Основы программирования - Java

ФИСТ 1 курс

Власенко Олег Федосович

Лекция 5

ООП.

Строки

Неформальное введение в ООП

Что такое ООП?

- Концепция *объектно–ориентированного программирования* (ООП) предлагает оперировать в программе не переменными и функциями, а *объектами*.
- Всё в программе является объектами.
- У объекта имеются свойства и методы.
- Свойства представляют собой переменные, принадлежащие объекту.
- Методы — функции, позволяющие получить / изменить информацию об объекте.

Объект Кот

Какие свойства есть у кота?

Объект Кот

- Порода
- Цвет
- Рост
- Возраст
- Дата последнего кормления
- Дата последнего поглаживания
- Дата последнего мяукания
- ...

А методы?

Объект Кот

- Мяукнуть
- Поесть
- Потребовать погладить
- Погулять
- ...

А что с другими животными?

Собака

Сравнение свойств Кота и Собаки

- Порода
- Цвет
- Рост
- Возраст
- Дата последнего кормления
- Дата последнего поглаживания
- **Дата последнего мяукания**

- Порода
- Цвет
- Рост
- Возраст
- Дата последнего кормления
- Дата последнего поглаживания
- **Дата последнего гавкания**

Сравнение методов Кота и Собаки

- **Мяукнуть**
- Поесть
- Потребовать погладить
- Погулять

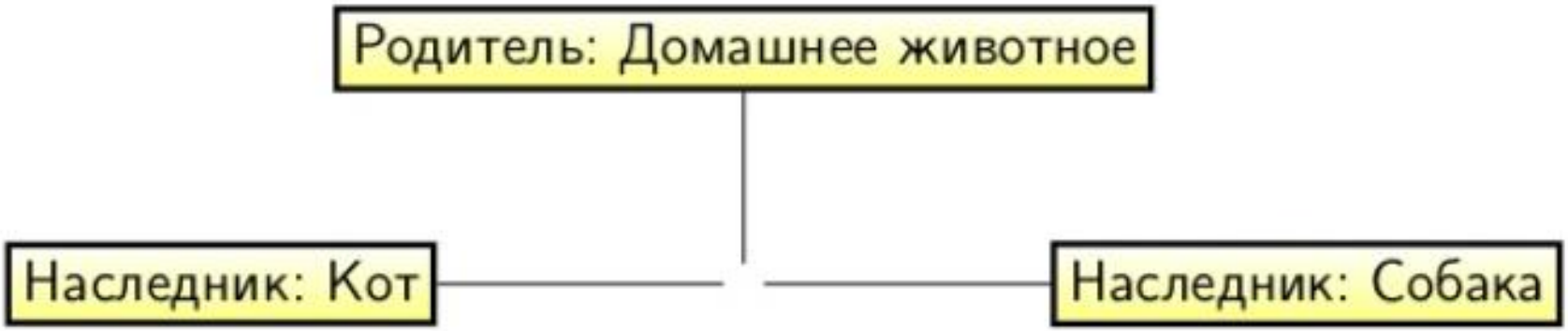
- **Гавкнуть**
- Поесть
- Потребовать погладить
- Погулять
- **Выгуляться**

Домашние животные

Принцип наследования

Общие свойства и методы объектов можно вынести в класс–**родитель**. Все "дети"–наследники автоматически получают их.

Схема наследования

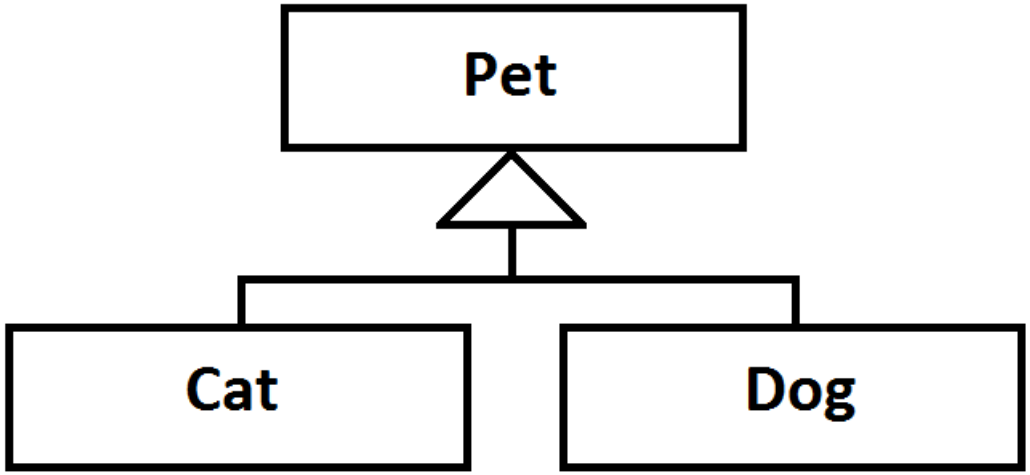


Несколько терминов

- Одинаковые объекты являются **экземплярами класса**.
- Кот — это, на самом деле, класс.
- А вот, например, кот Вася — это объект, то есть, представитель класса.
- Класс — это программная структура.
- В программе мы сначала создаём класс, а потом уже создаём (**инстанцируем**) объекты.

UML – диаграмма классов

UML = Unified Modeling Language



источник вдохновения

- http://www.slideshare.net/smirik/ruby-11754239

Примеры с ООП

Пример 1

Создать базовый класс Vehicle описывающий поведение траспортного средства. Создать класс Auto расширяющий Vehicle методом honk() (гудеть).

Vehicle должен содержать следующие методы:

moveForward(double speed)

moveBack(double speed)

turnLeft(double angle)

turnRight(double angle)

Vehicle должен хранить информацию о направлении движения (0 – на восток, 90 – на север, 180 – на запад, 270 – на юг), реализации методов должны выводить сообщения в консоль «Еду в направлении X со скоростью Y».

В Auto должен быть метод honk() (гудеть) – который в консоль выводит сообщение «Биип!».

Пример 1 - реализация

Класс-клиент:

```
public class Lect4Main1 {
    public static void main(String[] args) {
            Auto auto1 = new Auto(0);
            auto1.honk();
            auto1.moveForward(20);
            auto1.turnLeft(90);
            auto1.turnLeft(180);
            auto1.turnLeft(120);
            auto1.turnRight(30);
            auto1.moveBack(10);
            auto1.honk();
    }
}
```

Пример 1 - вывод

Console:

- Биип!
- Я еду в направлении 0.0 со скоростью 20.0
- Я еду в направлении 90.0 со скоростью 20.0
- Я еду в направлении 270.0 со скоростью 20.0
- Я еду в направлении 30.0 со скоростью 20.0
- Я еду в направлении 0.0 со скоростью 20.0
- Я еду в направлении 0.0 со скоростью -10.0
- Биип!

Пример 1 – класс Vehicle

```
public class Vehicle {
	private double speed;
	private double direction;

	public Vehicle(double initDirection) {
		direction = initDirection;
		speed = 0;
	}

	public void moveForward(double speed) {
		this.speed = speed;
		printStatus();
	}

	private void printStatus() {
		System.out.println("Я еду в направлении " + direction
				+ " со скоростью " + speed);
	}
```

Пример 1 – класс Vehicle (ч 2)

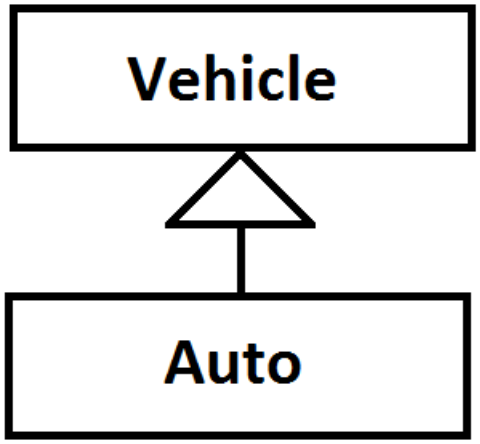
```
    public void moveBack(double speed) {
        this.speed = -speed;
        printStatus();
    }

    public void turnLeft(double angle) {
        direction = direction + angle;
        if (direction > 360) {
            direction = direction - 360;
        }
        printStatus();
    }

    public void turnRight(double angle) {
        direction = direction - angle;
        if (direction < 0) {
            direction = direction + 360;
        }
        printStatus();
    }
}
```

Пример 1 – класс Auto

```
public class Auto extends Vehicle {

        public Auto(int direction) {
                super(direction);
        }

        public void honk() {
                System.out.println("Биип!");
        }

}
```

UML – диаграмма классов

UML = Unified Modeling Language



Пример 2 – перевод числа из 10-й системы счисления в 2-ую

Пример:

N = 12

N % 2 = 12 % 2 = 0 → 0 N = N / 2 = 12 / 2 = 6 N > 0 +

N % 2 = 6 % 2 = 0 → 0 N = N / 2 = 6 / 2 = 3 N > 0 +

N % 2 = 3 % 2 = 1 → 1 N = N / 2 = 3 / 2 = 1 N > 0 +

N % 2 = 1 % 2 = 1 → 1 N = N / 2 = 1 / 2 = 0 N > 0 -

Полученные цифры нужно вывести в обратном порядке:

1100

Пример 2 – реализация через StringBuilder

```
public class Lect4Main2 {
	public static void main(String[] args) {
		int num = 49;
		StringBuilder str = new StringBuilder();
		do {
			int binDigit = num % 2;
			char binChar = binDigit == 0 ? '0' : '1';
			str.append(binChar);
			num = num / 2;
		} while (num > 0);
		str.reverse();
		System.out.println(str.toString());
	}
}
```

Пример 2 – реализация через стек

```
public class Lect4Main2_v2  {
        public static void main(String[] args) {
                int num = 129;
                StackChar stack = new StackChar(32);
                do {
                        int binDigit = num % 2;
                        char binChar = binDigit == 0 ? '0' : '1';
                        stack.push(binChar);
                        num = num / 2;
                } while (num > 0);
                while (!stack.isEmpty()) {
                        System.out.print(stack.pop());
                }
                System.out.println();
        }
}
```

реализация стека

```
public class StackChar {
        private char [] stack;
        private int top;

        public StackChar(int size) {
                stack = new char[size];
                top = 0;
        }

        public void push(char ch) {
                stack[top] = ch;
                top++;
        }

        …
}
```

реализация стека (ч 2)

```
public class StackChar {
        private char [] stack;
        private int top;

        …

        public char pop() {
                top--;
                return stack[top];
        }

        public boolean isEmpty() {
                return top == 0;
        }
}
```

Интерфейс

Интерфейс это совокупность методов и правил взаимодействия элементов системы. Другими словами интерфейс определяет как элементы будут взаимодействовать между собой.

- Интерфейс двери - наличие ручки
- Интерфейс автомобиля - наличие руля, педалей, рычага коробки передач.
- Интерфейс дискового телефона - трубка + дисковый набиратель номера

Когда вы используете эти "объекты" вы уверены в том что вы сможете использовать их подобным образом. Благодаря тому что вы знакомы с их интерфейсом

Интерфейс IStackChar

```
public interface IStackChar {
        void push(char ch);
        char pop();
        boolean isEmpty();
}
```

реализация стека №1

```
public class StackChar1 implements IStackChar {

        private char [] stack;
        private int top;

        public StackChar1(int size) {
                stack = new char[size];
                top = 0;
        }

        @Override
        public void push(char ch) {
                stack[top] = ch;
                top++;
        }
        …
}
```

реализация стека №2

```
import java.util.ArrayList;

public class StackChar2 implements IStackChar {
	ArrayList<Character> stack;

	public StackChar2(int size) {
		stack = new ArrayList<Character>(size);
	}

	@Override
	public void push(char ch) {
		stack.add(ch);
	}

	…

}
```

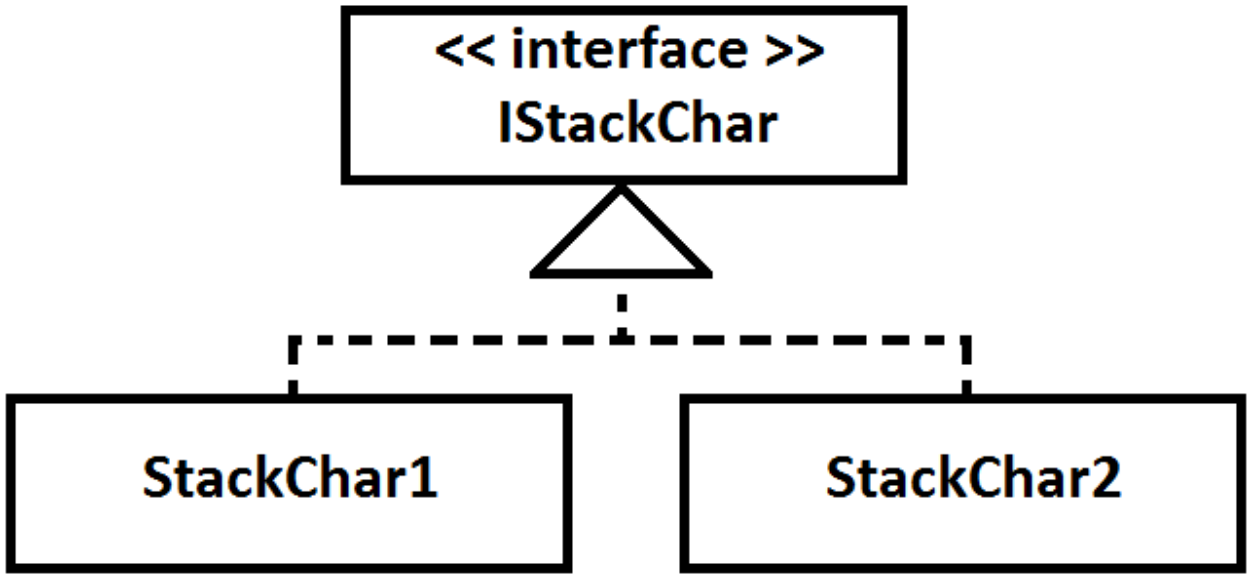
реализация стека №2 (ч 2)

```
public class StackChar2 implements IStackChar {
        ArrayList<Character> stack;
        …
        @Override
        public char pop() {
                int lastIndex = stack.size() - 1;
                char ch = stack.get(lastIndex);
                stack.remove(lastIndex);
                return ch;
        }

        @Override
        public boolean isEmpty() {
                return stack.isEmpty();
        }
}
```

Решение с интерфейсом

```
public class Lect4Main2_v3  {
        public static void main(String[] args) {
                int num = 26;
                //IStackChar stack = new StackChar1(32);
                IStackChar stack = new StackChar2(32);
                do {
                        int binDigit = num % 2;
                        char binChar = binDigit == 0 ? '0' : '1';
                        stack.push(binChar);
                        num = num / 2;
                } while (num > 0);
                while (!stack.isEmpty()) {
                        System.out.print(stack.pop());
                }
                System.out.println();
        }
}
```

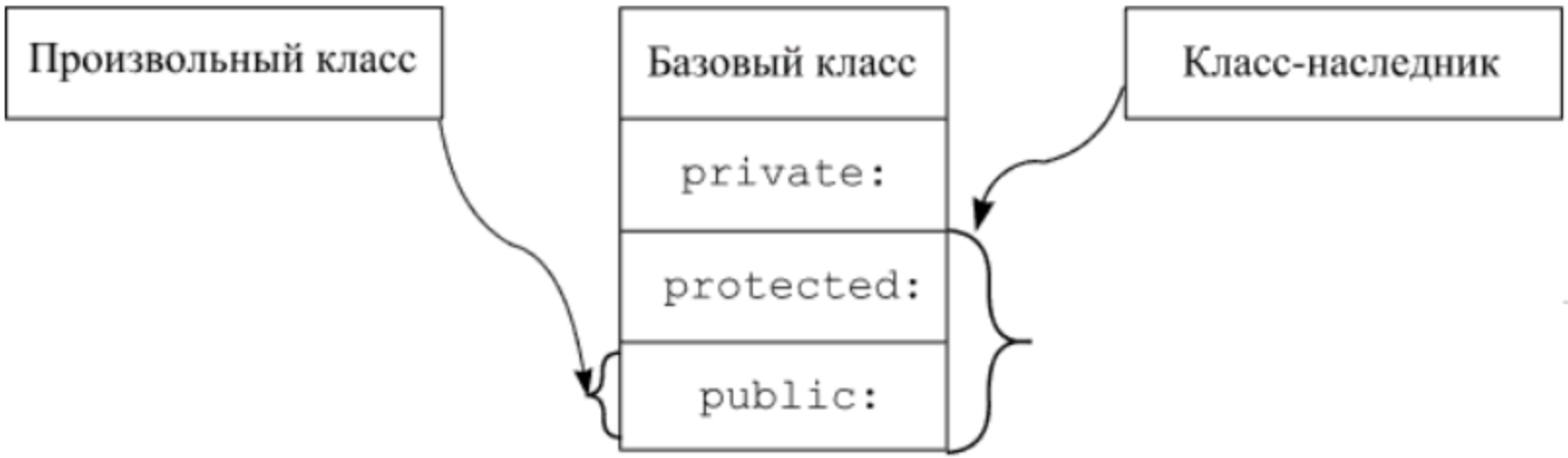
Диаграмма классов (UML)



Детали ООП в Java

Модификаторы доступа

- private: члены класса доступны только внутри класса;
- «default» (package-private) (модификатор, по-умолчанию): члены класса видны внутри пакета;
- protected: члены класса доступны внутри пакета и в наследниках;
- public: члены класс доступны всем;



Модификаторы доступа

- static - ссылка этого поля у любого экземпляра класса будет ссылаться на одно и то же значение
- final – это модификатор, позволяющий объявлять константные поля в классе.

Элементы класса

```
public class Sample {
    private int x; // переменная экземпляра класса
    private int y = 0; // переменная экземпляра класса
    public final int CURRENT_YEAR = 2012; // константа
    protected static int bonus; // переменная класса
    static String version = "Java SE 7"; // переменная класса
    protected Calendar now;
   public int method(int z) {
        return z++;
   }
}
```

Примеры использования статических методов

```
public class StaticSamples  {
        public static void main(String[] args) {
                double phi = Math.PI / 6;
                System.out.printf("sin(%f)=%f\n", phi, Math.sin(phi));
                phi = Math.PI / 4;
                System.out.printf("tan(%f)=%f\n", phi, Math.tan(phi));
                phi = Math.PI / 2;
                System.out.printf("tan(%f)=%f\n", phi, Math.tan(phi));
                System.out.printf("sin(%f)=%f\n", phi, Math.sin(phi));
                System.out.printf("cos(%f)=%f\n", phi, Math.cos(phi));
        }
}
```

Конструкторы

```
public class Quest {
    // конструктор без параметров (по умолчанию)
    public Quest() {
        System.out.println("Вызван конструктор без параметров!!!");
    }
    // конструктор с параметрами
    public Quest(int idc, String txt) {
        super(); /* вызов конструктора супер класса явным образом
                    необязателен,   компилятор вставит его
                    автоматически*/
       System.out.println("Вызван конструктор с параметрами!!!");
       System.out.println(id + " " + text);
   }
}
```

Порядок инициализации класса

```
public class Department {
    { System.out.println("logic"); };                    //2
    static { System.out.println("static logic"); }  //1
    private int id = 7;
    public Department(int d) {
        id = d;
        System.out.println("конструктор");          //3
    }
    int getId() {  return id;   }
   { id = 10; System.out.println("logic"); }          //2
}
```

Абстрактный класс

```
public abstract class AbstractCourse {
    private String name;
    public AbstractCourse() {
    }
    public abstract void changeTeacher(int id);
   /*определение  метода отсутствует */
    public void setName(String n) {
        name = n;
    }
}
```

Особенности наследования в Java

```
public class ArrayList<E> extends
AbstractList<E> implements List<E>,
RandomAccess, Cloneable, Serializable {

   …
}
```

Внимание: В Java класс наследуется от ОДНОГО класса, но реализует произвольное количество интерфейсов.

Литература

- http://www.slideshare.net/smirik/ruby-11754239 (источник вдохновения - наследование)

- http://hashcode.ru/questions/136909/%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D1%8B-%D0%B2-%D0%BE%D0%BE%D0%BF-java-%D0%BF%D0%BE-%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%BC%D1%83 (источник вдохновения - интерфейсы)

Спасибо за внимание!

Власенко Олег Федосович

E-mail: vlasenko.oleg@gmail.com

Vk: vk.com/oleg.f.vlasenko

Телефон: 8 902 246 05 47