COMP2396B: Object-Oriented Programming and Java
Tutorial 2: Classes and Objects

Mr. Marco Wong

Teaching Flow of this Tutorial

- Javadoc – Proper API Documentation
- Generating Javadoc Files in Eclipse
- Passing Arguments to main() method
  - Command Prompt
  - Eclipse

Javadoc – Proper API Documentation

- Course Learning Outcome 3: [Good Documentation Practices]
  - To learn and appreciate the importance and merits of ***proper comments in source code and API documentations***
- API Documentations (API Docs)
  - Or API Specifications (API Specs)
  - Java™ Platform, Standard Edition 12 API Specification
    - ✓ Overview (Java SE 15 & JDK 15) (oracle.com)
    - ✓ You can ***find MANY examples*** here!

Javadoc – Proper API Documentation

- Documentation Comments (Doc Comments)
  - The special comments in the Java source code that are delimited by the `/** ... */` delimiters.
  - These comments are processed by the Javadoc tool to generate the API docs.
- A doc comment is ***written in HTML*** and ***MUST precede*** a class, field, constructor or method declaration.
- It is made up of ***TWO parts***
  - A Description
    - ✓ There is ***ONLY ONE description block per doc comment***; you cannot continue the description following block tags.
  - Block tags
    - ✓ Starting with an @ character

Javadoc – Proper API Documentation

List of tags (IN ORDER)	Applicable to	Description
@author	classes and interfaces only	Identifies the author.
@version	classes and interfaces only	Specifies the version of a class.
@param	methods and constructors only	Documents a parameter.
@return	methods only	Documents a method's return value.
@exception	@throws is a synonym added in Javadoc 1.2	Identifies an exception thrown by a method or constructor.
@see		Specifies a link to another topic.
@since		States the release when a specific change was introduced.
@serial		Documents a default serializable field.
@deprecated		Specifies that a program element is deprecated.

Javadoc – Proper API Documentation

- Remarks for ordering multiple tags:
  - If desired, groups of tags, can be separated from the other tags by a blank line with a single asterisk.
  - Multiple ***@author*** tags should be listed ***in chronological order*** with the creator of the class listed at the top.
  - Multiple ***@param*** tags should be listed ***in argument-declaration order***. This makes it easier to visually match the list to the declaration.
  - Multiple ***@throws*** tags (also known as @exception as stated in the previous slide) should be ***listed alphabetically*** by the exception names.

Javadoc – Proper API Documentation (Example)

```
import java.io.*;

/**
 * This class demonstrates documentation comments.
 *
 * @author Herbert Schildt
 * @version 1.2
 */
public class SquareNum {
        /**
         * This method returns the square of num. This is a multiline description. You
         * can use as many lines as you like.
         *
         * @param num.
         * @return num squared.
         */
        public double square(double num) {
                return num* num;
        }
```

Description

Insert a blank comment line between the description and the list of tags

Tags in order

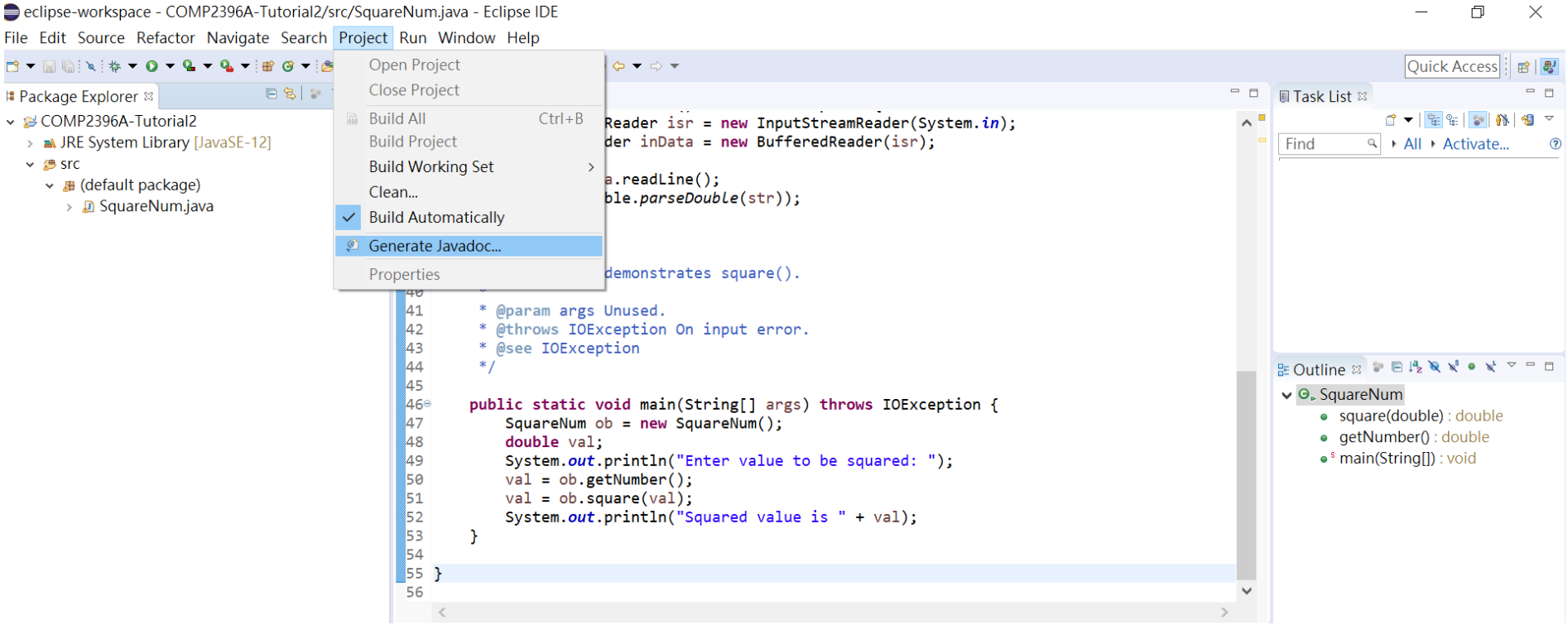
Write the first sentence as a short summary of the method, as Javadoc automatically places it in the method summary table (and index)

Javadoc – Proper API Documentation (Example)

```
        /**
         * This method inputs a number from the user.
         *
         * @return The value input as a double.
         * @exception IOException On input error.
         * @see IOException
         */
        public double getNumber() throws IOException {
                InputStreamReader isr = new InputStreamReader(System.in);
                BufferedReader inData = new BufferedReader(isr);
                String str;
                str = inData.readLine();
                return (Double.parseDouble(str));
                }

        /**
         * This method demonstrates square().
         *
         * @param args Unused.
         * @exception IOException On input error.
         * @see IOException
         */
        public static void main(String args[]) throws IOException {
                SquareNum ob = new SquareNum();
                double val;
                System.out.println("Enter value to be squared: ");
                val = ob.getNumber();
                val = ob.square(val);
                System.out.println("Squared value is " + val);
        }
}
```

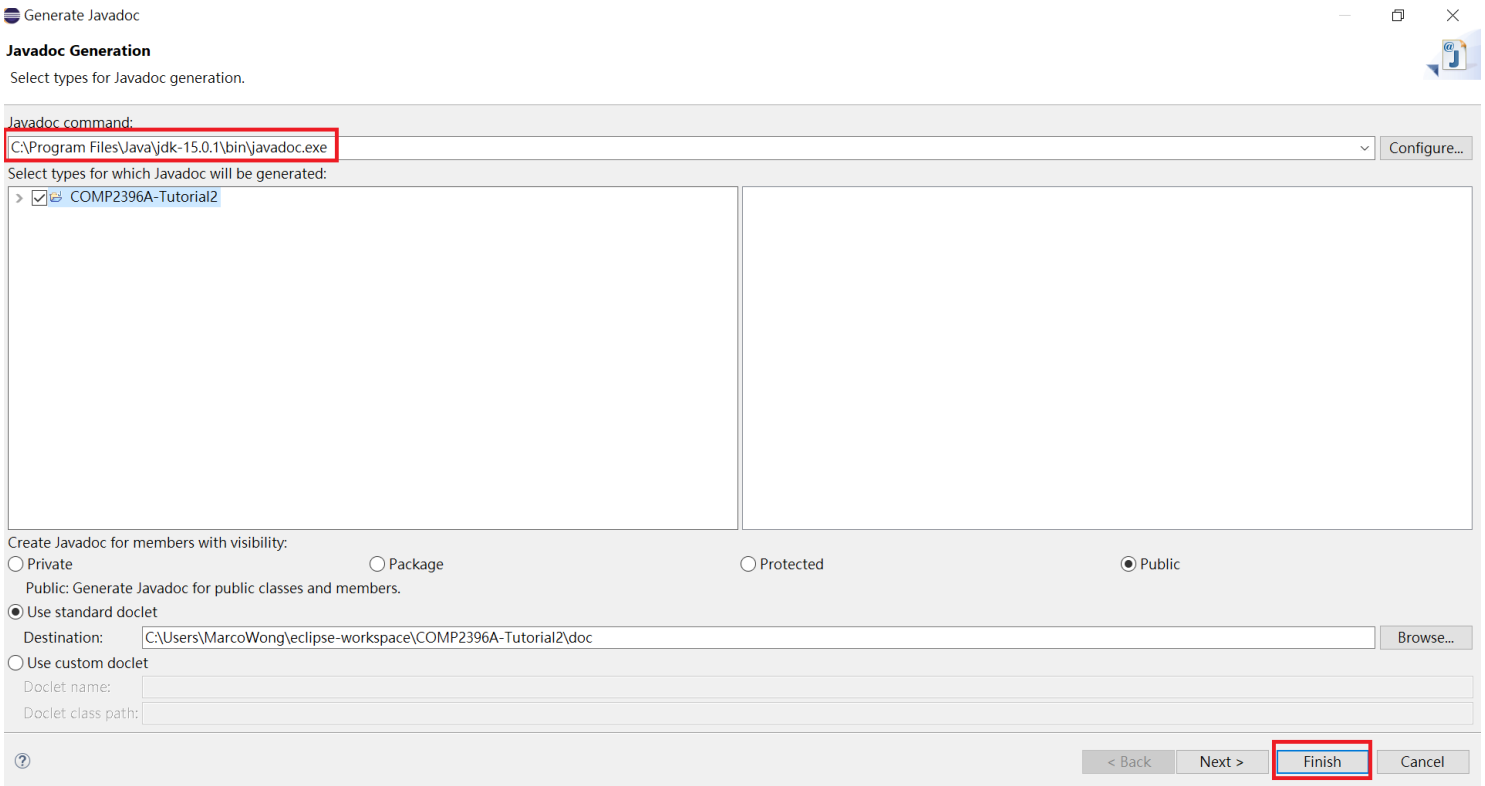
Generating Javadoc Files in Eclipse

- Step 1) Project ➔ Generate Javadoc…



Generating Javadoc Files in Eclipse

- Step 2) Configure Javadoc command by locating ***javadoc.exe***
  - The path is similar to: (It depends on your installation location and version) C:\Program Files\Java\jdk-15.0.1\bin\java**doc**.exe
- Step 3) Then Click "Finish".



Generating Javadoc Files in Eclipse

- Step 4) You can find the Javadoc files from Package Explorer.

Generating Javadoc Files in Eclipse

- Step 5) Double Click "SquareNum.html". You can see a webpage which is similar to Java API Specifications online.

Passing Arguments to main() method

- A Java application can ***accept ANY number of arguments*** from the command line.
- This allows the user to specify configuration information when the application is launched.
- We can pass arguments to main() method by:
  - Command Prompt
    - ✓ We may have to set the PATH environment variable.
  - Eclipse
- A sample program ***Echo.java*** in the following slide can be used as an example.

Passing Arguments to main() method

- Echo.java

```
public class Echo {

	public static void main(String[] args) {
		for (String s : args) {
			System.out.println(s);
		}
	}
}
```

Passing Arguments to main() method – Set the PATH environment variable

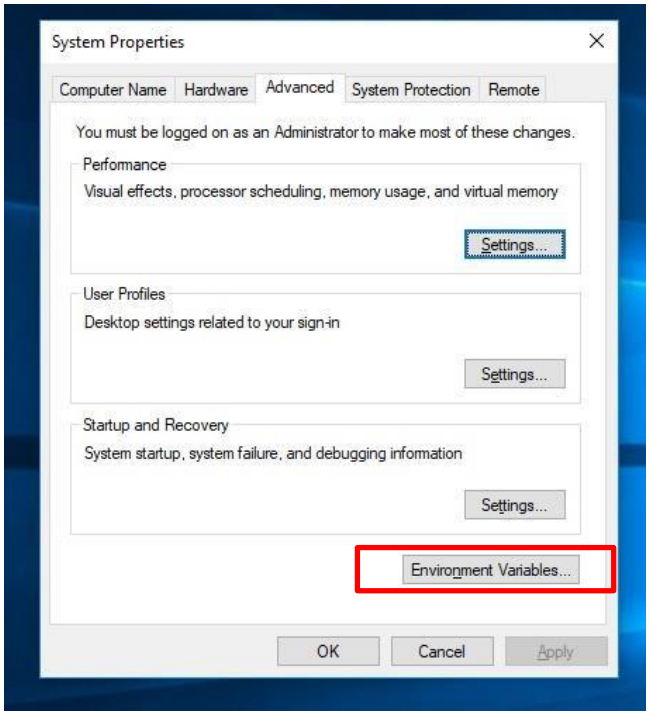
- Problem: When you type "javac Echo.java" in Command Prompt, you may get an error message.

```
Command Prompt
Microsoft Windows [Version 10.0.17134.472]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\user>cd Desktop

C:\Users\user\Desktop>javac Echo.java
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\user\Desktop>_
```

- Solution: Set the PATH environment variable

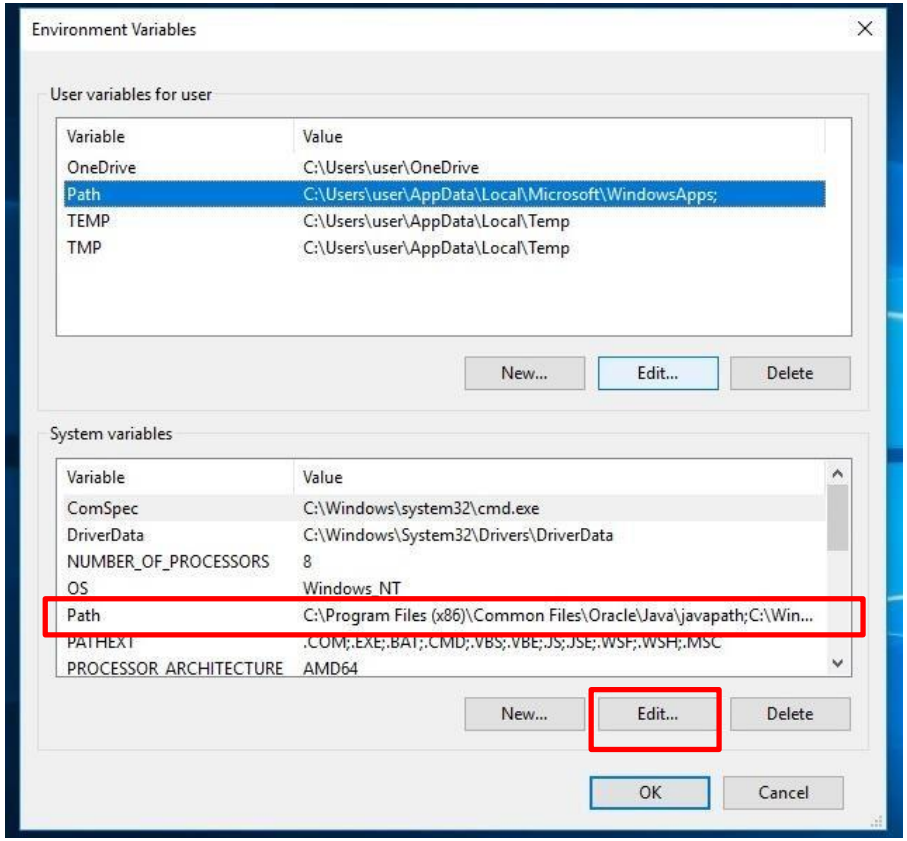
Passing Arguments to main() method – Set the PATH environment variable

- Step 1) In Search, search for and then select: System (Control Panel)
- Step 2) Click the "Advanced system settings".
- Step 3) Click "Environment Variables…".



Passing Arguments to main() method – Set the PATH environment variable

- Step 4) Find the "Path in system variables" and select it.
- Step 5) Click "Edit…".



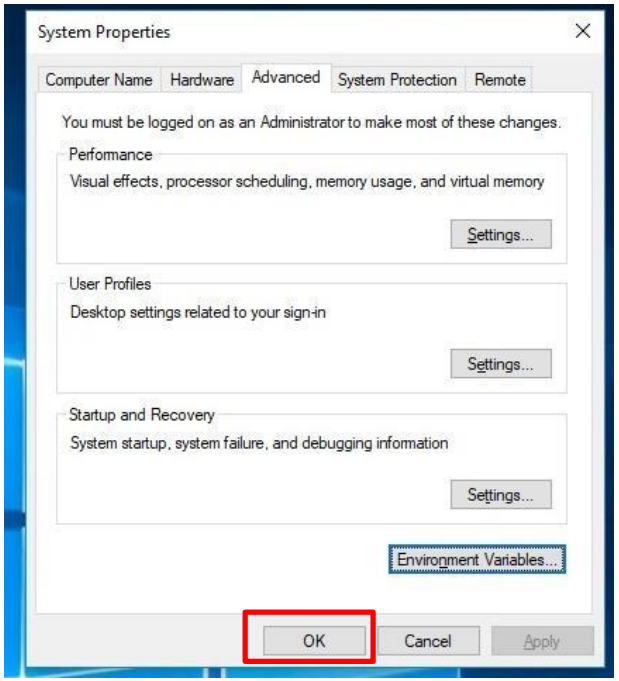
Passing Arguments to main() method – Set the PATH environment variable

- Step 6) Click "New".
- Step 7) Specify the PATH which is similar to "C:\Program Files\Java\jdk-15.0.1\bin".

  (Depends on your installation location and version)
- Step 8) Click "Move Up" until it is moved to the top.
- Step 9) Then Click "OK".

Passing Arguments to main() method – Set the PATH environment variable

- Step 10) Click "OK" to exit from Environment Variables.
- Step 11) Click "OK" to exit from System Properties.



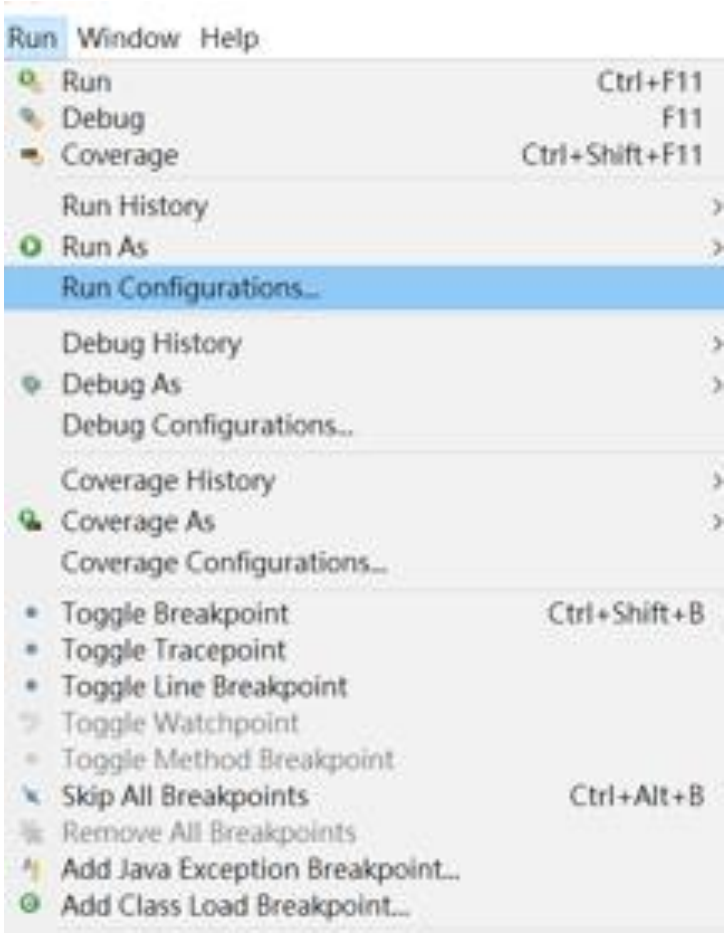
Passing Arguments to main() method – Run Echo.java in Command Prompt

- Step 1) ***REOPEN Command Prompt*** window, and run your java code.
- Step 2) Type "javac Echo.java" to compile "Echo.java".
- Step 3) Type "java Echo Hello COMP2396" to run the Echo program with passing the arguments "Hello COMP2396" to the main() method.

```
C:\Users\MarcoWong\eclipse-workspace\COMP2396A-Tutorial2\src>javac Echo.java

C:\Users\MarcoWong\eclipse-workspace\COMP2396A-Tutorial2\src>java Echo Hello COMP2396
Hello
COMP2396

C:\Users\MarcoWong\eclipse-workspace\COMP2396A-Tutorial2\src>
```

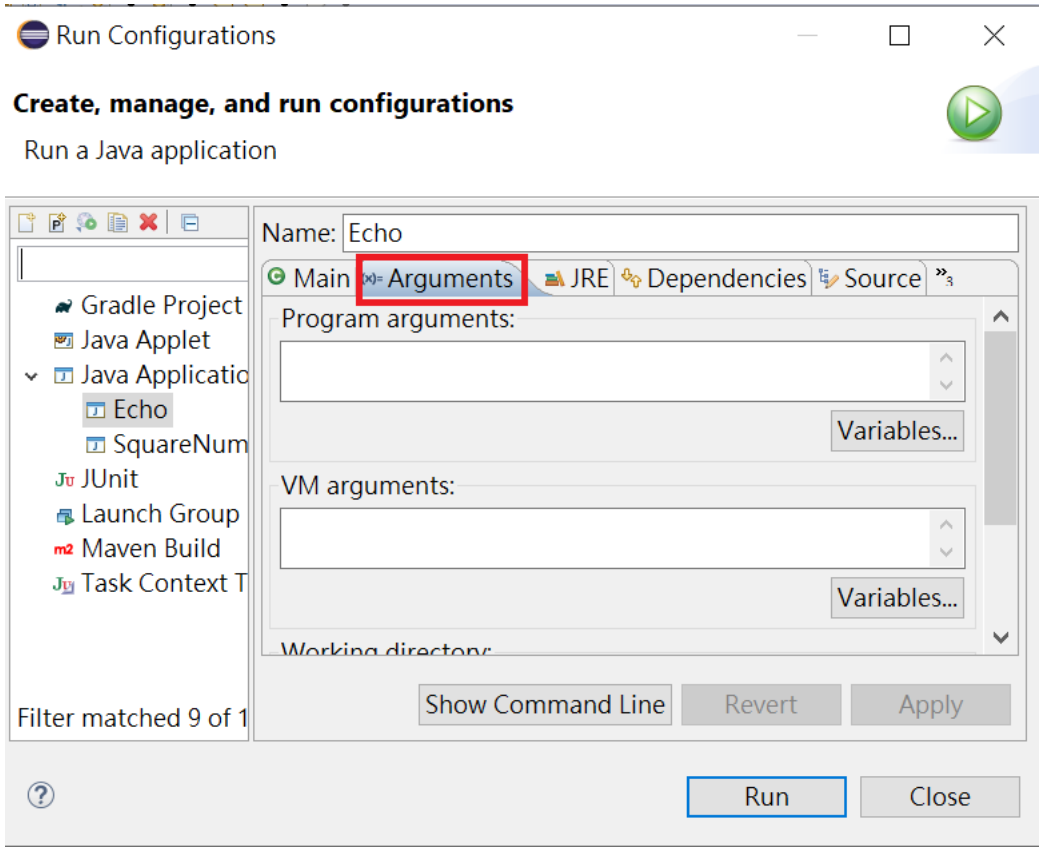
Passing Arguments to main() method – Run Echo.java in Eclipse

- Step 1) Run ➔ Run Configurations…



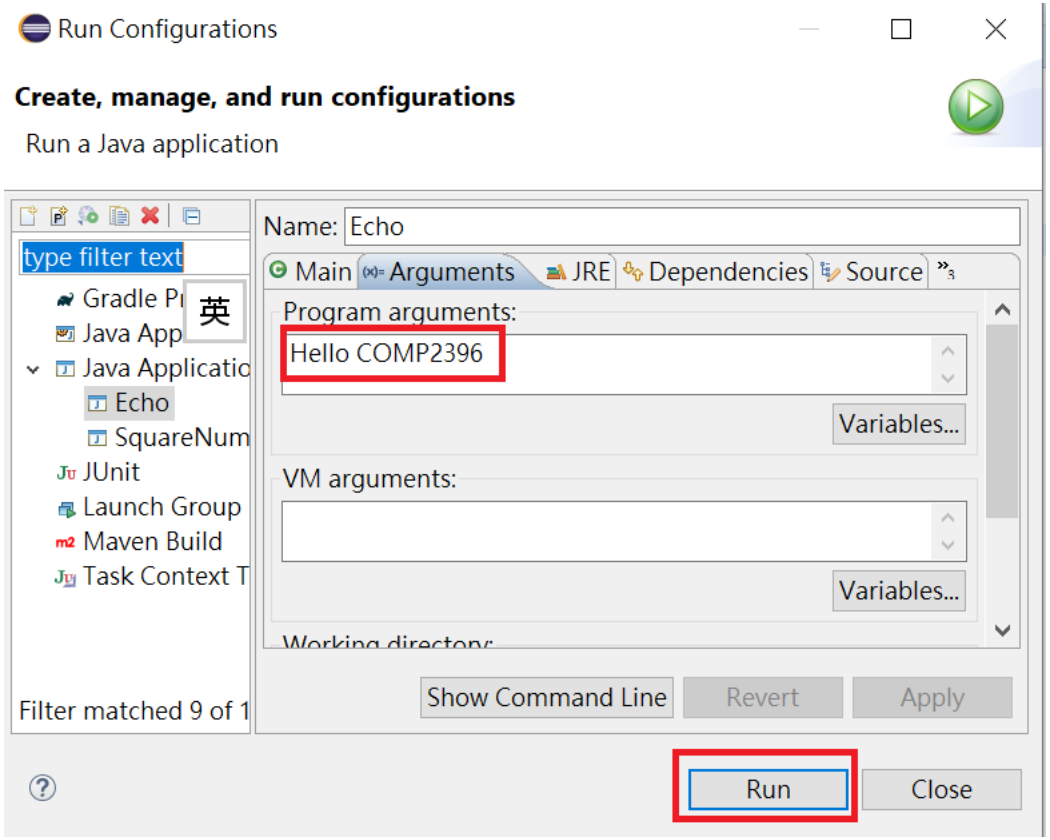
Passing Arguments to main() method – Run Echo.java in Eclipse

- Step 2) Click "Java Application", then select the "Echo" Class. Click "Arguments" tab.



Passing Arguments to main() method – Run Echo.java in Eclipse

- Step 3) Type "Hello COMP2396" as the arguments passed to the main() method.
- Step 4) Click "Run".



Passing Arguments to main() method – Run Echo.java in Eclipse

- You can see the same output from the Console tab.

SquareNum.java | SquareNum | Echo.java

```
1
2 public class Echo {
3
4     public static void main(String[] args) {
5         for (String s:args) {
6             System.out.println(s);
7         }
8
9     }
10
11 }
12
```

Problems | Javadoc | Declaration | Console

```
<terminated> Echo [Java Application] C:\Program Files\Java\jdk \bin\javaw.exe (2019年9月8日 下午10:31:45)
Hello
COMP2396
```

References

This tutorial is modified from COMP2396A: Object-Oriented Programming and Java
Tutorial 2: Classes and Objects by Mr. Justin Yum