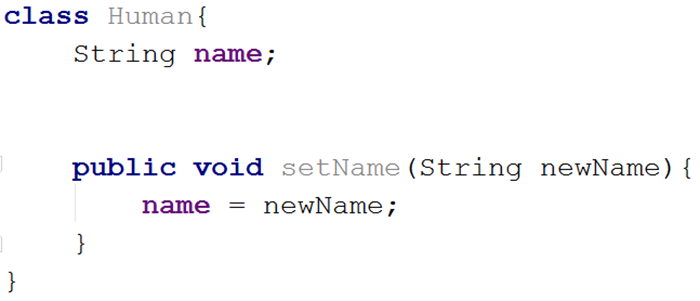
Ключевое слово this

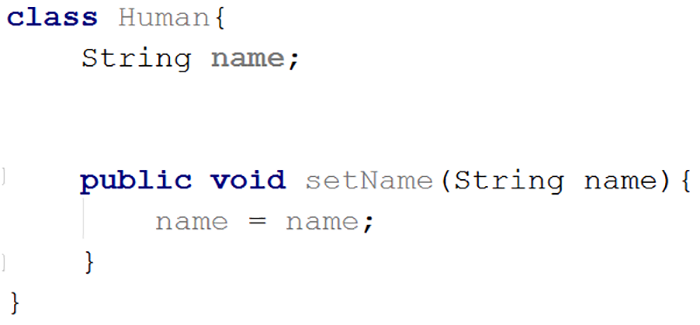
Используется если:

**- this.переменная –** означает, что эта переменная это переменная класса, а не переменная этого метода.

**— у переменной экземпляра и метода одинаковые имена**

Допустим, у нас есть класс Human, для которого определено поле «имя»: Давайте для переменной name создадим сеттер (setter вполне рабочий и никакого подвоха здесь нет): Обратите внимание, что в метод (сеттер) setName мы передаем переменную String newName. Мы ввели новую переменную и (в общем-то) могли назвать ее как угодно ведь она будет видна только в пределах {фигурных скобок} метода setName. Обратите внимание, что в сеттере есть одна строка:

name = newName;

То есть по факту мы ввели новую переменную newName и присвоили ее уже существующей в классе переменной name. Многим программистом казалось это странным, — вводить переменную с новым именем, если в итоге речь идет об одном и том же. Об имени в классе Human. Поэтому, разработчики языка задумались о том, чтобы удобно сделать использование одного имени переменной. Другими словами, зачем иметь два имени для переменной, обозначающей одно и то же. То есть хотелось бы сделать как-то так: 

Но в этом случае возникает проблема. У нас теперь две переменные, которые называются одинаково. Один String name принадлежит классу Human, а другой String name его методу setName. Поэтому Java – машина не знает, какую переменную вы имеете ввиду, когда пишете строку в сеттере:

name = name;

Java берет самую близкую – name из метода setName:

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:

То есть this сошлется на вызвавший объект, как было сказано в начале статьи. В результате чего имя человека через сеттер setName будет установлено создаваемому объекту. Ниже приведен программный код без использования ключевого слова this. В коде создается объект класса Human и присваивается ему имя:

А ниже программный код с ключевым словом this:

public class Solution{

public static void main(String[] args) {

Human human1 = new Human();

human1.setName("Volodya");

human1.print();

}

}

class Human{

String name;

public String getName() {

return name;

}

public void setName(String name){

**this**.name = name;

}

void print(){

System.out.println(name);

}

}

Таким образом, здесь this позволяет не вводить новые переменные для обозначения одного и того же, что позволяет сделать код менее «перегруженным» дополнительными переменными.

**— 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:

**class** Human{

**int** age;

**int** weight;

**int** height;

Human(**int** age, **int** weight){

**this**.age = age;

**this**.weight = weight;

}

Human(**int** age, **int** weight, **int** height){

//вы вызываете конструктор с двумя параметрами

**this**(age, weight);

//и добавляете недостающую переменную

**this**.height = height;

}

}

Здесь у нас сначала приводится конструктор с двумя параметрами, который принимает int age и int weight. Допустим, мы написали в нем две строчки кода:

**this**.age = age;

**this**.weight = weight;

а потом решили добавить еще один конструктор, с тремя параметрами, который помимо возраста и веса принимает еще и рост. В новом конструкторе вы бы могли написать так:

**this**.age = age;

**this**.weight = weight;

**this**.height = height;

Но вместо того, чтобы повторять уже написанный ранее код в этом конструкторе, вы можете с помощью ключевого слова this явно вызвать конструктор с двумя параметрами:

**this**(age, weight);

// и дописываете недостающую переменную:

**this**.height = height;

Таким образом, вы как-бы говорите конструктору с тремя параметрами:

* вызови this (этот) конструктор, который имеет два параметра.
* и добавить недостающую переменную.

- **неявно ключевое слово this передается во все методы, кроме статических** (поэтому this часто называют неявным параметром) и может быть использовано для обращения к объекту, вызвавшему метод.

Вы пишите такой код:

**class** **Foo** {

**void** **foo**(String str) { ... };

}

...

Foo f = **new** Foo();

f.foo("abc");

Компилятор же сгенерирует байткод, который примерно соответствует такому коду:

**class** **Foo** {

**static** **void** **foo**(Foo **this**, String str) { ... };

}

...

Foo f = **new** Foo();

Foo.foo(f, "abc");

this - это неявный первый параметр любого нестатического метода класса. У какого объекта вы метод вызываете, тот и будет передан в качестве параметра this.

- **если унаследоваться от класса A1 и сделать свой A2**, а в A2 вы сделать переменную такую же по имени как в A1 но другого типа, так вот “this” поможет использовать свою а не парент, а “super” будет использовать парент переменную.

**- использование this – может означать объект своего же класса. Это единственный способ для объекта передать себя** в качестве параметра при вызове метода или вернуть себя в качестве результата выполнения метода:

class Aaa

t=new Thread(**this**);

,где this - это ссылка на самого себя - то есть на экземпляр класса, в котором разворачивается действо.

Это получается тоже самое, что и

t=new Thread(**new Aaa()**);

или например

public MyClass registeredSelf() {

registrator.register(**this**); // Зарегистрируйте меня

**this**.registered = true;

return **this**;

}