Introduction to Java

Objectives	1. Install Java and create a runnable Java program. 2. Use Console I/O in Java.

Preliminary: Install Java JDK and BlueJ

Install 3 items on your computer:

1. Java Development Kit (JDK). http://java.oracle.com and click on "Java SE" under "Downloads".

2. BlueJ Development Environment: http://bluej.org.

3. Java Documentation (for reference).
http://www.oracle.com/technetwork/java/javase/downloads/index.html scroll down the page to "Java SE 8 Documentation" and click "Download".

You can view the Java API docs online at `http://docs.oracle.com/javase/8/docs/api/` but it better to install them on your computer.

You can download everything here: **`https://goo.gl/Ct7nuU`**

Problem 1: Use BlueJ interactively

1. In BlueJ, create a new project named Lab1. (You must open a project to use the CodePad.)

2. Use the BlueJ "CodePad" area to enter Java commands and see the result.

For example (what you type is shown in **bold**):

```
3*4
   12  (int)
"Hello" + "SKE"
   "HelloSKE"  (String)
Math.sqrt(5)
   2.2360697749 (double)
// This will print on the console.  If you don't see the
// console window, use View -> Show Terminal to display it.
System.out.println("Hello, console");
System.out.println("Hello, " + "SKE");
```

3. What is the difference between System.out.println(...) and System.out.print(...) ?

Try this in CodePad:

```
// compare this output:
System.out.println("Hi"); System.out.println("there");
// and this output:
System.out.print("Hi");  System.out.print("there");
```

Problem 2: Greet a person

Inside the Lab1 project, create a class named **`Greeter`** (file will be Greeter.java) to greet a person. Double click to edit the class and enter this code:

```
/**
 * Greet a person.
 */
public class Greeter {
    /** the main method starts the program. */
    public static void main(String[] args) {
        System.out.println("Hello, human.");
    }
}
```

Compile and run the program. To run: right click on the Greeter icon and choose "`void main(String[] args)`".

Problem 3: Greet a Person by Name

This problem involves (a) creating a Scanner object using the Java "new" command, (b) using Scanner to read a line from the console, (c) define a String variable (who) to store the value returned by the Scanner.

Modify the Greeter class:

```
import java.util.Scanner;
/**
 * Greet a person by name.
 */
public class Greeter {
    /** the main method starts the program. */
    public static void main(String[] args) {
        // create a Scanner to read data from console (System.in)
        Scanner console = new Scanner(System.in);
        // Ask the user for his name.
        System.out.print("What is your name? ");
        // read person's name using the nextLine( ) method
        String who = console.nextLine( );
        System.out.println("Hello, " + who);
    }
}
```

Notice that to join two strings you use `"Hello, " + who`

Problem 4: Greet a Person, and describe his computer

The System class has a method named getProperty(). It returns properties from the environment.

Try this is BlueJ CodePad. There is no semi-colon (;) so the result is printed in CodePad.

```
  System.getProperty( "os.name" )
  System.getProperty( "java.version" )
  System.getProperty( "user.name" )
```

If you want to print on the console, then use (notice the semi-colons after each command):

```
  String os = System.getProperty( "os.name" );
  System.out.println("You are using " + os );
```

Modify the Greeter program so that it also prints (a) what operating system he is using, (b) the version of Java, (c) his login name.

Example output:

```
What is your name?   Donald Trump
Hello, Donald Trump
Your computer is running Windows XP
Java version 1.8.0_111
You are logged in as trump
```

What Properties Does Java Know?

If you want to see ***all*** the properties that Java know, type this command in CodePad:

```
  System.getProperties().list( System.out );
```

Problem 5: Use Methods

The **`main( )`** method in our program is getting long. Define two separate methods for the commands you already wrote. Move (Cut-and-Paste) the commands from main() into the methods.

Then call each method from the **`main( )`** method, like this:

```
public class Greeter {
    public static void greetPerson() {
        // put commands to greet the person here
    }

    public static void showProperties() {
        // put commands to print the Java version and OS name here
        // don't print all properties (too long)
    }

    public static void main(String[] args) {
        greetPerson( );
        showProperties();
    }
}
```

Optional: Verify that Java Byte Code runs "Everywhere"

Java byte code should run on *any* computer with a Java Virtual Machine (JVM). The JVM is included in both the Java Developer Kit (JDK) that you installed and the *much smaller* Java Runtime (JRE). The only requirement is that the version of the JVM must be compatible with the version of the compiled code. This means that after you compile your app, anyone should be able to run it... on any platform.

1. Copy your compiled code (**`Greeter.class`**) to a shared folder, such as Google Drive.

2. Find someone using an OS *different from yours*. For example, if you are using Windows then find a Mac runnig OSX or PC running Linux (the lab machines have Linux). Ger them to download your Greeter.class from the shared folder.

3. Run it. Does it run the same?

Optional: Use Dialog Boxes instead of Boring Console

To make the Greeter look nicer. use dialog boxes instead of console I/O. The class JOptionPane has static methods to show different kinds of dialogs. We will use 2 methods:

JOptionPane.showMessageDialog(parent, "message") - display a message in dialog box. Use **`null`** for the parent parameter, since there is no parent window.

String reply = JOptionPane.showInputDialog("prompt string") - display a message and return user's reply as a String. The dialog box has a textbox where user can type.

```
import javax.swing.JOptionPane;  // "javax", not "java"

public class Greeter {
    public static void greetPerson() {
        String name =
            JOptionPane.showInputDialog("What's your name?");
        JOptionPane.showMessageDialog(null, "Hello, "+name);
    }

    public static void showProperties() {
        // Show the Java version and OS name using JMessageDialog
        // to display more than one line, include \n in the string
        // such as "Your Java version is 1.0\nYour OS is Android"
    }

    public static void main(String[] args) {
        greetPerson( );
        showProperties();
    }
}
```