

Понятие конструктора

Чем отличается конструктор от метода.

Beginner Java

Роберт Нильсен

Чтобы изучить Java, вы должны разобраться с понятием конструктора. Так как конструкторы имеют схожие характеристики с методами, новички часто путают их друг с другом. Однако, конструкторы и методы имеют важные отличия. Если вы не знакомы с этими фундаментальными принципами, то вам необходимо прочитать эту ознакомительную статью. *(1200 слов)*

Говорить что конструктор — это метод, все равно что говорить что австралийский утконос — просто млекопитающее. Чтобы понять кто такой утконос, необходимо знать чем он отличается от других млекопитающих. Чтобы понять сущность конструктора, аналогично, необходимо понять чем он отличается от простого метода. Любой, изучающий Java, в особенности для сертификации, должен знать эти отличия. В конце статьи я свел ключевые отличия конструкторов и методов в результирующую таблицу .

1. Назначение и функции

Конструкторы имеют только одно назначение — создать экземпляр класса. Или, как еще говорят, создать объект, как здесь:

```
Platypus p1 = new Platypus();
```

Назначение методов гораздо более общее. Основная функция метода — выполнить Java код.

2. Отличия описания

Конструкторы и методы имеют три отличия при описании: модификаторы доступа, возвращаемые типы и правила именования. Как и методы, конструкторы могут иметь любые модификаторы доступа: `public`, `protected`, `private`, или модификатор может отсутствовать (часто имеется ввиду `package` или `friendly`). Но в отличие от методов, конструкторы могут иметь модификаторы только доступа. Поэтому, конструктор не может быть `abstract`, `final`, `native`, `static`, or `synchronized`.

Возвращаемые типы тоже отличаются. Методы могут возвращать любой правильный тип, или ничего не возвращать, в данном случае возвращаемый тип описывается как `void`. Конструкторы же не имеют возвращаемого типа, они не могут возвращать даже тип `void`.

И в заключение, методы и конструкторы имеют различные имена. Конструкторы имеют одинаковые имена с именем класса в котором описаны, а методы, по договоренности, имеют имена отличные от имени класса. Если Java программа написана по правилам языка, имена методов начинаются с маленькой буквы, конструкторов — с большой. И еще, имя конструктора является существительным, так как имена классов обычно являются существительными, имена же методов обычно являются глаголами.

3. Использование ключевого слова "this"

Конструкторы и методы используют ключевое слово `"this"` совершенно по-разному. Метод использует `this` чтобы получить ссылку на экземпляр класса выполняющего этот метод. Статические методы не используют ключевое слово `"this"`, т.к. они не принадлежат экземпляру класса, поэтому `this` некуда ссылаться. Статические методы принадлежат классу как целому, но никак не экземпляру класса. Конструкторы используют `"this"` чтобы сослаться на другой конструктор в этом же классе, но с другим списком параметров. Ознакомьтесь с примером:

```
public class Platypus {
```

Понятие конструктора

```
String name;

Platypus(String input) {
    name = input;
}

Platypus() {
    this("John/Mary Doe");
}

public static void main(String args[]) {
    Platypus p1 = new Platypus("digger");
    Platypus p2 = new Platypus();
}
}
```

В данном примере имеется два конструктора. Первый получает строку-аргумент в качестве имени экземпляра класса. Второй не получает никаких аргументов, он просто вызывает первый конструктор используя имя "John/Mary Doe" по-умолчанию.

Если конструктор использует ключевое слово "this", то оно должно быть в первой строке, игнорирование этого правила приведет к ошибке компилятора.

4. Использование ключевого слова "super"

И методы и конструкторы используют `super` для ссылки на супер-класс (класс предок), но разными способами. Методы используют `super` для выполнения какого-либо переопределенного метода супер-класса, как показано в следующем примере:

```
class Mammal {
    void getBirthInfo() {
        System.out.println("born alive.");
    }
}

class Platypus extends Mammal {
    void getBirthInfo() {
        System.out.println("hatch from eggs");
        System.out.print("a mammal normally is ");
    }
}
```

```
        super.getBirthInfo();  
    }  
}
```

Эта программа вызывает `super.getBirthInfo()` переопределенный метод супер-класса `Mammal`.

Конструкторы используют `super` чтобы вызвать конструктор супер-класса. Если конструктор использует `super`, то этот вызов должен быть в первой строке, иначе компилятор выдаст ошибку. Ниже приведен пример:

```
public class SuperClassDemo {  
    SuperClassDemo() {}  
}  
  
class Child extends SuperClassDemo {  
    Child() {  
        super();  
    }  
}
```

В этом простом примере конструктор `Child()` содержит вызов `super`, который создает экземпляр класса `SuperClassDemo`, в дополнение к классу `Child`.

5. Код генерируемый компилятором

Начинающий Java программист может быть озадачен когда компилятор автоматически дополнит исходный код конструкторами. В тех случаях, когда вы пишете код класса не содержащий конструкторов, компилятор автоматически дополнит его конструктором без аргументов. Т.е. если вы напишете:

```
public class Example {}
```

это эквивалентно написанию:

```
public class Example {  
    Example() {}  
}
```

Компилятор автоматически дополняет код не содержащий вызовов `super`, добавляя вызов `super` с несколькими параметрами или вообще без параметров в первую строку

Понятие конструктора

конструктора. Таким образом, если вы напишете

```
public class TestConstructors {
    TestConstructors() {}
}
```

это будет идентично написанию:

```
public class TestConstructors {
    TestConstructors() {
        super;
    }
}
```

Наблюдательный новичок может быть удивлен тому, как вышеописанная программа может вызывать конструктор предка, когда `TestConstructor` не наследует ни одного класса. Ответом является то, что Java наследует класс `Object` в том случае, когда явно не указан класс предок. Компилятор автоматически добавляет конструктор без аргументов, если явно не описан конструктор, и автоматически дополняет вызовом `super` без аргументов, в случае отсутствия явного вызова `super`. Таким образом, следующие два примера являются функционально эквивалентными:

```
public class Example {}
```

и

```
public class Example {
    Example() {
        super;
    }
}
```

6. Наследование

Что не правильно в следующем сценарии? Юрист читает завещание некоего `Class A`. Члены семьи `Class A`. собрались вокруг большого конференц-стола, кто-то тихо рыдает. Юрист читает, — "Я, `Class A`., в чистом уме и здравии, оставляю все свои конструкторы своим детям!"

Проблема в том, что эти конструкторы не могут быть наследованы. К счастью для детей `Class A`., автоматически наследуют все методы родителя, таким образом дети

Class A. не останутся совсем обездоленными.

Помните что методы в Java наследуются, а конструкторы — нет! Рассмотрите следующий класс:

```
public class Example {
    public void sayHi {
        system.out.println("Hi");
    }

    Example() {}
}

public class SubClass extends Example {
}
```

Класс SubClass автоматически наследует метод sayHi определенный в родительском классе. В тоже время, конструктор Example () родительского класса не наследуется его потомком SubClass.

7. Подводя итоги

Так же как утконос отличается от обычного млекопитающего, так же конструкторы отличаются от методов; особенно по их назначению, описанию, и способам использования методов this и super. И в дополнение к этому, конструкторы отличаются отношением к наследованию и кодом, генерируемым компилятором. В следующей ниже таблице все эти детали собраны воедино для простоты восприятия. Также вы можете найти дополнительные ресурсы посвященные конструкторам и методам в разделе ресурсов.

Свойство	Конструкторы	Методы
Назначение	Создает экземпляр класса	Группирует операторы Java
Модификаторы	Не может быть abstract, final, native,static, или synchronized	Может быть abstract, final, native,static, или synchronized
Возвращаемый тип	Нет возвращаемого типа, не может быть даже void	void или любой корректный тип

Понятие конструктора

Имя	Такое же как и имя класса (по договоренности, первая буква — заглавная) — обычно существительное	Любое имя, за исключением имени класса. Имена методов начинаются со строчной буквы, по договоренности, и обычно являются глагол
<code>this</code>	Ссылается на другой конструктор в этом же классе. Если используется, то обращение должно к нему быть первой строкой конструктора	Ссылается на экземпляр класса-владельца. Не может использоваться статическими методами
<code>super</code>	Вызывает конструктор родительского класса. Если используется, должно обращение к нему быть первой строкой конструктора	Вызывает какой-либо переопределенный метод в родительском классе
Наследование	Конструкторы не наследуются	Методы наследуются
Автоматическое добавление кода компилятором конструктора	Если в классе не описан конструктор, компилятор автоматически добавляет в код конструктор без параметров	Отсутствует
Автоматическое добавление компилятором вызова конструктора класса-предка	Если конструктор не делает вызов конструктора <code>super</code> класса-предка (с аргументами или без аргументов), компилятор автоматически добавляет код вызова конструктора класса-предка без аргументов	Отсутствует

Таблица 1. Различия между конструкторами и методами.

8. Об авторе

[Robert Nielsen](#) is a Sun Certified Java 2 Programmer. He holds a master's degree in education, specializing in computer- assisted instruction, and has taught in the computer field for several years. He has also published computer-related articles in a variety of magazines.

9. Ресурсы

- Some books that cover the basics of constructors and methods are:
 - *The Complete Java 2 Study Certification Guide*, Simon Roberts et al. (Sybex, 2000):
<http://www.amazon.com/exec/obidos/ASIN/0782128254/qid=969399182/sr=1-2/102-9220485-9634548>
 - *Java 2 (Exam Cram)*, Bill Brogden (The Coriolis Group, 1999):
<http://www.amazon.com/exec/obidos/ASIN/1576102912/qid%3D969399279/102-9220485-9634548>
 - *Java in a Nutshell*, Davis Flanagan (O'Reilly & Associates, 1999):
<http://www.amazon.com/exec/obidos/ASIN/1565924878/o/qid=969399378/sr=2-1/102-9220485-9634548>
- Visit the Sun Microsystems Website for more coverage of methods and constructors:
<http://java.sun.com/docs/books/tutorial/trailmap.html>
- For more Java content for beginners, read *JavaWorld's* new **Java 101** column series:
<http://www.javaworld.com/javaworld/topicalindex/jw-ti-java101.html>

Reprinted with permission from the October 2000 edition of JavaWorld magazine. Copyright © ITworld.com, Inc., an IDG Communications company.

View the original article at: <http://www.javaworld.com/javaworld/jw-10-2000/jw-1013-constructors.html>

[Перевод на русский © Александр Серебряков, 2000](#)