Язык программирования Java. Начальный уровень

Мультимедийный курс

автор: Васильев А.Н.

www.vasilev.kiev.ua

Киев 2017

Лекция 6. Знакомство с классами и объектами

- Классы и объекты - основные понятия
- Описание класса
- Поля и методы класса
- Создание объекта класса
- Основные операции с объектами
- Перегрузка методов
- Использование конструктора

- На что жалуемся?
- На голову жалуется.
- Это хорошо. Лёгкие дышат, сердце стучит.
- А голова?
- А голова - предмет тёмный, исследованию не подлежит.

из к/ф "Формула любви"

Принципы ООП

- **Инкапсуляция - код и данные объединены в одно целое (реализация: классы и объекты)**
- **Полиморфизм - использование единого интерфейса (реализация: перегрузка методов)**
- **Наследование - передача свойств и характеристик (реализация: наследование классов)**

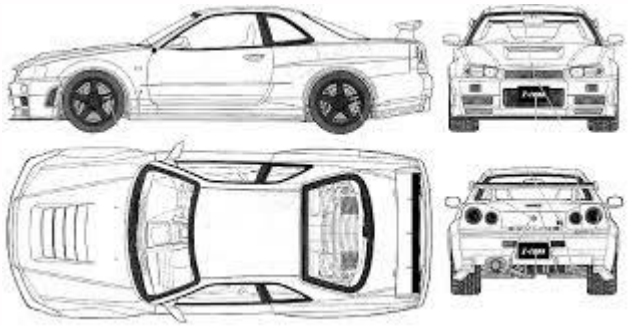
Структурное программирование

ООП

Класс vs. Объект

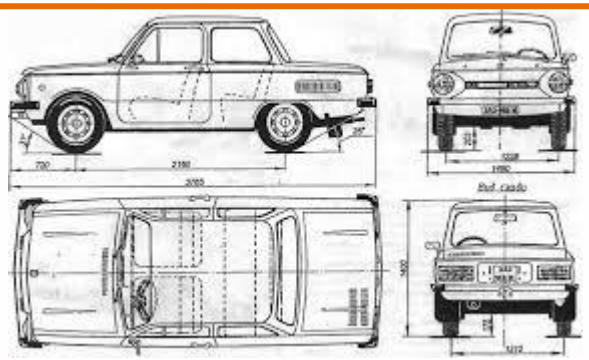
- Класс - некоторая схема или шаблон, на основе которой создаются объекты
- Объект - создается на основе класса и имеет поля и методы

Класс BMW



Объект класса BMW

Класс ZAZ



Объект класса ZAZ

Описание класса

Синтаксис

Ключевое слово `class`

Имя класса

Поля описываются как переменные: указывается тип и название поля

Метод описывается так:
- указывается тип результата;
- имя метода;
- список аргументов;
- команды, выполняемые при вызове метода (в блоке из круглых скобок)

```
class   имя {
    // Поля и методы
}
```

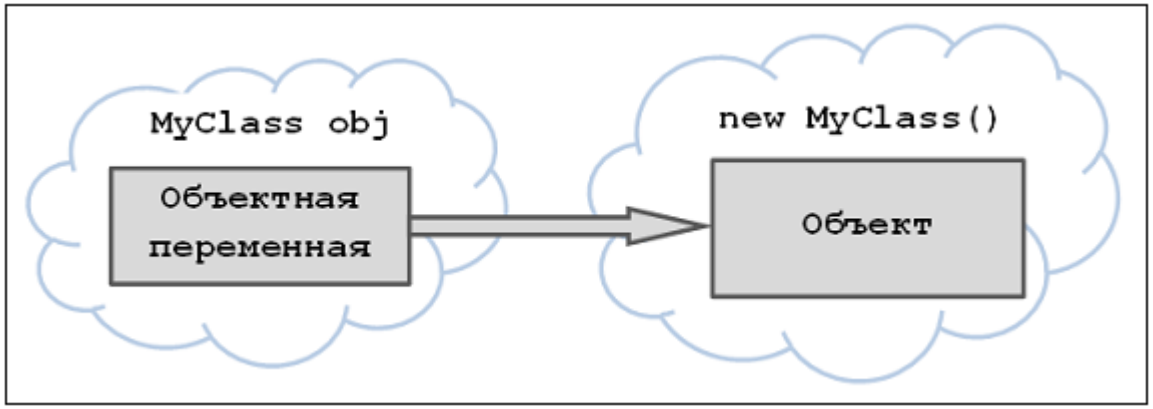
Поля и методы класса (члены класса) описываются в теле класса (блок из фигурных скобок)

Пример:

```
class MyClass{
  // Поле:
  int num;
  // Метод:
  void show(){
    System.out.println(num);
  }
}
```

Создание объекта

Общая схема: объектная переменная ссылается на объект




```
MyClass obj=new MyClass();
```

```
MyClass obj;
obj=new MyClass();
```

- Объявляется объектная переменная
- Создается объект
- Ссылка на объект записывается в объектную переменную

Тип объектной переменной - имя класса
Объект создается с помощью оператора `new`

Поля и методы объекта

- У объекта есть поля и методы - те, что описаны в классе, на основе которого создан объект. У каждого объекта поля и методы "свои"

- Для доступа к полям и методам объекта используется точечный синтаксис: после имени объекта через точку указывается имя поля или инструкция вызова метода

Например:

Описание класса

```
class MyClass{
   // Поле:
   int num;
   // Метод:
   void show(){
      System.out.println(num);
   }
}
```

Создание объекта класса

```
MyClass obj=new MyClass();
```

Обращение к полям и методам объекта

```
obj.num=123;
obj.show();
```

Открытый член - описан с идентификатором доступа `public` или без него
Закрытый член класса - описан с идентификатором доступа `private`

Программа: знакомство с классами и объектами

```
// Описание класса:
class MyClass{
   // Поле:
   int num;
   // Метод:
   void show(){
      System.out.println("Поле num: "+num);
   }
} // Класс с главным методом:
class UsingObjsDemo{
   // Главный метод:
   public static void main(String[] args){
      // Первый объект:
      MyClass A=new MyClass();
      // Объектная переменная:
      MyClass B;
      // Второй объект:
      B=new MyClass();
      // Присваивание значений полям объектов:
      A.num=123;
      B.num=321;
      // Вызов методов:
      A.show();
      B.show();
   }
}
```

Результат программы:

```
Поле num: 123
Поле num: 321
```

Программа: присваивание объектов

```
// Описание класса:
class MyClass{
   // Поле:
   int num;
   // Метод:
   void show(){
      System.out.println("Поле num: "+num);
   }
} // Класс с главным методом:
class CopyObjsDemo{
   // Главный метод:
   public static void main(String[] args){
      // Объектные переменные:
      MyClass A,B;
      // Создание объекта:
      A=new MyClass();
      // Присваивание объектных переменных:
      B=A;
      // Присваивание значений полям и вызов методов:
      A.num=111;
      B.show();
      B.num=222;
      A.show();
   }
}
```

Результат программы:

```
Поле num: 111
Поле num: 222
```

Программа: перегрузка методов

```
class MyClass{
   // Закрытые поля:
   private int num;
   private char symb;
   // Открытые методы:
   void set(int n,char s){ // Первая версия метода
      num=n;
      symb=s;
   }
   void set(int n){        // Вторая версия метода
      num=n;
      symb='B';
   }
   void set(){             // Третья версия метода
      set(0,'A');
   }
   void show(){
      System.out.println("Поля: "+num+" и "+symb);
   }
}
class MethodsDemo{
   public static void main(String[] args){
      MyClass obj=new MyClass(); // Создание объекта
      // Вызов методов:
      obj.set();
      obj.show();
      obj.set(100);
      obj.show();
      obj.set(200,'C');
      obj.show();
   }
}
```

Результат программы:

```
Поля: 0 и A
Поля: 100 и B
Поля: 200 и C
```

Конструктор

Конструктор - метод, который вызывается автоматически при создании объекта

- **Имя конструктора совпадает с именем класса**
- **Конструктор не возвращает результат и для него не указывается идентификатор типа результата**
- **Конструктор может иметь аргументы, которые передаются в конструктор при создании объекта**
- **Конструктор можно перегружать: класс может содержать несколько конструкторов**

Если в классе не описан ни один конструктор, то используется конструктор по умолчанию: он не выполняет никаких дополнительных действий.

Если в классе описана хотя бы одна версия конструктора, то конструктор по умолчанию больше не доступен.

Программа: использование конструктора

```
class MyClass{
   int num;
   char symb;
   void show(){
      System.out.println("Поля: "+num+" и "+symb);
   }
   // Конструкторы:
   MyClass(){
      num=100;
      symb='A';
   }
   MyClass(int n){
      num=n;
      symb='B';
   }
   MyClass(int n,char s){
      num=n;
      symb=s;
   }
}
class ConstructorsDemo{
   public static void main(String[] args){
      // Создание объектов:
      MyClass A=new MyClass();
      MyClass B=new MyClass(200);
      MyClass C=new MyClass(300,'C');
      // Проверка значений полей:
      A.show();
      B.show();
      C.show();
   }
}
```

Результат программы:

```
Поля: 100 и A
Поля: 200 и B
Поля: 300 и C
```

Статические поля и методы

- Статические члены класса описываются с ключевым словом `static`
- Статические члены относятся к класс, существуют вне зависимости от наличия объектов класса
- При обращении к статическому методу или полю вместо имени объекта указывается имя класса
- Статический метод может обращаться только к статическим полям

```
class MyClass{
    // Статическое поле:
    static int code=100;
    // Статический метод:
    static void show(){
        System.out.println("Поле code: "+code);
    }
}
class StaticDemo{
    public static void main(String[] args){
        // Вызов статического метода:
        MyClass.show();
        // Обращение к статическому полю:
        MyClass.code=200;
        // Вызов статического метода:
        MyClass.show();
    }
}
```

Результат программы:

```
Поле code: 100
Поле code: 200
```

Домашнее задание

- Напишите программу с классом, в котором есть символьное поле и метод, который возвращает значением код символа.

- Напишите программу с классом, у которого есть два целочисленных поля и метод, который выводит в консоль значение этих полей.

- Напишите программу с классом, в котором есть два символьных поля и перегруженный метод для присваивания значений полям, который можно вызывать без аргументов, с одним аргументом и с двумя аргументами.

- Напишите программу с классом, в котором есть два числовых поля и конструкторы, которые позволяют создавать объекты класса без передачи аргументов, с передачей одного аргумента и с передачей двух аргументов.