в порту:

```
class Container{
     protected int width;
     protected int height;
     protected int depth;

     public Container(int w, int h, int d){
          width = w;
          height = h;
          depth = d;
     }
}
```

Появляется класс для работы с тяжелыми контейнерами, и в нем — параметр **weight** (вес). Для родительского класса конструктор уже описан. Так зачем дублировать код?

```
class SuperContainer extends Container{
     protected int weight

     public SuperContainer(int w, int h, int d, int m){
          super(w, h, d);
          weight = m;
     }
}
```

Вызов метода **super()** всегда должен быть первым оператором, выполняемым внутри конструктора подкласса.

При вызове **super()** с нужными аргументами мы фактически обращаемся к конструктору **Container**, который инициализирует переменные **width**, **height** и **depth**, используя переданные ему значения соответствующих параметров. Остается инициализировать только добавленное значение **weight**.

Обратите внимание: если проставить у полей класса **Container** модификатор **private**, код продолжит обращаться к ним без проблем.

Второй способ применения ключевого слова **super** напоминает действие слова **this**, только при этом мы всегда ссылаемся на суперкласс подкласса, в котором используется вызываемая сущность. Общая форма:

```
super.метод();
super.свойство;
```

Имена свойств или методов могут совпадать.

```
class A {
    int i;
}

                        // Наследуемся от класса A
class B extends A {
    int i;              // Совпадающее имя скрывает переменную i в классе A

    B(int a, int b) {
        super.i = a;    // Обращаемся к переменной i из класса A
        i = b;          // Обращаемся к переменной i из класса B
    }

    void show() {
        System.out.println("i из суперкласса: " + super.i);
        System.out.println("i в подклассе: " + i);
    }
}
```

Будут по цепочке вызваны методы родительских классов, начиная с самого первого.

Ключевое слово final в сочетании с наследованием

Существует несколько назначений ключевого слова **final**:

Первый: создать именованную константу.

```
final int MONTHS_COUNT = 12; // final в объявлении поля или переменной
```

Второй: предотвратить переопределение методов.

```
public final void run() { // final в объявлении метода
}
```

Третий: запретить наследование от текущего класса.

```
public final class A {           // final в объявлении класса
}
public class B extends A { // Ошибка: у класса A не может быть подклассов
}
```