JAVA 17

BOOK: LEARN JAVA 17 PROGRAMMING
SECOND EDITION
NICK SAMOYLOV
BIZ APPS

CORE MODELS (“SINCE 9/2022”)

WEB APP DEVELOPER

Robert Kigobe 12-13-2023

Contents

- Overview of Java Programming: 12/13/2023

- Strings
- The enhanced switch statement

String literals

- Strings are anything enclosed in “”, ‘’ e.g “abc”,’abc’ or the keyword null
- String literals are all stored in the string pool
- String objects are not stored in the string pool and will need an intern() to join the pool
- Java 15 introduced a new String literal called a text block
- Until Java 8, Strings were internally represented as an array of characters – char[], encoded in UTF-16, so that every character uses two bytes of memory.
- With Java 9 a new representation is provided, called Compact Strings. This new format chooses the appropriate encoding between char[] and byte[] depending on the stored content. The amount of heap memory will be significantly lower, which in turn causes less Garbage Collector overhead on the JVM.

Strings code - requires ide formatting

```
public class StringClass {
    public static void main(String[] args) {
        //String literals java 14
        String s = "abc";        String t = "abc";        System.out.println("=========>String literals");
        System.out.println(s==t);         System.out.println("abc" == t);

        //String objects
        String s1 = new String("abc");        String t1 = new String("abc");


        System.out.println("\n\n=========>String objects");        System.out.println(s1==t1 );        System.out.println("abc" == t1);
    System.out.println("abc" == t1.intern());        System.out.println(t1 == "abc");

        //Text blocks
        //Old version
        String html =    "<html>\n"+
                         "    <body>\n"+
                         "        <p>Hello world.</p>\n" +
                         "    </body>\n" +
                         "</html>\n";

        //After Java 15
        String html1 = """
                 <html>
                   <body>
                       <p>Hello world.
                   </body>
                 </html>
                 """;

        System.out.println("\n\n=========>String text blocks");        System.out.println(html1);        System.out.println(html);
        //String Imutability
        String str1 = "abc";        String r1 = str1;        str1 = str1 + "def";        String r2 = str1;
        System.out.println("\n\n=========>String immutability");        System.out.println(r1 == r2);        System.out.println(r1.equals(r2));

    }}
```

The switch statement

Traditional Switch Statement vs. Enhanced Switch Statement

Traditional Switch Statement	Enhanced Switch Statement

```
switch (switchValue) {
    case 1:
        System.out.println("Value was 1");
        break;
    case 2:
        System.out.println("Value was 2");
        break;
    case 3: case 4: case 5:
        System.out.println("Was a 3, a 4, or a 5");
        System.out.println("Actually it was a " + switchValue);
        break;
    default:
        System.out.println("Was not 1, 2, 3, 4, or 5");
        break;
}
```

```
switch (switchValue) {
    case 1 -> System.out.println("Value was 1");
    case 2 -> System.out.println("Value was 2");
    case 3, 4, 5 -> {
        System.out.println("Was a 3, a 4, or a 5");
        System.out.println("Actually it was a " + switchValue);
    }
    default -> System.out.println("Was not 1, 2, 3, 4, or 5");
}
```

The switch statement

- The enhanced statement has no break.
- The enhanced switch statement allows expressions to be evaluated in the case blocks.
- If the enhanced statement is being called to return a value, yield is used to return the value needed and not return.

Enhanced switch code - requires an ide formatting

```
public class Switch {

    public static void main(String[] args) {

        int value = 3;          System.out.println("Traditional if statement");
        if (value == 1) {             System.out.println("Value was 1");
        } else if (value == 2) {            System.out.println("Value was 2");
        } else {            System.out.println("Value was not 1 or 2");
        }

        //Traditional switch statement
        //Only byte, short, int, char, String, enum
        //fall through happens when a break is not encountered executing all ines till a break is found
        int valueSwitch = 8;
        System.out.println("\n\nTraditional switch");
        switch (valueSwitch) {            case 1:                 System.out.println("Switch Value was 1");                 break;
            case 2:                 System.out.println("Switch Value was 2");                 break;

            case 3:             case 4:             case 5:

                System.out.println("Switch Value was a 3, a 4, or a 5");                System.out.println("Switch Value was
actually a " + valueSwitch);                 break;

            default:                System.out.println("Switch Value was not 1, 2, 3, 4, 5");                 break;         }

        //new switch features
        switch (valueSwitch) {            case 1 -> System.out.println("Switch Value was 1");
            case 2 -> System.out.println("Switch Value was 2");             case 3, 4, 5 -> {
                System.out.println("\n\nModern switch");                System.out.println("Switch Value was a 3, a 4, or a 5");
                System.out.println("Switch Value was actually a " + valueSwitch);             }
            default -> System.out.println("Switch Value was not 1, 2, 3, 4, 5");         }     }
```