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. Exceptions and Assertions
11. Nested classes
12. Enums
13. Wrapper classes for primitive types
14. Generics
15. Collections
16. Method overload resolution
17. Multithreads
18. Core Java Classes
19. Object Oriented Design

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

Class fields initialization 2/5

```
1.  public class ClassInitDemo1 {
2.      private static boolean b;
3.      private static byte by;
4.      private static int i;
5.      private static double d;
6.      private static char c;
7.      private static String st;
8.      private static int[] arr;
9.      public static void main(String[] arg){
10.         System.out.println("boolean: "+b);
11.         System.out.println("byte : "+by);
12.         //...
13.         System.out.println("String : "+st);
14.         System.out.println("Array: "+arr);
15.     }
16. }
```

Console output:

```
boolean : false
byte : 0
int : 0
double : 0.0
char :
String : null
Array: null
```

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. }
```

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:

```
1.  public class ClassInitDemo1 {
2.      //...
3.      private static byte by = 127;
4.      private static int i = 234*by;
5.      //...
6.      public static void main(String[] arg) {
7.          System.out.println("int : " + i);
8.      }
9.  }
```

Class fields initialization 5/5

- You can call static methods for fields initialization :

```
1.  public class ClassInitDemo1 {
2.      //...
3.      private static int i = doInit();
4.      //...
5.      private static int doInit() {
6.          System.out.println("Init i value:");
7.          return 123;
8.      }
9.      //...
10.     public static void main(String[] arg) {
11.         System.out.println("Start main");
12.         System.out.println("int : " + i);
13.     }
14. }
```

Console output:

Init i value:
Start main
int : 123

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:

```
1.  public class ClassInitDemo1 {
2.      //...
3.      private int i;
4.      private String str;
5.      //...
6.      {
7.          i= 12345;
8.          str = "Hi!";
9.      }
10.     public static void main(String[] arg) {
11.         ClassInitDemo1 cdm = new ClassInitDemo1();
12.         System.out.println("int : " + cdm.i);
13.         System.out.println("String: " + cdm.str);
14.     }
15. }
```

Non-static initialization block 2/3

```
1.  public class Car {
2.      private static int numOfCars;
3.      //...
4.      public Car() {
5.          //...
6.          numOfCars++;
7.      }
8.      public Car(int maxSpeed) {
9.          //...
10.         numOfCars++;
11.     }
12. }
```

Non-static initialization block 3/3

```
1.  public class Car {
2.      private static int numOfCars;
3.      //...
4.      {
5.          numOfCars++;
6.      }
7.      public Car() {
8.          //...
9.      }
10.     public Car(int maxSpeed) {
11.         //...
12.     }
13. }
```

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:

```
1.  public class ClassInitDemo1 {
2.      private static int x;
3.      //...
4.      static {
5.          x = 1234;
6.      }
7.      public static void main(String[] arg) {
8.          System.out.print(x);
9.      }
10. }
```

Static initialization block 2/2

```
1.  public class ClassInitDemo1 {
2.      //...
3.      private static char[] alph;
4.      //...
5.      static {
6.          alph = new char[26];
7.          int i = 0;
8.          for (char c = 'a'; i < alph.length; c++,i++) {
9.              alph[i] = c;
10.         }
11.     }
12.     public static void main(String[] arg) {
13.         System.out.print(Arrays.toString(alph));
14.     }
15. }
```

Module contents

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

Order of Initialization block execution 1/3

```
1.  public class ClassInitDemo1 {
2.      public ClassInitDemo1() {
3.          System.out.println("constructor");
4.      }
5.      { // non-static
6.          System.out.println("non-static block");
7.      }
8.      static { // static
9.        System.out.println("static block");
10.     }
11.     public static void main(String[] arg) {
12.         System.out.println("main");
13.         ClassInitDemo1 cdm = new ClassInitDemo1();
14.     }
15. }
```

Console output:

static block
main
non-static block
constructor

Order of Initialization block execution 2/3

```
1.  public class ClassInitDemo1 {
2.  private static int x = 100;
3.      static { // static 1
4.          x = 1;
5.          System.out.println("static block 1");
6.      }
7.      static { // static 2
8.          x = 2;
9.          System.out.println("static block 2");
10.     }
11.     public static void main(String[] arg) {
12.         System.out.println("main");
13.         System.out.println("x="+x);
14.     }
15. }
```

Console output:

```
static block 1
static block 2
main
x=2
```

Order of Initialization block execution 3/3

```
1.  public class ClassInitDemo1 {
2.      private int y = 100;
3.      {   y = 1;// non-static 1
4.          System.out.println("non-static block 1");
5.      }
6.      public ClassInitDemo1() {
7.          y = 3;
8.          System.out.println("constructor");
9.      }
10.     {   y = 2;// non-static 2
11.         System.out.println("non-static block 2");
12.     }
13.     public static void main(String[] arg) {
14.         System.out.println("main");
15.         ClassInitDemo1 cdm = new ClassInitDemo1();
16.         System.out.println("y=" + cdm.y);
17.     }
18. }
```

Console output:

main
non-static block 1
non-static block 2
constructor
y=3

Module contents

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

Initialization of final variable 1/3

```
1.  public class ClassInitDemo1 {
2.      private final int X = 100;
3.      private final int Z;
4.      private final int W;
5.      {
6.          Z = 200;
7.          System.out.println("non-static block");
8.      }
9.      public ClassInitDemo1() {
10.         W = 300;
11.         System.out.println("constructor");
12.     }
13.     public static void main(String[] arg) {
14.         System.out.println("main");
15.         ClassInitDemo1 cdm = new ClassInitDemo1();
16.         //...
17.     }
18. }
```

Initialization of final variable 2/3

```
1.  public class ClassInitDemo1 {
2.      private final int Z;
3.      public ClassInitDemo1() {
4.          Z = 300;
5.          System.out.println("constructor  1");
6.      }
7.      public ClassInitDemo1(int z) {
8.          Z = z;
9.          System.out.println("constructor 2");
10.     }
11.     public static void main(String[] arg) {
12.         System.out.println("main");
13.         ClassInitDemo1 cdm = new ClassInitDemo1();
14.         //...
15.     }
16. }
```

Initialization of final variable 3/3

```
1.  public class ClassInitDemo1 {
2.      private final int Z = 100;
3.      {
4.          Z = 200;
5.          System.out.println("non-static block");
6.      }
7.      public ClassInitDemo1() {
8.          Z = 300;
9.          System.out.println("constructor");
10.     }
11.     public static void main(String[] arg) {
12.         System.out.println("main");
13.         ClassInitDemo1 cdm = new ClassInitDemo1();
14.         //...
15.     }
16. }
```