JAVA PROGRAMMING BASICS

Module 2: Java Object-oriented Programming

Training program

1. Classes and Instances
2. The Methods
3. The Constructors
4. Static Elements
5. **Initialization sections**
6. Package
7. Inheritance and Polymorphism
8. Abstract classes and Interfaces
9. String processing
10. Wrapper classes for primitive types
11. Exceptions and Assertions
12. Nested classes
13. Enums
14. Generics
15. Collections
16. Method overload resolution
17. Multithreads
18. Core Java classes
19. Object Oriented Design
20. Functional Programming

Module contents

1. Initialization sections
   - Class fields initialization
   - Nonstatic initialization block
   - Static initialization block
   - Order of Initialization block execution
   - Initialization of final variable

Module contents

1. Initialization sections
   - Class fields initialization
   - Nonstatic initialization block
   - Static initialization block
   - Order of Initialization block execution
   - Initialization of final variable

Class fields initialization 1/5

- Fields that are declared but not initialized will be set to a reasonable default by the compiler

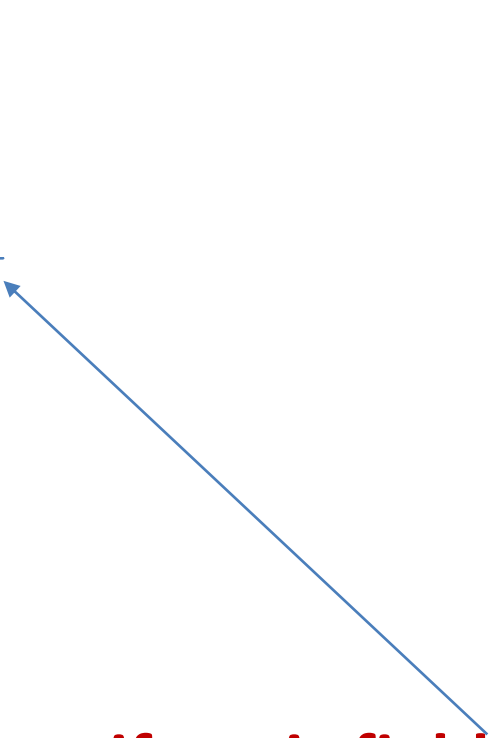
Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	“\u0000”
String (or any object)	null
boolean	false

See defaultfieldsinit.ClassFieldsDefaultInit & ClassStaticFieldsDefaultInit

Class fields initialization 3/5

- You can often provide an initial value for a field in its declaration:

```
1.  public class ClassInitDemo1 {
2.      private static boolean b = true;
3.      private static byte by = 127;
4.      private static int i = 2000;
5.      private static double d = 5.89;
6.      private static char c = 'A';
7.      private static String st = "Hi !";
8.      private static int[] arr = {1,2,3};
9.          //...
10. }
```



javap –c имя_класса → **{} ≡ <clinit> - if static field init**

See defaultfieldsinit.ClassStaticFieldsInit

Class fields initialization 4/5

- You can specify the name of a previously declared class field's in the expression of a declared class field's.

See defaultfieldsinit.ClassStaticFieldsInitUse

- You can call static methods for fields initialization.

See defaultfieldsinit.ClassStaticFieldsInitByMethod

main() method is called last.

Module contents

1. Initialization sections
   - Class fields initialization
   - Non-static initialization block
   - Static initialization block
   - Order of Initialization block execution
   - Initialization of final variable

Non-static initialization block 1/3

- Initializer blocks for instance variables used to share a block of code between multiple constructors:

See nonstaticinitblockpkg.SharedInitBlockCar

this ~~**return**~~

copies init block to each constructor,
see javap -c

Module contents

1. Initialization sections
   - Class fields initialization
   - Nonstatic initialization block
   - Static initialization block
   - Order of Initialization block execution
   - Initialization of final variable

Static initialization block 1/2

- A static initialization block is a normal block of code enclosed in braces { }, and preceded by the static keyword.

See staticinitblockpkg.StaticInitBlock

- A static initialization blocks are used for class with complex initialization.

See staticinitblockpkg.ComplexClassInit

~~this~~ ~~return~~

Module contents

1. Initialization sections
   - Class fields initialization
   - Nonstatic initialization block
   - Static initialization block
   - Order of Initialization block execution
   - Initialization of final variable

See initblocksorder.InitBlocksOrderDemo

Module contents

1. Initialization sections
   - Class fields initialization
   - Nonstatic initialization block
   - Static initialization block
   - Order of Initialization block execution
   - Initialization of final variable

See finalvarsinitpkg.FinalVarsInitDemo