
Introduction to java

# COMP1030
# Lecture #9

# Housekeeping

- Our goal - 100% pass rate in this course.
- Review the lecture slides ahead of time.
- Review the lecture slides after class with a study group.
- Repeat the lab at home 1-2 times.
- Take notes for each lecture.
- Attend live stream tutorials every Monday night at 7pm via webex.


# Housekeeping

- Assignment #1 Answer key is posted under the assignments link
- Midterm-Exam question paper and answer key are posted under course information link
- Labs will no longer be checked as homework

# Review

- Exception handling
- Multidimensional arrays

# toString Method

- toString – all objects in java have a toString method that returns a String representation of the object.
- The toString method is implicitly called whenever a String representation is needed (for example for concatenation)

# toString Method

- Any objects you create have a toString method.
- Where does this method come from?

# Big "O" Object

- Every class in java is a child of (subclass) of the Object (Big O object) class.

java.io

**Class FileReader**

java.lang.Object
    java.io.Reader
        java.io.InputStreamReader
            java.io.FileReader

# Big "O" Object

java.lang

## Class Object

java.lang.Object

---

`public class` **`Object`**

Class `Object` is the root of the class hierarchy. Every class has `Object` as a superclass. All objects, including arrays, implement the methods of this class.

# Big "O" Object

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| protected Object | **clone**()<br>Creates and returns a copy of this object. |
| boolean | **equals**(Object obj)<br>Indicates whether some other object is "equal to" this one. |
| protected void | **finalize**()<br>Called by the garbage collector on an object when garbage collection determines that there are no more references to the object. |
| Class<?> | **getClass**()<br>Returns the runtime class of this Object. |
| int | **hashCode**()<br>Returns a hash code value for the object. |
| void | **notify**()<br>Wakes up a single thread that is waiting on this object's monitor. |
| void | **notifyAll**()<br>Wakes up all threads that are waiting on this object's monitor. |
| String | **toString**()<br>Returns a string representation of the object. |
| void | **wait**()<br>Causes the current thread to wait until another thread invokes the **notify()** method or the **notifyAll()** method for this object. |
| void | **wait**(long timeout)<br>Causes the current thread to wait until either another thread invokes the **notify()** method or the **notifyAll()** method for this object, or a specified amount of time has elapsed. |
| void | **wait**(long timeout, int nanos)<br>Causes the current thread to wait until another thread invokes the **notify()** method or the **notifyAll()** method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed. |

java

# System.out.println

java.lang

## Class System

java.lang.Object
    java.lang.System

### Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `static PrintStream` | `err`<br>The "standard" error output stream. |
| `static InputStream` | `in`<br>The "standard" input stream. |
| `static PrintStream` | `out`<br>The "standard" output stream. |

### out

```
public static final PrintStream out
```

The "standard" output stream. This stream is already open and ready to accept output data.

java.io

## Class PrintStream

java.lang.Object
    java.io.OutputStream
        java.io.FilterOutputStream
            java.io.PrintStream

**All Implemented Interfaces:**

Closeable, Flushable, Appendable, AutoCloseable

Java

# PrintStream Behaviour

| | |
|---|---|
| void | println()<br>Terminates the current line by writing the line separator string. |
| void | println(boolean x)<br>Prints a boolean and then terminate the line. |
| void | println(char x)<br>Prints a character and then terminate the line. |
| void | println(char[] x)<br>Prints an array of characters and then terminate the line. |
| void | println(double x)<br>Prints a double and then terminate the line. |
| void | println(float x)<br>Prints a float and then terminate the line. |
| void | println(int x)<br>Prints an integer and then terminate the line. |
| void | println(long x)<br>Prints a long and then terminate the line. |
| void | println(Object x)<br>Prints an Object and then terminate the line. |
| void | println(String x)<br>Prints a String and then terminate the line. |

# String class

- A String is a sequence of characters.
- A String can include letters, digits and various special characters.

# String class Constructors (partial list)

- The String class has several constructors:

**Constructors**

| Constructor and Description |
| --- |
| `String()`<br>Initializes a newly created `String` object so that it represents an empty character sequence. |
| `String(byte[] bytes)`<br>Constructs a new `String` by decoding the specified array of bytes using the platform's default charset. |
| `String(byte[] bytes, Charset charset)`<br>Constructs a new `String` by decoding the specified array of bytes using the specified charset. |
| `String(byte[] ascii, int hibyte)`<br>**Deprecated.**<br>*This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the `String` constructors that take a `Charset`, charset name, or that use the platform's default charset.* |
| `String(byte[] bytes, int offset, int length)`<br>Constructs a new `String` by decoding the specified subarray of bytes using the platform's default charset. |
| `String(byte[] bytes, int offset, int length, Charset charset)`<br>Constructs a new `String` by decoding the specified subarray of bytes using the specified charset. |
| `String(byte[] ascii, int hibyte, int offset, int count)`<br>**Deprecated.**<br>*This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the `String` constructors that take a `Charset`, charset name, or that use the platform's default charset.* |
| `String(byte[] bytes, int offset, int length, String charsetName)`<br>Constructs a new `String` by decoding the specified subarray of bytes using the specified charset. |

# String class Constructors

```
1  // Fig. 14.1: StringConstructors.java
2  // String class constructors.
3
4  public class StringConstructors
5  {
6     public static void main(String[] args)
7     {
8        char[] charArray = {'b', 'i', 'r', 't', 'h', ' ', 'd', 'a', 'y'};
9        String s = new String("hello");
10
11       // use String constructors
12       String s1 = new String();
13       String s2 = new String(s);
14       String s3 = new String(charArray);
15       String s4 = new String(charArray, 6, 3);
16
17       System.out.printf(
18          "s1 = %s%ns2 = %s%ns3 = %s%ns4 = %s%n", s1, s2, s3, s4);
19    }
20 } // end class StringConstructors
```

```
s1 =
s2 = hello
s3 = birth day
s4 = day
```

**Fig. 14.1** | String class constructors.

# String class methods

```
1  // Fig. 14.2: StringMiscellaneous.java
2  // This application demonstrates the length, charAt and getChars
3  // methods of the String class.
4
5  public class StringMiscellaneous
6  {
7     public static void main(String[] args)
8     {
9        String s1 = "hello there";
10       char[] charArray = new char[5];
11
12       System.out.printf("s1: %s", s1);
13
14       // test length method
15       System.out.printf("%nLength of s1: %d", s1.length());
16
17       // loop through characters in s1 with charAt and display reversed
18       System.out.printf("%nThe string reversed is: ");
19
20       for (int count = s1.length() - 1; count >= 0; count--)
21          System.out.printf("%c ", s1.charAt(count));
22
23       // copy characters from string into charArray
24       s1.getChars(0, 5, charArray, 0);
25       System.out.printf("%nThe character array is: ");
26
27       for (char character : charArray)
28          System.out.print(character);
29
30       System.out.println();
31    }
32 } // end class StringMiscellaneous
```

```
s1: hello there
Length of s1: 11
The string reversed is: e r e h t   o l l e h
The character array is: hello
```

**Fig. 14.2** | `String` methods `length`, `charAt` and `getChars`. (Part 2 of 2.)

# String class – indexOf

```
public class IndexOfExample{
   public static void main(String args[]) {
       String str1 = new String("This is a BeginnersBook tutorial");
       String str2 = new String("Beginners");
       String str3 = new String("Book");
       String str4 = new String("Books");
       System.out.println("Index of B in str1: "+str1.indexOf('B'));
       System.out.println("Index of B in str1 after 15th char:"+str1.indexOf('B', 15));
       System.out.println("Index of B in str1 after 30th char:"+str1.indexOf('B', 30));
       System.out.println("Index of string str2 in str1:"+str1.indexOf(str2));
       System.out.println("Index of str2 after 15th char"+str1.indexOf(str2, 15));
       System.out.println("Index of string str3:"+str1.indexOf(str3));
       System.out.println("Index of string str4"+str1.indexOf(str4));
       System.out.println("Index of harcoded string:"+str1.indexOf("is"));
       System.out.println("Index of hardcoded string after 4th char:"+str1.indexOf("is", 4));
   }
}
```

Output

```
Index of B in str1: 10
Index of B in str1 after 15th char:19
Index of B in str1 after 30th char:-1
Index of string str2 in str1:10
Index of str2 after 15th char-1
Index of string str3:19
Index of string str4-1
Index of harcoded string:2
Index of hardcoded string after 4th char:5
```

Retreived from: https://beginnersbook.com/2013/12/java-string-indexof-method-example/

# String class – substring

```
1  // Fig. 14.6: SubString.java
2  // String class substring methods.
3
4  public class SubString
5  {
6     public static void main(String[] args)
7     {
8        String letters = "abcdefghijklmabcdefghijklm";
9
10       // test substring methods
11       System.out.printf("Substring from index 20 to end is \"%s\"%n",
12          letters.substring(20));
13       System.out.printf("%s \"%s\"%n",
14          "Substring from index 3 up to, but not including 6 is",
15          letters.substring(3, 6));
16    }
17 } // end class SubString
```

```
Substring from index 20 to end is "hijklm"
Substring from index 3 up to, but not including 6 is "def"
```

**Fig. 14.6** | String class substring methods.

# String class – concatenating strings

```
1  // Fig. 14.7: StringConcatenation.java
2  // String method concat.
3
4  public class StringConcatenation
5  {
6     public static void main(String[] args)
7     {
8        String s1 = "Happy ";
9        String s2 = "Birthday";
10
11       System.out.printf("s1 = %s%ns2 = %s%n%n",s1, s2);
12       System.out.printf(
13          "Result of s1.concat(s2) = %s%n", s1.concat(s2));
14       System.out.printf("s1 after concatenation = %s%n", s1);
15    }
16 } // end class StringConcatenation
```

```
s1 = Happy
s2 = Birthday

Result of s1.concat(s2) = Happy Birthday
s1 after concatenation = Happy
```

**Fig. 14.7** | String method concat.

# String class – misc. methods

```
1  // Fig. 14.8: StringMiscellaneous2.java
2  // String methods replace, toLowerCase, toUpperCase, trim and toCharArray.
3
4  public class StringMiscellaneous2
5  {
6     public static void main(String[] args)
7     {
8        String s1 = "hello";
9        String s2 = "GOODBYE";
10       String s3 = "   spaces   ";
11
12       System.out.printf("s1 = %s%ns2 = %s%ns3 = %s%n%n", s1, s2, s3);
13
14       // test method replace
15       System.out.printf(
16          "Replace 'l' with 'L' in s1: %s%n%n", s1.replace('l', 'L'));
17
18       // test toLowerCase and toUpperCase
19       System.out.printf("s1.toUpperCase() = %s%n", s1.toUpperCase());
20       System.out.printf("s2.toLowerCase() = %s%n%n", s2.toLowerCase());
21
22       // test trim method
23       System.out.printf("s3 after trim = \"%s\"%n%n", s3.trim());
24
25       // test toCharArray method
26       char[] charArray = s1.toCharArray();
27       System.out.print("s1 as a character array = ");
28
29       for (char character : charArray)
30          System.out.print(character);
31
32       System.out.println();
33    }
34 } // end class StringMiscellaneous2
```

Output

```
s1 = hello
s2 = GOODBYE
s3 =    spaces

Replace 'l' with 'L' in s1: heLLo

s1.toUpperCase() = HELLO
s2.toLowerCase() = goodbye

s3 after trim = "spaces"

s1 as a character array = hello
```