Инициализация полей в Java (Java fields initialization)

headtail Введение

Как известно, в Java поля (fields) могут принадлежать классу*(static)* или объекту*(не static)* (а методу*(локальные)*??). Поля, принадлежащие классу, являются статическими, а поля, принадлежащие объекту, - нестатическими. Статические поля доступны без создания объекта класса. Соответственно инициализироваться статические и нестатические поля должны в разное время: одни до создания объекта класса, а другие после.

Типы инициализации полей объектов и классов

Существуют следующие методы инициализации полей:

|  |  |  |
| --- | --- | --- |
| **Название** | **Применимость** | **Описание** |
| Инициализация в месте объявления поля | Поля класса, поля объекта | Применяется, если инициализация может быть произведена коротким выражением и доступен контекст, необходимый для ее проведения |
| Инициализационный блок | Поля класса, поля объекта | Применяется, если инициализационный код неудобно записывать одним выражением или же, например, нужна обработка проверяемых исключений. В случае объектов может применяться для инициализации полей объектов анонимных классов. |
| Конструктор класса | Поля объекта | Применяется, если для инициализации нужны параметры конструктора |

Далее мы рассмотрим каждый тип инициализации подробнее.

Инициализация статических полей в месте объявления

Начнем с примера:

class Integer {

...

public static final int SIZE = 32;

...

}

Здесь инициализируется статическое поле SIZE класса Integer. Сама инициализация произойдет во время загрузки класса. В этом легко убедиться выполнив следующий код:

public class StaticInitializationTime {

public static class C {

static int i = value();

static int value() {

System.out.println("C.i initialized");

return 1;

}

}

public static void main(String[] args) throws ClassNotFoundException {

System.out.println("Before class loading");

Class.forName(C.class.getName());

System.out.println("After class loading");

}

}

В результате будет выведено:

Before class loading

C.i initialized

After class loading

Как видно из вывода переменная *i* инициализируется в результате загрузки класса. Попробуйте выполнить код, закомментировав строку, содержащую Class.forName(...).

Инициализация в статическом блоке

В некоторых случаях инициализацию неудобно проводить в месте объявления переменной. Например, если в результате выполнения инициализирующего выражения происходит проверяемое исключение. Или же, если инициализация производится путем выполнения кода, который не может быть представлен в виде выражения. Для таких случаев в Java предусмотрен специальный языковой элемент - инициализационный статический блок. Покажем на примере:

static List<Character> alphabet;

static {

alphabet = new ArrayList<Character>();

for (char c='a'; c<='z'; c++) alphabet.add(c);

}

Переменная alphabet инициализируется в статическом блоке. Инициализация происходит во время загрузки класса аналогично как и в предыдущем примере.

Инициализация статических полей в месте объявления и статические блоки выполняются в порядке их объявления в классе. Давайте выполним следующий код:

public class ClassFieldsInitOrder {

static int i1 = initialize("i0");

static int i2;

static { i2 = initialize("i1"); }

static int i3 = initialize("i2");

static int i4;

static { i4 = initialize("i4"); }

static int initialize(String name) {

System.out.println(name);

return 0;

}

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

}

На консоль будет выведено:

i0

i1

i2

i4

Инициализация полей объекта

В отличии от полей класса, поля объекта инициализируются во время конструирования экземпляра класса. В Java существует 3 типа такой инициализации:

* инициализация в месте объявления
* инициализация в нестатическом блоке
* инициализация в конструкторе

Инициализация полей объекта в месте объявления

Покажем на примере:

public class Blog {

...

private List<Post> posts = new ArrayList<Post>

...

}

Инициализация поля posts будет произведена во время конструирования объекта Blog.

Инициализация полей объекта в нестатическом блоке

Использование инициализационных блоков является альтернативой предыдущему способу инициализации. Данный тип инициализации может использоваться, если:

* во время инициализации необходимо обработать проверяемое исключение
* значение поля не удобно вычислять с помощью выражения (например, для этого нужен специально созданный класс или метод, который не хочется создавать только для этих целей)
* необходимо инициализировать поле анонимного класса (в анонимном классе невозможно объявить конструктор)

Пример обработки проверяемого исключения:

class Year2000Problem {

Date start;

{

try { start = new SimpleDateFormat("dd.MM.yyyy").parse("01.01.2000"); }

catch (ParseException impossible) {}

}

}

Инициализация полей объекта в конструкторе

Часто инициализацию полей объекта имеет смысл проводить только с учетом значений параметров конструктора. В таких случаях ее производят в самом конструкторе. Пример:

class User {

...

String login;

User(String login) { this.login = login; }

...

}

Инициализация в конструкторе и наследование

Выполним следующий код:

class A  
{  
 int ns1 = init("ns1", "[Class A] In non-static fields");  
 int ns2 = init("ns2", "[Class A] In non-static fields");  
 int ns3;  
 int ns4;  
  
 static int s1 = init("s1", "[Class A] In static fields");  
 static int s2 = init("s2", "[Class A] In static fields");  
 static int s3;  
  
 A()  
 {  
 ns4 = init("ns4", "[Class A] In constructor");  
 }  
  
  
 {  
 ns3 = init("ns3", "[Class A] In non-static initialization block");  
 }  
  
 static  
 {  
 s3 = init("s3", "[Class A] In static initialization block");  
 }  
  
 static int init(String msg, String whereInit)  
 {  
 System.out.println(msg + " " + whereInit);  
 return 0;  
 }   
}

class B extends A  
{  
 int ns1B = init("ns1B", "[Class B] In non-static fields");  
 int ns2B = init("ns2B", "[Class B] In non-static fields");  
 int ns3B;  
 int ns4B;  
  
 static int s1B = init("s1B", "[Class B] In static fields");  
 static int s2B = init("s2B", "[Class B] In static fields");  
 static int s3B;  
  
 B()  
 {  
 ns4B = init("ns4B", "[Class B] In constructor");  
 }  
  
 {  
 ns3B = init("ns3B", "[Class B] In non-static initialization block");  
 }  
  
 static  
 {  
 s3B = init("s3B", "[Class B] In static initialization block");  
 }  
  
 static int init(String msg, String whereInit)  
 {  
 System.out.println(msg + " " + whereInit);  
 return 0;  
 }

public static void main(String[] args) throws ClassNotFoundException   
 {   
 B testB = new B();   
 }

В итоге в выводе получаем:

s1 [Class A] In static fields  
s2 [Class A] In static fields  
s3 [Class A] In static initialization block  
s1B [Class B] In static fields  
s2B [Class B] In static fields  
s3B [Class B] In static initialization block  
ns1 [Class A] In non-static fields  
ns2 [Class A] In non-static fields  
ns3 [Class A] In non-static initialization block  
ns4 [Class A] In constructor  
ns1B [Class B] In non-static fields  
ns2B [Class B] In non-static fields  
ns3B [Class B] In non-static initialization block  
ns4B [Class B] In constructor

**Вывод:**1. Инициализируются статические поля и блоки статической инициализации   
суперкласса;  
2. Инициализируются статические поля и блоки статической инициализации   
подкласса;  
3. Инициализируются нестатические поля и блоки нестатической инициализации + происходит  
 инициализация в конструкторе суперкласса *(именно в такой последовательности)*;  
4. И только потом происходит Инициализация нестатических полей, выполнение блока нестатической инициализации и инициализация в конструкторе подкласса *(именно в такой последовательности)*.

Константы **public static final** – инициализируются один (самый первый) раз при создании класса и дальше хранятся где-то в отдельном пуле для констант, без привязки к отдельным объектам, классам и т.д.

--  
Желаю удачи!  
Дмитрий Пекар, июнь 2009

Если Вам понравилась статья, проголосуйте за нее

[Голосов](http://www.quizful.net/post/java-fields-initialization): 81 Голосовать

|  |  |  |
| --- | --- | --- |
|  | [Поделиться](http://vk.com/share.php?url=http%3A%2F%2Fwww.quizful.net%2Fpost%2Fjava-fields-initialization) |  |

[Share](http://www.facebook.com/sharer.php?u=www.quizful.net%2Fpost%2Fjava-fields-initialization&t=%D0%98%D0%BD%D0%B8%D1%86%D0%B8%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F%20%D0%BF%D0%BE%D0%BB%D0%B5%D0%B9%20%D0%B2%20Java%20(Java%20fields%20initialization))

Комментариев: 19

[↑обновить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#4006289234525578030)**http://www.quizful.net/user/dionisius1976?avatar  dionisius1976**16.01.2016 | 13:08:03

Добрый день.  
Как понять "((B)this).b" в коде   
  
public class InheritanceInitOrder {  
static class A {  
String a;  
A() {  
a = "a";  
System.out.println("a initialized");  
System.out.println("b=" + ((B)this).b);  
}  
}  
  
static class B extends A {  
String b;  
B() {  
b = "b";  
System.out.println("b initialized");  
System.out.println("b=" + b);  
}  
}  
  
???

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#6585047921394848433)**http://www.quizful.net/user/shark19?avatar  shark19**14.02.2016 | 14:52:28

мы обращаемся к переменной b класса B этого экземпляра, чтобы убедиться, что она равна null :)

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#5924089707219598391)**http://www.quizful.net/user/soundtrack?avatar  soundtrack**12.04.2014 | 21:20:03

расскажите, пожалуйста, про порядок инициализации static final полей класса. Насколько я понимаю, они инициализируются ещё раньше нефинальных статиков. Но каков между ними порядок? Объясните, пожалуйста, эти 2 примера. (в обоих случаях во время вызова "геттера" значение уже присвоено вне зависимости от порядка записи)  
public class TestClass {   
static final int f = 5;  
static final int b = getf();  
public static int getf() {  
return f;  
}  
public static int getb() {  
return b;  
}   
public static void main(String[] args) {   
System.out.println(TestClass.f);  
System.out.println(TestClass.b);  
}   
  
}   
  
и второй:   
  
public class TestClass {   
static final int f = 5;  
static final int b = getf();  
public static int getf() {  
return f;  
}  
public static int getb() {  
return b;  
}  
public static void main(String[] args) {   
System.out.println(TestClass.f);  
System.out.println(TestClass.b);  
}   
}

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#6946349923431690514)**http://www.quizful.net/user/ksp107?avatar  ksp107**13.04.2014 | 01:27:45

В некоторых случаях поля с модификатором final (в т.ч. и static final) являются не обычными полями, а константами. Это значит, что их инициализация вообще не выполняется, а компилятор вместо обращения к этим полям сразу подставляет их конкретное значение.  
В вашем примере константой будет поле  
| static final int f = 5;  
а команды вида  
| return f;  
| System.out.println(TestClass.f);  
сразу компилируются как  
| return 5;  
| System.out.println(5);

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#6037251268921313597)**http://www.quizful.net/user/cane?avatar  cane**28.02.2013 | 19:54:40

Помимо описанных 3-х способов, инициализировать поля экземпляра класса можно с помощью финальных методов:  
  
public class Whatever {  
protected List<String> allSeasons = initializeInstanceVariable();  
private final List<String> initializeInstanceVariable() {  
return Arrays.asList("winter", "spring", "summer", "autumn");  
}  
  
public static void main(String[] args) {  
Whatever w = new Whatever();  
for (String s : w.allSeasons) {  
System.out.println(s);  
}  
}  
}

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#779496153953193353)**http://www.quizful.net/user/Cyanide?avatar  Cyanide**23.10.2013 | 14:33:06

private String text = inputText();  
private String inputText() {  
FileInputStream in = null;  
byte[] result = null;  
try {  
in = new FileInputStream("d:Data.txt");  
result = new byte[in.available()];  
while (true) {  
int count = in.read(result);  
if (count == -1) {  
break;  
}  
}  
} catch (IOException e) {  
System.err.print("IOExeption in textReader()");  
} finally {  
try {  
in.close();  
} catch (IOException e) {  
System.err.print("IOExeption in.close()");  
}  
}  
return new String(result);  
}  
так тоже работает, но не всегда. кто знает почему?

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#4536892094671120715)**http://www.quizful.net/user/Gorbush?avatar  Gorbush**13.06.2011 | 22:05:00

Кстати - JAD декомпилирует неверно - на него лучше не ориентироваться.  
Если чуть доразвить класс InitOrder4Jad, то jad после компиляции декомпиленного выдаст другой результат.  
Если дополнить класс в самом конце файла класса  
static int i5 = setTo("i5", 1);  
  
public static int setTo(String info, int val){  
System.out.println(info+" TO "+val);  
return val;  
}  
  
Вызов выдаст  
i0 TO 1  
i1 TO 0  
i1 TO 1  
i1 TO 2  
i1 TO 3  
i1 TO 4  
i1 TO 5  
i1 TO 6  
i1 TO 7  
i1 TO 8  
i1 TO 9  
i5 TO 1  
i2 TO 1  
i3 TO 1  
i4 TO 1  
  
А JAD создаст файл в котором   
инициализатор i1 (статический блок) будет в самом конце, после инициализатора i5

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#6330230840320766817)**http://www.quizful.net/user/NutsDev?avatar  NutsDev**19.12.2010 | 11:49:16

"инициализация полей базового класса происходит полностью до инициализации производного класса, т.е. сначала выполняются все инициализаторы базового класса, а потом все инициализаторы производного класса."  
Я немного развил ваш пример:  
  
class A  
{  
int ns1 = init("ns1", "[Class A] In non-static fields");  
int ns2 = init("ns2", "[Class A] In non-static fields");  
int ns3;  
int ns4;  
  
static int s1 = init("s1", "[Class A] In static fields");  
static int s2 = init("s2", "[Class A] In static fields");  
static int s3;  
  
A()  
{  
ns4 = init("ns4", "[Class A] In constructor");  
}  
  
  
{  
ns3 = init("ns3", "[Class A] In non-static initialization block");  
}  
  
static  
{  
s3 = init("s3", "[Class A] In static initialization block");  
}  
  
static int init(String msg, String whereInit)  
{  
System.out.println(msg + " " + whereInit);  
return 0;  
}   
}

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#2112786352187181987)**http://www.quizful.net/user/NutsDev?avatar  NutsDev**19.12.2010 | 11:48:35

class B extends A  
{  
int ns1B = init("ns1B", "[Class B] In non-static fields");  
int ns2B = init("ns2B", "[Class B] In non-static fields");  
int ns3B;  
int ns4B;  
  
static int s1B = init("s1B", "[Class B] In static fields");  
static int s2B = init("s2B", "[Class B] In static fields");  
static int s3B;  
  
B()  
{  
ns4B = init("ns4B", "[Class B] In constructor");  
}  
  
{  
ns3B = init("ns3B", "[Class B] In non-static initialization block");  
}  
  
static  
{  
s3B = init("s3B", "[Class B] In static initialization block");  
}  
  
static int init(String msg, String whereInit)  
{  
System.out.println(msg + " " + whereInit);  
return 0;  
}   
}

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#1564403998219293609)**http://www.quizful.net/user/NutsDev?avatar  NutsDev**19.12.2010 | 11:47:41

public static void main(String[] args) throws ClassNotFoundException   
{   
B testB = new B();   
}  
  
В итоге в выводе получаем:  
s1 [Class A] In static fields  
s2 [Class A] In static fields  
s3 [Class A] In static initialization block  
s1B [Class B] In static fields  
s2B [Class B] In static fields  
s3B [Class B] In static initialization block  
ns1 [Class A] In non-static fields  
ns2 [Class A] In non-static fields  
ns3 [Class A] In non-static initialization block  
ns4 [Class A] In constructor  
ns1B [Class B] In non-static fields  
ns2B [Class B] In non-static fields  
ns3B [Class B] In non-static initialization block  
ns4B [Class B] In constructor

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#9156503135783102374)**http://www.quizful.net/user/NutsDev?avatar  NutsDev**19.12.2010 | 11:46:15

Следовательно порядок несколько отличен от того к-ый вы указали, а имеено:  
1. Иниициализация статических полей, выполнение блока статической инициализации   
суперкласса;  
2. Иниициализация статических полей, выполнение блока статической инициализации   
подкласса;  
3. Иниициализация нестатических полей, выполнение блока нестатической инициализации   
и инициализация в конструкторе суперкласса;  
4. Иниициализация нестатических полей, выполнение блока нестатической инициализации   
и инициализация в конструкторе подкласса.

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#6171580194820696587)**http://www.quizful.net/user/reyand?avatar  reyand**23.04.2012 | 14:52:01

Да, сначала действительно инициализируются статические переменные и блоки класса-родителя, затем потомка, а лишь потом все остальные блоки инициализации (в той же последовательности: родитель -> потомок).

[ответить](http://www.quizful.net/post/java-fields-initialization)

[#](http://www.quizful.net/post/java-fields-initialization#6975943443857014803)**http://www.quizful.net/user/AlexanderSib?avatar  AlexanderSib**12.11.2010 | 13:57:50

Как видно из вывода переменная i инициализируется в результате загрузки класса. Попробуйте выполнить код, закомментировав строку, содержащую Class.forName(...).  
  
Это не совсем верно, инициализация статических переменных не происходит при загрузке класса. Например если заменить строку Class.forName(C.class.getName()) на System.out.println(C.class.getName()), то класс будет загружен, но при этом переменная i не будет инициализирована. Просто сам метод Class.forName производит инициализацию класса, есть ещё одна перегруженная версия этого метода с дополнительными параметрами в которых можно указать инициализировать класс при загрузке или нет.