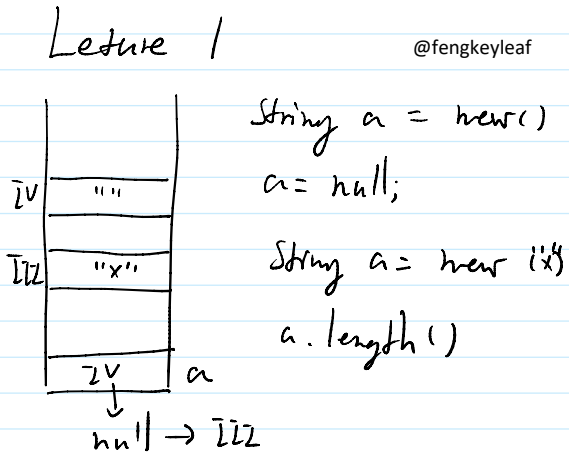
Week 2

Sunday, August 23, 2020 8:33 PM

Lecture 1 @fengkeyleaf



- https://docs.oracle.com/en/java/javase/14/docs/api/java.base/java/lang/String.html
- https://docs.oracle.com/javase/tutorial/essential/regex/literals.html
- Strings are constants
- All String literals are instances of the String class and exist once in a JVM.

```
1   class StringLiteral {
2
3    public static void main( String args[] ) {
4        String aString = "you";
5        String bString = "yo" + "u";    // compiler
6
7        if ( "you".equals(aString) )
8            System.out.println("1. equal");
9        if ( bString.equals(aString) )
10           System.out.println("2. equal");
11       if ( "yo" + "u" == aString )
12           System.out.println("3. ==");
13       if ( bString == aString )
14           System.out.println("4. ==");
15       if ( bString == new String("you") )
16           System.out.println("5. ==");
17       else
18           System.out.println("6. !=");
19
20    }
21   }
```

compare address

Result:

1. equal
2. equal
3. ==
4. ==
6. !=

The values of variable of type String are inmutable.

```
1   /**
2    * Deal with Strings objects.
3    *
4    * @version    $Id$
5    *
6    * @author     hp bischof
7    *
8    * Revisions:
9    *          $Log$
10   */
11
12  class StringThing
13  {
14    public static void main(String args[])
15    {
16      String aString;
17
18      aString = new String("Last Stop Wonderland! ");
19      System.out.println( aString.length() );
20      System.out.println( aString.toUpperCase() );
```

22

关于java中判断字符串相等==和equal 详解
https://blog.csdn.net/weixin_42350212/article/details/80768041

java中String的相等判断(==和equals())详解
https://blog.csdn.net/gotobar/article/details/43762523

Java 比较（==, equals, compareTo, compare)
https://www.jianshu.com/p/27c9a31b57e7

4.3.3. The Class String
https://docs.oracle.com/javase/specs/jls/se14/html/jls-4.html#jls-4.3.3

15.18.1. String Concatenation Operator +
https://docs.oracle.com/javase/specs/jls/se14/html/jls-15.html#jls-15.18.1

15.29. Constant Expressions
https://docs.oracle.com/javase/specs/jls/se14/html/jls-15.html#jls-15.29

3.10.5. String Literals
https://docs.oracle.com/javase/specs/jls/se11/html/jls-3.html#jls-3.10.5

Java集合之EnumSet
https://blog.csdn.net/moakun/article/details/80617845

Java 到底是值传递还是引用传递?
https://www.zhihu.com/question/31203609

Java基础—break label 带标签的break语句的用法
https://www.jianshu.com/p/3e12d2c9bcf3
https://www.cs.rit.edu/~hpb/Lectures/2201/605/605-80.html

Java Scanner 类
https://www.runoob.com/java/java-scanner-class..html

Java Scanner类的常用方法及用法（很详细）
https://blog.csdn.net/qq_40164190/article/details/81917208

```
21          System.out.println( aString.toUpperCase() + ".");
22          System.out.println( aString.length() + 1 );
23          System.out.println( 1 + aString.length() );
24          System.out.println( 1 + aString + 1 );
25          System.out.println( aString + ( 1 + 1 ) );
26       }
27    }
28
```

Result:

```
22
LAST STOP WONDERLAND!
LAST STOP WONDERLAND! .
23
23
1Last Stop Wonderland! 1
Last Stop Wonderland! 2
```

Other Example

```
1    /**
2     * "abc" versus new String("abc")`
3     */
4
5    class StringL {
6
7     public static void method(String id, String literal, String aNewString)      {
8           System.out.println(id + " in method");
9           System.out.print("\tliteral= aNewString\n      ");
10           System.out.println( literal == aNewString);
11     }
12     public static void main( String args[] ) {
13           String aString = "abc";
14           System.out.print("abc == aString\n      ");
15           System.out.println("abc" == aString);
16
17           String newString = new String("abc");
18           System.out.print("abc == new String(abc)\n      ");
19           System.out.println("abc" == newString);
20
21           method("1", "abc", "abc");
22           method("2", "abc", new String("abc") );
23           method("3", "abc", "ab" + "c");
24           method("4", "abc", "" + "abc");
25       }
26    }
```

```
1    public class X_s1 {
2
3       public static void method_a()       {
4           String one  = "1";
5           String ten  = "10";
6           do     {
7                 System.out.println("a_1: " + ( one == ten )  );
8                 one = one + "0";
9           } while ( one == ten );
10           System.out.println("a_2: " + ( one == ten )  );
11      }
12      public static void method_b()       {
13           int one  = 1;
14           int ten  = 10;
15
16           System.out.println("b_1: " + 3 * one + 0 * ten );
17      }
18      public static void method_c()       {
19           String right = new String(10 * 1 - 10 + "");
20           String middle  = "00".substring(0, 1);
21      {String left  = "0";
22      String leftLeft  = "0";
23           int    aInt  = 0;
24
25           System.out.println("c_1: " +     ( "0" == left )   );
26           System.out.println("c_2: " +     ( "0" == middle   ) );
27           System.out.println("c_3: " +     ( left  +  aInt + aInt ) );
28           System.out.println("c_4: " +     ( ( left  =  left + aInt + aInt ) ) + left );
29           System.out.println("c_5: " +     ( ( leftLeft = left = ( left  =  left + aInt + aInt ) ) ) + left );
30           System.out.println("c_6: " +     ( leftLeft ) );
31                // c_3: 000123
32                // c_4: 0000001234
33                // c_5: 0000000000
34                // c_6: 0000012345
35           System.out.println("c_7: " +     ( ( "1" + ""  ) == "1" ) );
36           System.out.println("c_8: " +     ( ( aInt  + "1" ) == ( left + "1" ) ) );
37      }
38      public static void main(String argv[]) {
39           method_c();
40           method_a();
41           method_b();
42      }
43   }
```

Execution:

```
% java X_s1
c_1: true
c_2: false
c_3: 000
c_4: 000000
c_5: 0000000000
c_6: 00000
c_7: true
c_8: false
a_1: false
a_2: false
b_1: 30
```

```
1   /**
2    * Play with the String class
3    *
4    * @version    $Id$
5    *
6    * @author     Hpb
7    *
8    *    $Log$
9    */
10
11  class String_1 {
12
13    public static void main( String args[] ) {
14        String aString = "David";
15        String bString = "David Bowie";
16
17        if ( "hello".equals("hello") )
18            System.out.println("equal");
19        if ( "David" == aString )
20            System.out.println("David == aString ");
21        System.out.println(aString.length());
22        System.out.println(aString.charAt(0));
23
24        System.out.println(aString.indexOf("vid"));
25
26        System.out.println(aString.substring(2,3));
27        System.out.println(aString.substring(
28                            aString.indexOf("a"),
29                            aString.indexOf("i")
30                                ));
31
32        System.out.println(aString.concat(" Bowie").length());
33
34        String help = bString.substring(0, aString.length());
35        System.out.println("-->" + help + "<--" );
36        if ( "David" == help )
37            System.out.println("David == help ");
38        if ( "David" == aString )
39            System.out.println("David == bString ");
40    }
41  }
```

Result:

```
equal
ava String_1
equal
David == aString
5
D
2
v
av
11
-->David<--
David == bString
```

A question was asked:

```
1   class Sq {
2
3     public static void main(String args[]) {
4        String aString = "0";
5        String bString = "0" + "1";
6        System.out.println(aString + "1" == "01" );
7        System.out.println(bString  == "01" );
8        System.out.println("0" + "1" == "01");
9
10    }
11  }
```

```
1
2   class StringAndInteger
3   {
4     public static void main(String args[])
5     {
6       System.out.println("Well, 3 + 4 = " + 7 );
7       System.out.println("Well, 3 + 4 = " + 3 + 4 );
8       System.out.println("Well, 3 + 4 = " + ( 3 + 4 ) );
9     }
10  }
11
```

"Well, 3+4 = 7"
"Well, 3+4 = 34"
"Well, 3+4 = 7"

```
1
2   class StringAndInteger2
3   {
4     public static void main(String args[])
5     {
6       System.out.println(3 + 7);
7       System.out.println(3 + 7 + "abc");
8       System.out.println(3 + 7 + "abc" + 1);
9       System.out.println(3 + 7 + "abc" + 1 + 2);
10      System.out.println("" + 3 + 7 + "abc" + 1 + 2);
11      System.out.println("" + ( 3 + 7 ) + "abc" + ( 1 + 2 ));
12    }
13  }
14
```

10
"10abc"
"10abc1"
"10abc12"
"37abc12"
"10abc3"

```
1   /**
2    * Play with the String class
3    *
4    * @version    $Id$
5    *
6    * @author     Hpb
7    *
8    *    $Log$
9    */
10
11  class StringUse {
12
13    public static void compare(String aString, String bString)    {
14        if ( aString.equals(bString) )
15            System.out.println("\tequal");
16        else
17            System.out.println("\t! equal");
18        if ( aString == bString)
19            System.out.println("\t== ");
20        else
21            System.out.println("\t! ==");
22
23    }
24    public static void main( String args[] ) {
25        String aString = "David";
26        String bString = "David";
27        compare(aString, bString);
28
29        System.out.println("Using New");
30        aString = new String("David");
31        bString = new String("David");
32        compare(aString, bString);
33
34        System.out.println("Concatenation 1");
35        aString = "Da" + "vid";
36        bString = "" + "David";
37        compare(aString, bString);
38
39        System.out.println("Concatenation 2");
40        aString = "Da" + "vid";
41        bString = "D" + "a" + "vid";
42        compare(aString, bString);
43
44    }
45  }
```

same
"equal" "=="
"equal" "!="
"equal" "=="
"equal" "=="

Execution:

```
% java StringUse
equal
  ==
Using New
  equal
  ! ==
Concatenation 1
  equal
  ==
Concatenation 2
  equal
  ==
```

```
/**
 * Use of this!
 */

class UseOfThis
{
  int id;
  UseOfThis(int id) {
      this.id = id;
  }
  private void method_2()
  {
      System.out.println("method_2: " + this);
  }

  private void method_1()
  {
      System.out.println("method_1: " + this);
      this.method_2();
      method_2();
  }
  public String toString()    {
      return "" + id;
  }

  public static void main(String args[])
  {
    UseOfThis aUseOfThis = new UseOfThis(1);
    UseOfThis bUseOfThis = new UseOfThis(2);

    System.out.println(aUseOfThis);
    System.out.println(bUseOfThis);

    aUseOfThis.method_1();
    bUseOfThis.method_1();
  }
}
```

Bitmap Solution for the problem to identify a few different cases:

```
int one   = 1;  // 0001
int three = 2;  // 0011
int bit   = 2   // 0010
if ( one == ( one & bit ) )
      //  0001 &&
      //  0010
      // ==   0000 != 0001
  ...
if ( three == ( three & bit ) )
      //  0011 &&
      //  0001
      // ==   0000 == 0010
```

Enum Solution for the problem to identify a few different cases

```
import java.lang.Enum;

public class EnumExample       {
enum TheColors {
        RED, GREEN, BLUE
 };
  public static void whatTheColors(TheColors theTheColors)     {
      if ( theTheColors == TheColors.RED )
            System.out.println("Roses are red");
        else    // can this happen?
             System.out.println("Unkonw color: " + theTheColors);
  }
  public static void main( String args[] ) {
      whatTheColors(TheColors.RED);
  }
}
```

From the documentation: "The space and time performance of this class should be good enough to allow its use as a high-quality, type safe alternative to traditional int-based "bit flags." Even bulk operations (such as containsAll and retainAll) should run very quickly if their argument is also an enum set." More later: Collection framework.

```
import java.util.EnumSet;
import java.util.Set;

public class EnumSetExample    {
enum Color {
        RED, GREEN, BLUE
 };
  public static void main( String args[] ) {
      EnumSet<Color> redGreen = EnumSet.of(Color.RED, Color.GREEN);
       EnumSet<Color> complementOf = EnumSet.complementOf(redGreen);
       System.out.println("redGreen       = " + redGreen );
```

{RED, GREEN}
{BLUE}

```
12              System.out.println("complementOf   = " + complementOf );
13      }
14  }
```

Result:

```
redGreen       = [RED, GREEN]
complementOf   = [BLUE]
```


The conditional operator may be used to choose between second and third operands of numeric type, or second and third operands of type boolean, or second and third operands that are each of either reference type or the null type. All other cases result in a compile-time error.