super и this разница между ключевыми словами в Java

**this** и **super** - это два специальных ключевых слова в Java, которые представляют соответственно текущий экземпляр класса и его суперкласса.

Java-программисты часто путают эти слова и обнаруживают слабую осведомленность об их специальных свойствах, о которых нередко спрашивают на интервью по Java Сore. Вот, например, пара вопросов, из того, что сразу приходит на ум, о **this** и **super**:

- Можно ли присвоить другое значение ключевому слову **this** в Java?

- Какая разница между ключевыми словами **this** и **super** в Java?

Не знаете? Ну что ж, здесь я ответа не даю — его можно найти в конце статьи.

Так вот, как я уже сказал в начале, главное отличие между **this** и **super** в Java в том, что **this** представляет текущий экземпляр класса, в то время как **super** - текущий экземпляр родительского класса.

Вот один из примеров использования переменных **this** и **super** — вы наверняка уже видели примеры вызовов конструкторов одного из другого, т.н. вызовы конструкторов по цепочке, это возможно благодаря использованию ключевых слов **this** и **super**. Внутри класса для вызова своего конструктора без аргументов используется **this()**, тогда как **super()**используется для вызова конструктора без аргументов, или как его ещё называют, конструктора по умолчанию родительского класса. Между прочим, таким способом вызывать можно не только конструктор без аргументов, а и вообще любой другой конструктор, передав ему соответствующие параметры.

Скоро мы увидим пример такого использования **this** и **super**.

Ещё **this** и **super** в Java используются для обращения к переменным экземпляра класса и его родителя. Вообще-то, к ним можно обращаться и без префиксов **super** и **this**, но только если в текущем блоке такие переменные не перекрываются другими переменными, т.е. если в нем нет локальных переменных с такими же именами, в противном же случае использовать имена с префиксами придется обязательно, но это не беда, т.к. в таком виде они даже более читабельны. Классическим примером такого подхода является использование **this** внутри конструктора, который принимает параметр с таким же именем, как и у переменной экземпляра.

Дальше в статье мы узнаем, какие ещё есть отличия между **super** и **this**, и рассмотрим некоторые примеры их использования.

**Чем this и super похожи**

Прежде чем рассматривать отличия ключевых слов this и super, давайте посмотрим на некоторые их сходства:

1. И this, и super — это нестатические переменные, соответственно их нельзя использовать в статическом контексте, а это означает, что их нельзя использовать в методе main. Это приведет к ошибке во время компиляции "на нестатическую переменную this нельзя ссылаться из статического контекста". То же самое произойдет, если в методе main воспользоваться ключевым словом super.
2. И this, и super могут использоваться внутри конструкторов для вызова других конструкторов по цепочке, нпр., this() и super() вызывают конструктор без аргументов наследующего и родительского классов соответственно.

В примере ниже мы сначала передаем вызов из конструктора без аргументов класса B в конструктор этого же класса B, принимающий один параметр типа String, из которого, в свою очередь, с помощью super(**""**) вызывается конструктор с одним аргументом из суперкласса.

**class** A{

A(){

System.out.println("Конструктор без аргументов класса A");

}

A(String args){

System.out.println("Конструктор с одним аргументом класса A");

}

}

**class** B **extends** A{

B(){

**this**(""); // вызов конструктора с одним аргументом класса B

System.out.println("Конструктор без аргументов класса B");

}

B(String args){

**super**(""); // вызов конструктора с одним аргументом класса A

System.out.println("Конструктор с одним аргументом класса B");

}

}

// Тест-класс и вывод

**public** **class** Test {

**public** **static** **void** main(String args[]) {

B b = **new** B();

}

}

**Вывод: *Конструктор с одним аргументом класса A Конструктор с одним аргументом класса B Конструктор без аргументов класса B***

1. Внутри конструктора this и super должны стоять выше всех других выражений, в самом начале, иначе компилятор выдаст сообщение об ошибке. Из чего следует, что в одном конструкторе не может быть одновременно и this(), и super().

**Различия в super и this**

Теперь мы знаем как использовать ключевые слова super и this и понимаем для чего они нужны. Но есть ещё один вариант использования этих ключевых слов, о котором я не сказал - во Внутренних классах, где с их помощью очень удобно ссылаться на внешний класс, используя форму записи Outer.this для его текущего экземпляра и Outer.super — для его родителя. Не забудьте вместо Outer подставить имя внешнего класса. А теперь давайте кратко перечислим главные отличия между ключевыми словами this и super

1. переменная this ссылается на текущий экземпляр класса, в котором она используется, тогда как super — на текущий экземпляр родительского класса.

1. Каждый конструктор при отсутствии явных вызовов других конструкторов неявно вызывает с помощью super() конструктор без аргументов родительского класса, при этом у вас всегда остается возможность явно вызвать любой другой конструктор с помощью либо this(), либо super().

Пожалуй, это все, что можно сказать о различиях между ключевыми словами this и super в Java и о том, как они используются в программах. Как мы увидели, основное их назначение - вызывать один конструктор из другого и ссылаться на переменные экземпляра, объявленные в текущем классе и его родительском классе. Не забывайте, что это не совсем обычные переменные, а сейчас - ответ на мой вопрос, который я задавал в первом параграфе. Нет, переменной this нельзя присвоить новое значение, потому что она объявлена как final. Можете попробовать сделать это в IDE - получите ошибку компиляции "нельзя присвоить новое значение переменной this — она объявлена как final".