# *Programming I (420-B10-HR)*

# *Lab 4 - Using Eclipse, Math Methods and Introduction to Strings*

Date assigned: Tuesday, September 15, 2015

Date due: **Tuesday, September 15, 2015**

**Learning Objectives**

Upon successful completion of this lab exercise, the student will be able to:

1. use the Eclipse IDE (Integrated Development Environment) to:
   1. create an Eclipse Project;
   2. create a class in an Eclipse project;
   3. use the Eclipse editor;
   4. compile and execute a JAVA program using Eclipse;
   5. reformat a class automatically;
2. use the **nextLine()** method of the **Scanner** class to read an entire line from the keyboard;
3. use the methods in the **Math** class to do mathematical functions;
4. use class constants to improve class maintainability;
5. use the cast operator to convert from one data type to another;
6. create and use **String** objects;
7. retrieve the length of a string;
8. use the **valueOf()** method to convert a primitive data type to String;
9. concatenate a string;

**Methods Used:**

***String class methods:***

**int length ()**

**String valueOf (***primitive datatype***)**

**To be handed in:**

1. A copy of your ***username*\_B10\_L04\_Eclipse** folder should be zipped and uploaded to **Moodle**.
2. The **Lab 4 Review Quiz** should be completed in **Moodle**.

***Note:*** Five (5) per cent will be given for organization. In order to obtain the marks:

* All the files and folders must be correctly named.
* All the Java classes must be correctly formatted using the Format option of Eclipse.
* The completed lab folder must be zipped and uploaded to **Moodle**.

**To Start:**

1. Download and unzip the **B10\_L04\_Eclipse** folder from **Moodle** to your **H:\420-B10\Labs** folder. Rename it to ***username*\_B10\_L04\_Eclipse**.

# Starting Eclipse

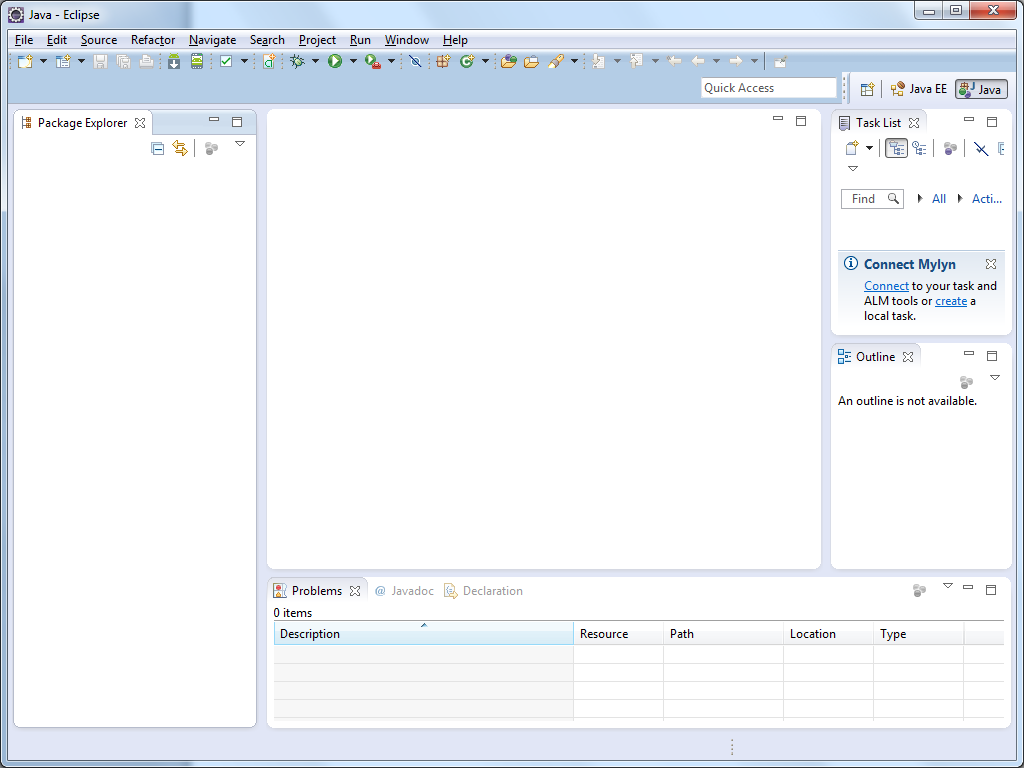
**Purpose**: To allow you to use the **Eclipse** Integrated Development Environment (IDE).

**To Do**:

## Select **Eclipse (****)**from the task bar.

## If you are asked which workspace to use, select **H:\420-B10\Labs** as your workspace. **DO NOT CLICK** the “Use this as the default and do not ask again” check box. Click **OK**.

## Click the **Workbench** symbol (the symbol to the far right of the Welcome screen.) The workbench window will open. Select **Open Perspective** from the **Window** menu and select **Java**. The workbench should look like:



The Class Pane

The Package Explorer

The Outline Pane

## Close the **Task List**. You won’t need it. You can also move the Outline Pane if you wish. I like to move it under the Package Explorer so that the Class Pane is wider.

If you accidentally close a frame you can open it again by selecting **Window** 🡪**Show View** 🡪 *the pane you wish to open*

## If you weren't asked to specify the workspace in step 2:

### Select **Switch Workspace** from the **File** menu.

### Click **Other** and Browse to your **H:\420-B10\Labs** folder.

### Click **OK**. Eclipse will restart.

# Creating a Project

***Purpose:*** To create a project in **Eclipse**.

Projects:

In order to create a Java program using Eclipse, you must first set up a project that will contain the program.

Generally a project is used to hold the files that make up an Eclipse application or applet. These files can be in any directory.

To Do:

## Select **File🡪New🡪Project…🡪Java Project** from the menu. Click **Next**.

## Type ***username*\_B10\_L04\_Eclipse** in the **Project Name** field.

## Click **Finish**.

Now we want to set up the formatting standards for use in all our classes.

## Right-click on the ***username*\_B10\_L04\_Eclipse** project name in the Package Explorer. Select **Properties**.

## Expand the ▷ beside **Java Code Style**. Click **Formatter**.

## Click **Configure Workspace Settings…**

## Click the **Import...** button below the **Active profile** drop-down box.

## Select *HeritageFormat.xml*.

## Close the **Properties** window.

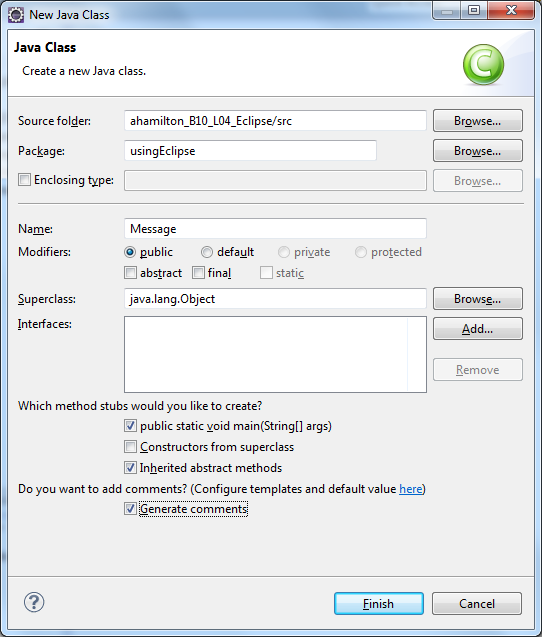
# Creating a New Class in a Project

Purpose: Learn how to create a new class in a project.

To Do:

## Select **New** from the **File** menu again and select **Class.**

## The name of the class we want to create is **Message**. We're going to create a package called **usingEclipse** for this class. We want a **main()** method in the class and we want to include Javadoc comments. Complete the dialog box for the New Java Class as shown here (using your username instead of sstark)



## The skeleton for the **Message** class with a **main()** method will now be in the editing window of **Eclipse**. Notice that Java keywords are in burgundy. Comments are in green. JavaDoc comments are in light blue. This will help you ensure that you have correctly typed keywords. User-defined identifiers are in black.

## Complete the comments before the class declaration as follows (Your username name will already be included. Replace it with your first and last names.):

**/\*\***

**\* Contains various output messages**

**\* @author *USERNAME***

**\*/**

## Add comments to the closing braces for the main() method ( *//main()* ) and the class ( *// Message class* ).

## Run your program. To do this, click the run icon () on the toolbar at the top or right click in the class and select **Run as** 🡪 **Java Application**. Click **OK** to the **Save and Launch** dialog. Nothing will happen because you have no executable code in your **main()** method.

# Reading a String

Purpose: Learn how to read a String that contains more than one word.

To Do:

## Replace the TODO comment in the **main()** method with the following lines:

String greeting;

greeting = "Hello ";

System.*out*.println(greeting);

## Format your program by right-clicking in the class and selecting **Source** 🡪 **Format**. This will result in your program being formatted so that it is easy to read. It is a good idea to do this frequently. It will make it easier to spot errors.

## Run your program again. This time the word "Hello " should be displayed in the **Console** panel at the bottom of the screen. (See figure on page 5.)

## Refer to your notes or to last week's lab and add a **Scanner** object called **keyboard** in the **main()** method. (Note, the import statement goes **after** the package statement.)

## Declare a new **String** variable called **name**.

## Prompt the user for his or her name.

## Use the **keyboard.next()** method to read the user's name.

## Change the **println()** statement to:

**System.*out*.println(greeting + name);**

## Run your program again. What is output this time?

## What is your name?

*Enter name*

Hello Philip

## Add a statement after the **println()** statement to change the greeting to "Goodbye " and copy the **println()** statement to follow the new assignment statement.

## Run the program again. Your output should look similar to this:

What is your name? Sandra

Hello Sandra

Goodbye Sandra

## Run your program again. Only this time enter your first and last name when prompted. What happens? It ignores my last name.

***Explanation***: The **next()** method stops reading when it encounters whitespace. Whitespace characters are blanks and line breaks. So when you used next() to read your name, it stopped at the space between your first and last names. To read everything on a line, use the **nextLine()** method.

## Modify the program to use **nextLine()** instead of **next()** to read your name. Run the program again and enter your full name. This time it should work.

# Exploring Eclipse

Purpose: Learn how to quickly manage classes using the Package Explorer and the Outline panel.

To Do:

The **Project Explorer** in **Eclipse** allows you to explore and manage your projects and classes.

## Expand your project as shown to display your packages and classes in the **Package Explorer**. You should have two packages – **usingEclipse** and **threeRs\_II**. The **threeRs\_II** package should contain three classes – **L04\_Constants**, **L04\_Math\_class** and **L04\_Numeric\_Data\_Types**.

**The Package Explorer:**

## You can double-click on the class name in the **Package Explorer** to open the class. If you right-click on the class name you can see the options available for the class. Double click on **L04\_Numeric\_Data\_Types.java** to open it.

**The Outline Pane:**

## To go directly to any method or instance variable definition, click on the name in the **Outline Pane**. Try it by clicking **main(String[])**.

# Java's Numeric Types

***Purpose*:** Learn how to use the different Java numeric data types.

***To Do*:**

## Change the author name for **L04\_Numeric\_Data\_Types.java** to yourself.

## What is 2147483647 + 1? 2147483648

## Run **L04\_Numeric\_Data\_Types**. What answer does the program give for the above question? -2147483648

Why? Because the answer is set as an integer.

Change the data type of i **long**. What is the answer to the addition now? 2147483648

**Overflow**: A number that is too large to store in a given data type.

## Add a new **int** variable called **j** to the **main()** method. Initialize it to **Integer.MIN\_VALUE**.

## Add a **println()** statement to output the words "The smallest int value is " followed by the value of **j**.

## Add another **println()** statement similar to the one that adds 1 to **i**. Only this time subtract 1 from **j**.

## Run the program.

What is the minimum value for an integer? -2147483647

What does the minimum value minus 1 calculate to in Java? -2147483647

**Underflow**: A number that is too small to store in a given data type.

# Using Math class methods

***Purpose*:** Learn how to use the Java Math class methods.

***To Do:***

## Open and run **L04\_Math\_class**.

## Change the author to yourself.

## Add **printlns** to calculate and print the floor and square root of x.

## Run the program several times with different values for **j** and **x**.

# Using Constants in Java

***Purpose*:** Learn how to use constants in a Java class.

***To Do:***

*Definition*: A **symbolic constant** is a location in memory, referenced by an identifier, where a data value that cannot be changed is stored.

General Form of a Symbolic Constant:

**public static final** *datatype* *CONSTANTNAME* ***=*** *value****;***

## Open the **L04\_Constants** class.

## **Question**: Which identifier is a symbolic constant? GST

## Change the author to yourself.

## Run **L04\_Constants**.

## Add the line

GST = 0.06;

to the **main()** method. What happens? I get errors

## Save the file. Click the **Problems** tab in the bottom panel. Expand the **Errors** list. What is the message associated with this statement?

The final field L04\_Constants.GST cannot be assigned.

## Delete the line that you added in the previous step and modify the program to add the QST calculation:

### Add another symbolic constant, **QST**. It should have a value of **0.085**.

### Add a **double** variable called **qstTaxAmount**.

### Add an assignment statement that calculates and assigns the **qstTaxAmount**. The QST is the **sale** amount plus the **gstTaxAmount** times the **QST** rate.

### Change the **totalBill** assignment statement to include the **qstTaxAmount**.

### Add a **println()** statement to print the QST after printing the GST.

## Test your changes. Your output should look like:

The sale amount is 100.0

The GST owing is 5.0

The QST owing is 8.925

The total bill is 113.925

## Modify the program to use the Math.round() method to round the total bill to two decimal places. The last line of your output should now look like:

The total bill is 113.93

# The Cast operator

***Purpose*:** Learn how to temporary change the data type of a value.

***To Do*:**

## Create a new class called **CastingDemo**. It should include a **main()** method and be part of the **three\_Rs\_II** package.

## Change the author in the comments to your name. Add the following in the comments before your name:

\* This program illustrates how the cast operator works.

## Add the following comment lines before the @param in the Javadoc comments for the main() method.

\* This method demonstrates:

\* - how casting can be used to solve problems with integer division

\* - what happens when a real number is cast to an integer

\* - and how a char is coded as an integer.

## Add the following lines to the **main()** method:

**int** firstNum = 5, secondNum = 6, thirdNum = 10;

**double** newNum;

newNum = firstNum / secondNum \* thirdNum;

System.*out*.println("The answer to " + firstNum + "/"

+ secondNum + "\*"+ thirdNum + " is " + newNum);

## Run the program. What is the value of **newNum**? 0.0

## We want the answer to be a double precision number. We can temporary convert an integer to a double, by preceding the integer with (double). Change the assignment calculation to:

**newNum = (double) firstNum / secondNum \* thirdNum;**

## Run the program again. What is the value of **newNum** this time? 8.33333…

## Add a **double** variable called **myDouble** and an **int** variable called **doubleInt** to the program.

## Add a **Scanner** object called **keyboard** to read from the keyboard.

## Prompt the user for a real number and use the keyboard object to read it into **myDouble**.

## Use the cast operator to assign **myDouble** to **doubleInt**.

## Write a **System.out.println()** statement to display **myDouble** and **doubleInt** with an appropriate message.

## Run the program a few times. Try entering 5.1, 5.5 and 5.7. Does the **(int)** cast operator truncate or round a real number? truncate

## Add a char called **ch**. Initialize it to the first letter of your first name.

## Write an **System.out.println()** statement to display **ch** and **(int)ch**. What was **(int)ch**? 112 Why?

# String Basics

***Purpose:*** Learn to construct a string, retrieve the string length, convert a primitive data type to String and concatenate Strings.

***To Do:***

## Create a new class in the **strings** package called ***Username*StringTest** (replace ***username*** with your logon username). Specify that you want a main() method and that you want to generate comments.

## Add a line in the comments before the author name that states "This program illustrates the String.len() and String.valueOf() methods."

## Change the author name to your name instead of your username.

## Code the **main()** method as follows:

**public class *Username*StringTest**

**{**

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

**{**

**String one = "one";**

**String two = "two";**

**String three;**

**String four = two;**

**String five;**

**String six = "";**

**int numberFour = 4;**

**char a = 'a';**

**System.out.println("one is " + one);**

**System.out.println("two is " + two);**

**System.out.println("three is " + three);**

**System.out.println("four is " + four);**

**System.out.println("five is " + five);**

**System.out.println("six is " + six);**

**} // main()**

**} // StringTest class**

## Two words in the **main()** method will be underlined with a squiqqly red line and a light bulb with an x in a red box will be displayed in the left margin for each of these lines. Eclipse highlights compile errors in this way. Hover over the underlined three. You should see the following error message:

The local variable three may not have been initialized

## The error occurred because **three** and **five** have not been initialized. Add statements to assign the **String three** to **"three"** andthe **String five** to **"five"** before the **println** statements.

## Run the program.

**String Length:**

## Add the following line to the end of the program and run it:

**System.out.println("The length of one is " + one.length());**

## Add code to display the lengths of the Strings **three** and **six**. Run the program.

**Conversion to String:**

## Add the following line before the line to print **four**:

**four = numberFour;**

What happens? Error : can’t convert int to string.

*This is because* **numberFour***is an integer and* **four** *is a String. You can use the* **valueOf()** *method of the* **String** *class to convert any primitive value to its* **String** *equivalent.*

## Change the line that you added in the previous step to the following and run the program:

**four = String.valueOf(numberFour);**

## Add a line before the line to display **six** which assigns **six** to the variable **a**.

## Run the program to test your changes.

**String Concatenation:**

## Declare a new **String** object called **numbers**.

## Add the following statements at the end of the program:

**numbers = one + two;**

**System.out.println("numbers is " + numbers);**

## Run the program.

The **+** sign is the String *concatenation* operator.

***Example***:

**String a = "Science";**

**String c = "Computer " + a;**

The string **c** is the concatenation of the strings "**Computer** " and **a**. It now has the value "**Computer Science**".

## Modify the **numbers** assignment statement so that **numbers** becomes the concatenation of the **String** objects **one**, **two**, **three**, **four** and **five** and any **String** literals that you need. (Include a space between each of the numbers.)

## Add a **println** statement to display the length of **numbers**.

## Run the program to test your changes. Your output should look like:

**numbers is one two three 4 five**

# Review Exercise

## Log on to **Moodle**, go to the **Programming I** course page and complete the **Lab 4 Review Quiz**.