

Введение в ООП. Часть 1

НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

ЦЕЛЬ

Сформировать представление о классах и объектах

ПЛАН ЗАНЯТИЯ

ООП

КЛАССЫ

ОБЪЕКТЫ

Конструкторы

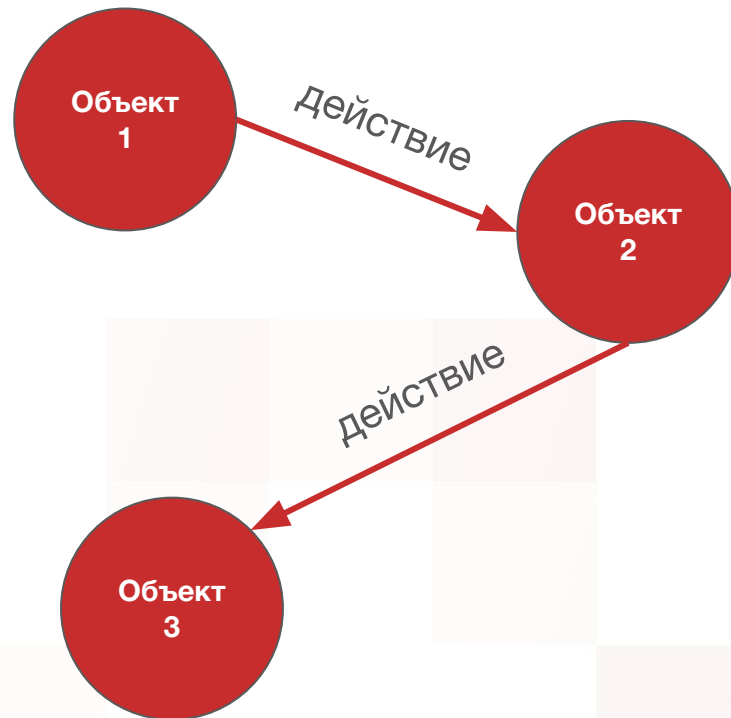
this

Массив объектов

ООП

Объектно-ориентированное программирование - подход в программировании, когда логика программы выстроена вокруг объектов, взаимодействующих между собой. Каждый объект создается на основе некоторого класса.

ООП позволяет структурировать программу для большего понимания и управления, обеспечивает повторное использование кода за счет полиморфизма и наследования и упрощает отладку и обслуживание кода.



КЛАССЫ

Класс - абстрактный тип данных, **шаблон**, на основе которых создаются **объекты**.

Внутри класса мы имеем возможность определить набор разнотипных переменных. Такие переменные определяют **состояние** будущих объектов и называются **полями**.

Абстракция - набор значимых характеристик какой-либо сущности в контексте решения задачи.

Класс является **ссылочным типом** данных

```
class Human {
```

```
    // поля
```

```
    int age;
```

```
    boolean isEmployed;
```

```
    String name;
```

```
}
```

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

Класс - как шаблон объекта

Абстракция

Поля

ОБЪЕКТЫ

Объект - конкретный **экземпляр** класса. Каждый объект имеет свои собственные значения полей

Переменная, созданная на основе класса **неформально** называется **объектной переменной** и может иметь **null** в качестве значения, либо **ссылаться** на созданный объект.

// h0, h1, h2 - объектные переменные

```
Human h0 = null;
```

// создание объектов

```
Human h1 = new Human();
```

```
Human h2 = new Human();
```

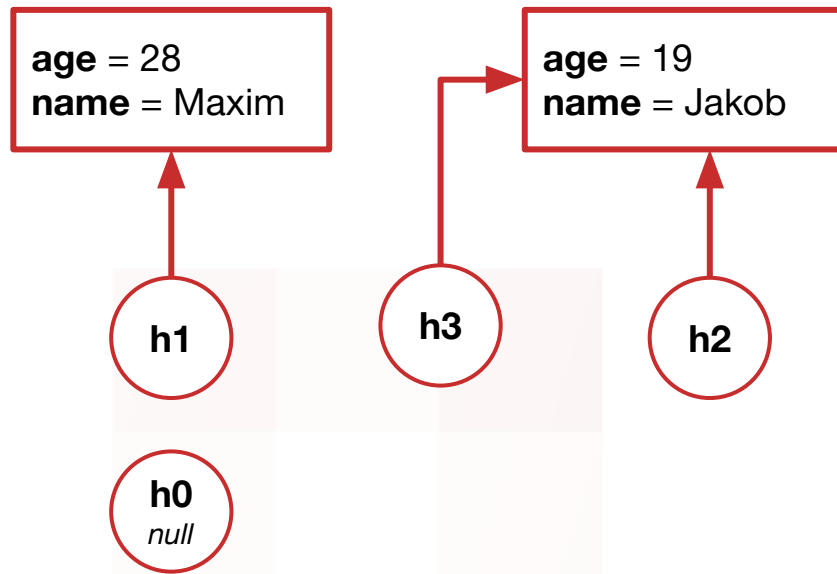
```
Human h3 = h2;
```

```
h1.age = 28;
```

```
h1.name = "Maxim";
```

```
h2.age = 19;
```

```
h2.name = "Jakob";
```



СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

■
Объект - экземпляр класса

■
Состояние объекта

■
Объектная переменная

КОНСТРУКТОРЫ

Конструктор - блок инструкций, выполняющий инициализацию объекта.

Инициализация - определение начального состояния объекта при его создании. Другими словами, внесение значений в поля создаваемого объекта.

Конструктор по умолчанию - пустой конструктор без параметров, автоматически добавляемый в каждый класс после компиляции.

В случае, если программист определяет **пользовательский конструктор**, конструктор по умолчанию не будет добавлен.

```
class Human {
```

```
    // поля
```

```
    int age;
```

```
    boolean isEmployed;
```

```
    String name;
```

```
    // собственный конструктор без параметров
```

```
    Human() {
```

```
        age = 18;
```

```
        isEmployed = true;
```

```
        name = "Anonymous";
```

```
    }
```

```
}
```

```
    // h1 ссылается на работающего человека 18 -ти лет с именем Anonymous
```

```
    Human h1 = new Human();
```

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

■
Конструктор

■
Конструктор по умолчанию

■
Пустой конструктор

КОНСТРУКТОРЫ

В классе может присутствовать несколько конструкторов, отличающихся формальными параметрами. Такие конструкторы называются **перегруженными**.


```
class Human {  
    // ...  
    // конструктор без параметров  
    Human() {  
        age = 18;  
        isEmployed = true;  
        name = "Anonymous";  
    }  
  
    // конструктор с параметрами  
    Human(int a, boolean emp, String n) {  
        age = a;  
        isEmployed = emp;  
        name = n;  
    }  
}
```

// имеем возможность использовать оба конструктора для создания объектов

```
Human h1 = new Human();
```

```
Human h2 = new Human(33, false, "Max");
```

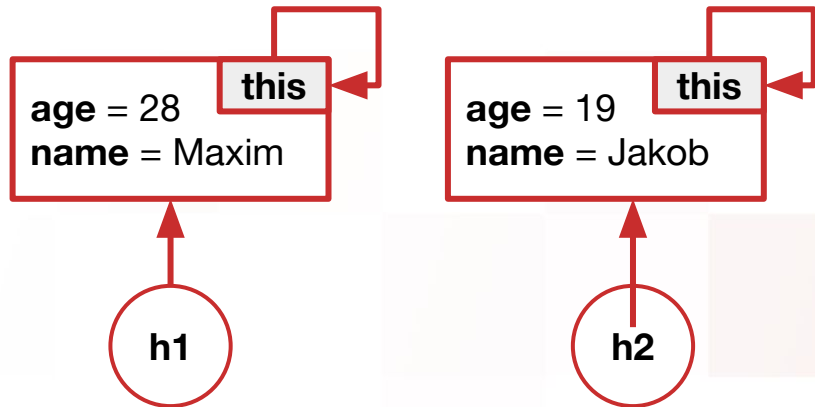
СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО



Перегруженные конструкторы и
необходимость их использования

this

this - ключевое слово, которое можно использовать в конструкторах (а также методах). Представляет собой ссылку на объект, для которого вызывается конструктор (или метод).



this позволяет обращаться к полям класса для устранения **коллизии имен**:

```
class Human {  
    // ...  
    Human(int age, boolean isEmployed, String name) {  
        this.age = age;  
        this.isEmployed = isEmployed;  
        this.name = name;  
    }  
}
```

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

this

Коллизия имен

this

Также с помощью **this** можно вызывать один конструктор из другого конструктора. Это позволяет избежать дублирования кода.

```
class Human {  
    // ...  
    Human() {  
        age = 18;  
        isEmployed = true;  
        name = "Anonymous";  
    }  
  
    Human(String name) {  
        this(); // вызов первого конструктора  
        this.name = name; // инициализация имени  
    }  
}
```


СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО



`this` для вызова конструктора внутри
конструктора

МАССИВ ОБЪЕКТОВ

В java для решения различных задач может понадобиться создание **массива объектов**.

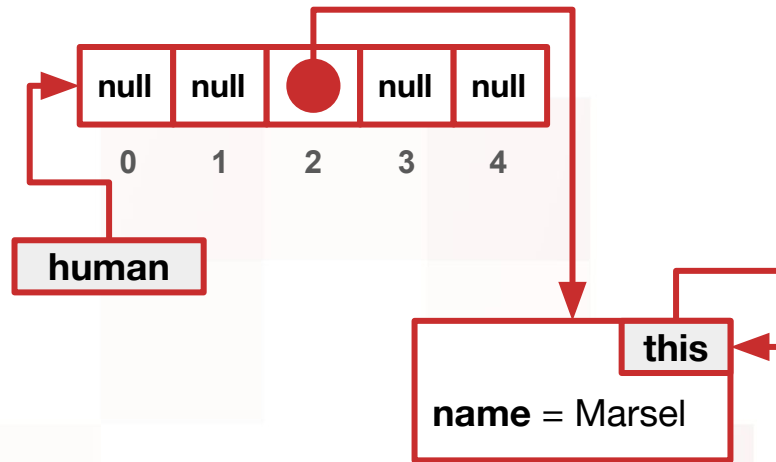
При создании такого массива важно понимать, что каждый элемент такого массива является объектной переменной и по умолчанию имеет значение **null**.

```
Human[] humans = new Human[5];
```



Для того, чтобы элемент массива указывал на объект, необходимо явно создать его:

```
humans[2] = new Human("Marsel");
```



СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО



Массив объектов



Инициализация массива объектов

ПОИГРАЕМ ;)

ООП

Класс

Объект

Конструктор

this

Массивы объектов

ДОМАШНЕЕ ЗАДАНИЕ

Переписать CurrencyConverter на ООП:

- сделать новый класс CurrencyConverter с полями `double[] currencyRates`, `String[] availableCurrencies`
- Не должно быть статических методов в CurrencyConverter
- Обращение к полям объекта должны быть через `this`.
- Должно быть 3 конструктора: один пустой (для задания значений по умолчанию), второй - `double[] currencyRates`, а третий должен принимать `double[] currencyRates`, `String[] availableCurrencies`
- В классе CurrencyConverter не должно быть ввода/вывода и цикла `while`, то есть вам нужно вынести весь ввод вывод и интерактивность в другое место (например в отдельный класс CurrencyConverterCLI)

Все должно быть на гитхабе в репозитории CurrencyConverter



Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH