Курс «Программирование на Java» - Классы и Объекты

IT-Academy

Повторяем:

1) Что такое массив ?
2) Как я могу обращаться к элементу массива?
3) Какое свойство массива я могу получить и использовать в своем коде?
4) Как сортировать массив в Java?
5) Как использовать многомерные массивы в Java?

Рассматриваемые вопросы

- Модификаторы доступа (Access Modifiers)
- Концепция ООП
- Идентификаторы
- Понятие класс, объект

Концепция ООП

Объектно-ориентированное программирование - парадигма программирования, в которой главной идеей являются понятия объектов и классов

ООП возникло в результате развития идей процедурного программирования, где данные и функции (методы) их обработки формально не связаны

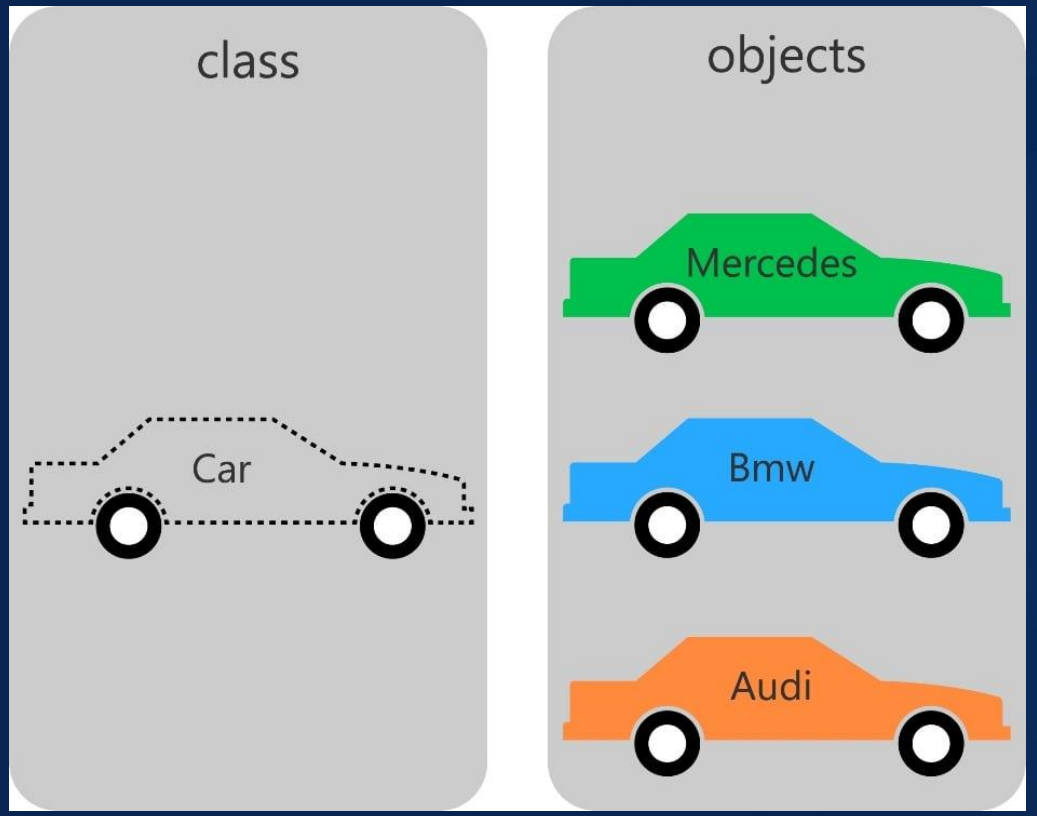
```
public class ModifiersExample {

    public int a;
    protected int b;
    int c;
    private int d;

}
```

Модификаторы доступа

Java предоставляет ряд модификаторов доступа, чтобы задать уровни доступа для классов, переменных, методов и конструкторов.

Существует четыре доступа:

- **public** - видимый для всех
- **protected** - видимый для пакета и всех подклассов
- **default** (без модификатора)- видимый только в пакете
- **private** - видимый только для класса



Класс и Объект

Класс - прототип, чертеж, определяет структуру и поведение создаваемых объектов

Объект - конкретный, реальный экземпляр класса

IT-Academy

Поле и метод класса

Класс в Java имеет два основных элемента:

Поле (field) - имеет идентификатор, тип данных и значение (переменное или постоянное)

Метод (method) - имеет идентификатор, аргументы (входные параметры), возвращаемое значение или void, тело

Хороший тон - иметь приватные поля и методы для доступа к ним.

Поговорим о функциях(методах)

Определение метода в Java начинается с ключевого слова "public" (или других модификаторов доступа, таких как "private" или "protected"), за которым следует возвращаемый тип метода (например, int, double, String и т.д.), затем название метода и в круглых скобках параметры метода (если они есть).

Методы в Java

```
public static int addNumbers(int a, int b) {
    int sum = a + b;
    return sum;
}
```

В этом примере метод "addNumbers" имеет модификатор доступа "public", возвращаемый тип - "int", название метода - "addNumbers", а параметры - "int a" и "int b".

Внутри метода выполняется операция сложения двух чисел и результат возвращается с помощью ключевого слова "return".

IT-Academy

Методы в Java

Методы могут быть вызваны из других частей программы с помощью их названия и передачи необходимых параметров в круглых скобках. Например:

```
public static void main(String[] args) {
    int result = addNumbers( a: 1,  b: 4);
    |
    System.out.println(result);
}
```

В этом примере метод "addNumbers" вызывается с параметрами 5 и 7, и результат (12) сохраняется в переменной "result".

IT-Academy

Методы в Java

Методы в Java могут принимать как параметры, так и не принимать их. Если метод не принимает параметры, то круглые скобки остаются пустыми.
Например:

```
public void printHello() {
    System.out.println("Hello!");
}
```

Метод использует параметры. Вызывающий код передает аргументы.

IT-Academy

Методы в Java

Методы также могут возвращать различные типы данных, включая примитивные типы **(int, double, boolean и т.д.)**, объекты и массивы. Если метод не возвращает никакого значения, то возвращаемый тип указывается как **"void"**. Например

```
public void printHello() {
    System.out.println("Hello!");
}
```

Метод использует параметры. Вызывающий код передает аргументы.

IT-Academy

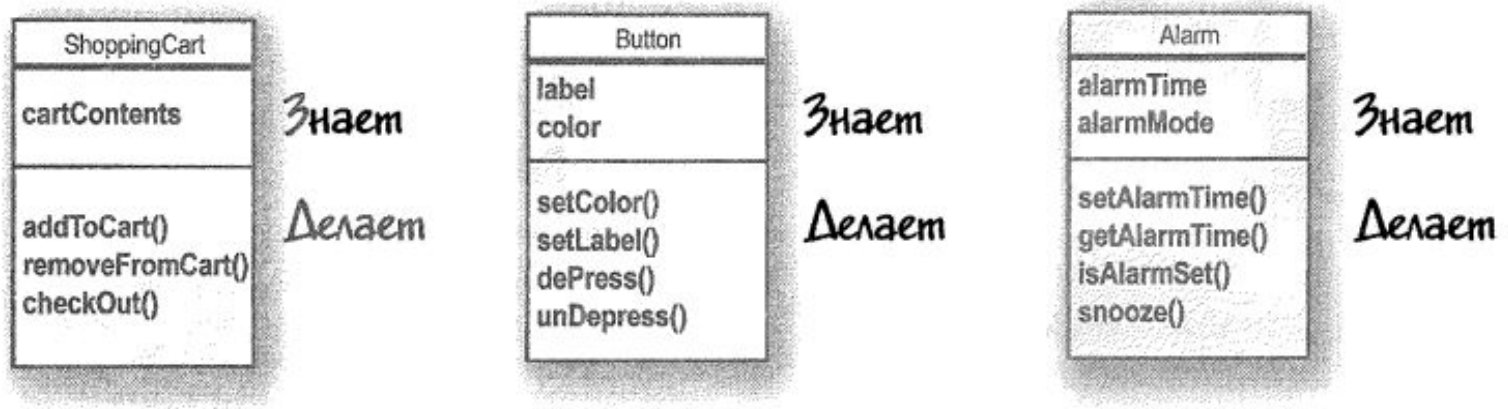
Класс и Объект

!!!По сути любой созданный нами КЛАСС – это новый ссылочный тип данных.

Поле и метод класса

Проектируя класс, думайте об объектах, которые будут созданы на его основе. Думайте:

- о вещах, которые объект **знает;**
- вещах, которые объект **делает.**



То, что объект о себе **знает**, называется

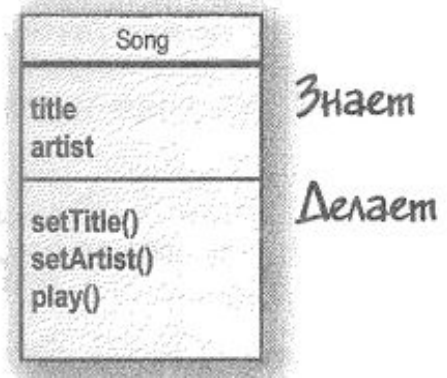
- **переменной экземпляра.**

Переменные экземпляра (состояние)

То, что объект может **делать**, называется

- **методом.**

Методы (поведение)

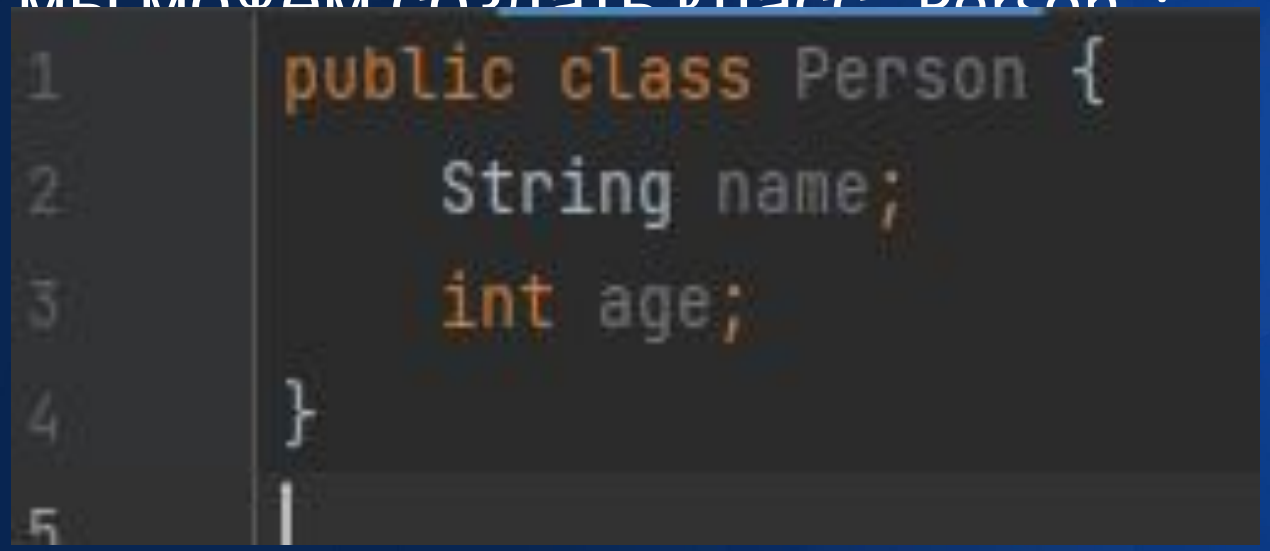


IT-Academy

Для создания класса в Java мы используем ключевое слово "class",

за которым следует имя класса.

Например, мы можем создать класс "Person":



IT-Academy

Класс в Java

```
public class Car {
    // поля класса
    private String manufacturer;
    private String color;
    private Integer engineVolume;

    // конструктор
    public Car(String manufacturer, String color, Integer engineVolume) {
        this.manufacturer = manufacturer;
        this.color = color;
        this.engineVolume = engineVolume;
    }

    // методы
    public void paint() {
        color = "white";
    }

    public void move() { /* ... */ }
}
```

Конструкторы

Конструктор - это специальный метод, который вызывается при создании объекта класса.
Он используется для инициализации полей объекта.
Конструктор имеет тот же имя, что и класс, и не имеет возвращаемого значения

```
no usages
public Computer(String model, String cpu, String gpu, int cores) {
    this.model = model;
    this.cpu = cpu;
    this.gpu = gpu;
    this.cores = cores;
}

1 usage
public Computer() {
}
```

```
1  package com.victoria.app.core.test;
2
   no usages
3  public class Computer {
4
       no usages
5      public Computer(String model, String cpu, String gpu, int cores) {
6          this.model = model;
7          this.cpu = cpu;
8          this.gpu = gpu;
9          this.cores = cores;
10     }
11
       1 usage
12     private String model;
       1 usage
13     private String cpu;
       1 usage
14     private String gpu;
       1 usage
15     private int cores;
16
17 }
18
```

IT-Academy

Конструкторы

Мы можем использовать конструктор с параметрами:

```
Computer computer = new Computer( model: "AMD Ryzen 149353",  cpu: "AMD Ryzen 5 3600",  gpu: "GeForce RTX 3090",  cores: 6);
```

Или можем использовать конструктор без параметров и вручную устанавливать значения полей:

```
Computer computer = new Computer();
computer.setModel("AMD Ryzen 149353");
computer.setCores(6);
computer.setCpu("AMD Ryzen 5 3600");
computer.setGpu("GeForce RTX 3090");
```

IT-Academy

Конструкторы

```
1  public Car(String model, int maxSpeed) {
2     this.model = model;
3     this.maxSpeed = maxSpeed;
4  }
```

А создание объекта теперь выглядит так:

```
1  public static void main(String[] args) {
2     Car bugatti = new Car("Bugatti Veyron", 407);
3  }
```

Обрати внимание, как создается конструктор.

Он похож на обычный метод, но у него нет типа возвращаемого значения. При этом в конструкторе указывается название класса, тоже с большой буквы. В нашем случае — `Car`.

Кроме того, в конструкторе используется новое для тебя ключевое слово `this`.

"this" по-английски — "этот, этого". Это слово указывает на конкретный предмет.

IT-Academy

Конструкторы

Если выразить в одном предложении ответ на вопрос “Зачем нужен конструктор?”, можно сказать: для того, чтобы объекты всегда находились в правильном состоянии.

Когда ты используешь конструкторы, все твои переменные будут корректно проинициализированы, и в программе не будет машин со скоростью 0 и прочих “неправильных” объектов.

Конструкторы

Давайте наглядно проверим где и какой конструктор вызывается.
Создадим класс с тремя конструкторами. В каждом из них проинициализируем поля и
вызовем метод sout.

Например:

```
public Computer(String model, String cpu, String gpu, int cores) {
    this.model = model;
    this.cpu = cpu;
    this.gpu = gpu;
    this.cores = cores;
    System.out.println("это конструктор с 4 полями");
}
```

IT-Academy

Ключевое слово this.

Как правило, применять `this` нужно **в двух случаях:**

1. Когда у переменной экземпляра класса и переменной метода/конструктора одинаковые имена;
2. Когда нужно вызвать конструктор одного типа (например, конструктор по умолчанию или параметризированный) из другого. Это еще называется явным вызовом конструктора.

Вот и все, на самом деле не так много, — всего два случая, когда применяется это страшное ключевое слово. Теперь давайте рассмотрим эти две ситуации на примерах.

IT-Academy

Ключевое слово this.

```
public void setName(String name){
    name = name;
}
```

Это все одно и то же

и получается, что вы просто присваиваете значение переменной `name` из этого метода, ей же. Что конечно не имеет никакого смысла. Поэтому нужен был какой-то способ, чтобы отличить переменную `name` из класса `Human`, от переменной `name` из метода `setName`.Для решения этой проблемы и было введено ключевое слово `this`, которое в данном случае укажет, что нужно вызывать переменную не метода, а класса `Human`:

```
class Human{
    String name;
    int age;

    public void setName(String name){
        this.name = name;
    }
}
```

IT-Academy

Ключевое слово this.

Пример второй — Применение this для явного вызова конструктора

Вызов одного конструктора из другого может пригодиться тогда, когда у вас (как ни странно) несколько конструкторов и вам не хочется в новом конструкторе переписывать код инициализации, приведенный в конструкторе ранее. Запутал? Все не так страшно как кажется. Посмотрите на код ниже, в нем два конструктора класса `Human`:

```
1  class Human{
2      int age;
3      int weight;
4      int height;
5
6      Human(int age, int weight){
7          this.age = age;
8          this.weight = weight;
9      }
10     Human(int age, int weight, int height){
11         //вы вызываете конструктор с двумя параметрами
12         this(age, weight);
13         //и добавляете недостающую переменную
14         this.height = height;
15     }
16 }
```

```
// идентификатор пакета
package com.academy;

public class Phone {
    //поле
    private int price;

    // конструктор
    public Phone(int price) {
        this.price = price;
    }

    // метод, возвращающий значение
    public int getPrice() {
        return price;
    }

    // метод с параметрами
    public void setPrice(int price) {
        this.price = price;
    }
}
```

Идентификаторы

Идентификаторы - это имена пакетов, классов, интерфейсов, объектов, полей, методов, переменных, параметров методов и т.д.

Названия идентификаторов выбираются по следующим правилам

- должны начинаться с буквы или символа “_” и “$”
- могут содержать латинские буквы, символы подчеркивания или цифры без пробелов
- названия идентификаторов не должны совпадать с ключевыми словами языка Java

Длина идентификатора в Java любая

IT-Academy

Объекты

Все объекты имеют одинаковые наборы полей данных (атрибуты объекта), но с независимыми значениями этих данных для каждого объекта

Значения полей данных объекта задают его состояние, а методы - его поведение. Сами объекты безымянны, и доступ к ним осуществляется только через ссылочные переменные

```
// тип     идентификатор      конструирование объекта
   Man          man          =          new Man();
```

Конструкторы

Конструктор - это метод класса, который инициализирует новый объект после его создания. Имя конструктора всегда совпадает с именем класса, в котором он расположен. У конструкторов нет типа возвращаемого результата - никакого, даже void

```
public class Man {
    // поля
    private String firstName;
    private String lastName;

    //конструкторы
    public Man() {
    }
    public Man(String firstName) {
        this.firstName = firstName;
    }
    public Man(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
}
```

IT-Academy

Конструкторы по умолчанию

```
public class Course {
    private String title;
    private String description;

    // no constructors

    public String getTitle() { return title; }

    public void setTitle(String title) { this.title = title; }

    public String getDescription() { return description; }

    public void setDescription(String description) { this.description = description; }

    public static void main(String[] args) {
        //вызов конструктора по умолчанию
        final Course course = new Course();
        course.setTitle("Java Course");
    }
}
```

Конструкторы с параметрами

```
public class Course {
    private String title;
    private String description;

    public Course(String title) {
        this.title = title;
    }

    public Course(String title, String description) {
        this.title = title;
        this.description = description;
    }

    public String getTitle() { return title; }

    public String getDescription() { return description; }

    public static void main(String[] args) {
        final Course course = new Course("Java Course", "Java Core Basics");
    }
}
```

Задание

Создайте класс "Круг", который будет содержать поля радиус и цвет. Создайте конструктор для
этого класса и методы для получения и изменения каждого из полей. Также создайте метод для
вычисления площади круга.

Статические методы/поля

Методы также могут быть статическими или нестатическими. Статический метод можно вызвать без создания экземпляра класса, а нестатический метод требует создания экземпляра класса. Например:

```
public static int addNumbers(int a, int b) {
  return a + b;
}
public int multiplyNumbers(int a, int b) {
  return a * b;
}
```