



Introduction to java

# COMP1030

## Lecture #2



# 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 during class.



# Review

- Advantages of java over C++
- Java vs javascript
- OO concepts
- JVM
- javadoc
- PATH variable
- javac.exe (JDK)
- Java.exe (JRE)



# Java comments

Java supports 3 styles of commenting

// for comments on one line

/\* for comments on  
multiple lines \*/

/\*\* javadoc comments \*/



# Java comments

## Course comment standards

- A meaningful comment for each code section
- Every program to begin with:

```
/** Application Purpose:
```

```
*   Author:
```

```
*   Date:
```

```
*   Time:
```

```
*/
```



# Semicolons, blocks and whitespace

- A statement is single line of code terminated by a ;

Total = a + b + c;



- A block is a collection of statements bounded by opening and closing braces. Blocks can be nested.

```
{  
  code here  
}
```

Notice the tabbed  
whitespace – this is  
convention not syntax



# Whitespace

- Whitespace is ignored by the compiler, therefore any amount of whitespace is allowed.
- Use whitespace to enhance program readability



# Identifiers

- Identifiers are names given to variables (state) , classes (blueprints for objects) and methods (behaviour)
- An identifier is a series of characters consisting of letters, digits, underscores and \$.
- Identifiers cannot begin with a digit, cannot be an underscore alone, or cannot contain whitespace





# Java keywords

abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	void
volatile	while		



# Java types

- Logical      boolean
- Textual      char, String
- Integral      byte, short, int, long
- Floating      double, float



# Java types

Type	Contains	Default	Size	Range
boolean	true or false	false	1 bit	NA
char	Unicode character	\u0000	16 bits	\u0000 to \uFFFF
byte	Signed integer	0	8 bits	-128 to 127
short	Signed integer	0	16 bits	-32768 to 32767
int	Signed integer	0	32 bits	-2147483648 to 2147483647
long	Signed integer	0	64 bits	-9223372036854775808 to 9223372036854775807
float	IEEE 754 floating point	0.0	32 bits	$\pm 1.4\text{E-}45$ to $\pm 3.4028235\text{E}+38$
double	IEEE 754 floating point	0.0	64 bits	$\pm 4.9\text{E-}324$ to $\pm 1.7976931348623157\text{E}+308$



# Java String class

- String is one of the few classes which supports “implicit” instantiation
- The java String class is part of the java.lang package
- Strings are immutable
- A String has its literal enclosed in double quotes

Example:

```
String errorMsg = "Record Not Found"
```



# Java String methods

---

```
public class String
```

---

```
    String(String s)
```

*create a string with the same value as s*

```
    int length()
```

*number of characters*

```
    char charAt(int i)
```

*the character at index i*

```
    String substring(int i, int j)
```

*characters at indices i through (j-1)*

```
    boolean contains(String substring)
```

*does this string contain substring?*

```
    boolean startsWith(String pre)
```

*does this string start with pre?*

```
    boolean endsWith(String post)
```

*does this string end with post?*

```
    int indexOf(String pattern)
```

*index of first occurrence of pattern*

```
    int indexOf(String pattern, int i)
```

*index of first occurrence of pattern after i*

```
    String concat(String t)
```

*this string with t appended*

```
    int compareTo(String t)
```

*string comparison*

```
    String toLowerCase()
```

*this string, with lowercase letters*

```
    String toUpperCase()
```

*this string, with uppercase letters*

```
    String replaceAll(String a, String b)
```

*this string, with as replaced by bs*

```
    String[] split(String delimiter)
```

*strings between occurrences of delimiter*



# Java Arithmetic Operators

## Arithmetic Operators

A = 10, B = 20

Operator	Description	Example
+	Addition	A + B will give 30
-	Subtraction	A - B will give -10
*	Multiplication	A * B will give 200
/	Division	B / A will give 2
%	Modulus	B % A will give 0
++	Increment	B++ gives 21
--	Decrement	B-- gives 19



# Java Programming Conventions

Name	Convention
Class Name	Should start with uppercase letter and be a noun e.g. <b>S</b> tring, <b>C</b> olor, <b>B</b> utton, <b>S</b> ystem, <b>T</b> hread, <b>A</b> rray <b>L</b> ist, <b>S</b> tring <b>B</b> uffer etc.
Interface Name	Should start with uppercase letter and be an adjective e.g. <b>R</b> unnable, <b>R</b> emote, <b>L</b> ist, <b>A</b> ction <b>L</b> istener etc.
Method Name	Should start with lowercase letter and be a verb e.g. <b>a</b> ction <b>P</b> erformed(), <b>m</b> ain(), <b>p</b> rint(), <b>p</b> rintln() etc.
Variable Name	Should start with lowercase letter e.g. <b>f</b> irst <b>N</b> ame, <b>l</b> ast <b>N</b> ame, <b>o</b> rd <b>N</b> umber etc.
Package Name	Should be in lowercase letter e.g. java.lang, java.sql, java.util etc.
Constants Name	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.



# Java import statement

- The import statement provides a mechanism to make specific functionality (classes) available to your program code.
- The import statement is required to use any class that is not automatically imported by your program code.
- By convention the **import statement** is written directly before the class definition.





# Java import statement

- The only exception to this rule are classes that exist within the `java.lang` package. These classes are automatically available to the compiler without importing them.
- However, the compiler will not complain if you import them anyway.



# Class declaration

- To declare a class in java you simply use the keywords `public` (most often) and `class` and the name of the class using the appropriate naming convention. All code within the class is indented.

```
public class Employee
{
    code...
    code...
}
```



# The main method

- From lab #1 we have seen that in order to execute a java program we need to execute the runtime environment by typing:  
java ClassName
- Gosling designed java such that when you run “java” it will only look for one thing – a method called main – which, generally speaking, must look like this:

```
public static void main (String [] args)
```



# Variable Assignment

- To assign a value to a variable you need three things:
  - An identifier
  - The assignment operator
  - A value to assign (literal or another variable)
  - A Type

```
int age = 32;
```



# Displaying information

- To display information on the screen we use a built in java function called println.
- `System.out.println("hello");`
- `System.out.println(age);`



# The + operator

- The "+" operator is context sensitive, based upon the operands on either side.
- $3 + 5$  means addition: 8
- "3" + "5" mean concatenation: "35"
- `System.out.println("hello" + "world")` will display "helloworld" on the screen.






# charAt

- The charAt() method is functionality that lives in the String object.
- It is used to determine the character at a specific location in a string.
- All Strings are zero indexed.



# charAt

 Execute    Share   Source File   STDIN	 Result
<pre>1 public class HelloWorld 2 { 3 4     public static void main(String []args) 5     { 6         String testString = "house"; 7         System.out.println(testString.charAt(3)); 8     } 9 }</pre>	<pre>\$javac HelloWorld.java \$java -Xmx128M -Xms16M HelloWorld s</pre>

