COMP2396B Object-oriented Programming and Java

Tutorial 2 – Java Syntax Overview and Javadoc

Outline:

1. Overview on Java Basic Syntax
2. Passing Arguments to main() Method
3. API Documemtation – Javadoc
4. Take-home Exercises

Part 1: Overview on Java Basic Syntax

To learn OOP in this course effectively, we need to be familiar with the basics of Java syntax. Here is a list of items that you should master at the beginning of the course:

1. Primitive Data Types
2. Flow of Control
3. Array Operations
4. Write User-defined Class Member Methods
5. Member Functions of the String Class
6. Member Functions of the Math Class
7. Accept User Input & Console Output

1. Primitive Data Types

Java supports 8 primitive data types:

	Type	Bit Depth	Value Range
	boolean	JVM-specific	true or false
	char	16 bits	0 to 65535
integer	byte	8 bits	–128 to 127
integer	short	16 bits	–32768 to 32767
integer	int	32 bits	–2147483648 to 2147483647
integer	long	64 bits	–huge to huge
floating point	float	32 bits	varies
floating point	double	64 bits	varies

For each data type, you should be familiar with:

1. Variable declaration
2. Operators and operands (e.g. + - * / % > < =, etc.)
3. Data type conversion

To revise the concepts, you may find the following readings useful:

Topic	Readings
Startoff a Java Program	**Course Materials:** Lecture 1 – P.20-31 Tutorial 1 – P.3-12 **Online Materials:** Java Syntax Basics: https://www.w3schools.com/java/java_syntax.asp
Variable Declaration & Operands	**Course Materials:** Lecture 3 – P.2-7 **Online Materials:** Java Data Types: https://www.w3schools.com/java/java_data_types.asp Java Variables: https://www.w3schools.com/java/java_variables.asp Java Operators: https://www.w3schools.com/java/java_operators.asp https://www.w3schools.com/java/java_booleans.asp
Data Type Conversion	**Course Materials:** Lecture 3 – P.8-9 **Online Materials:** Java Type Casting: https://www.w3schools.com/java/java_type_casting.asp

2. Flow of Control

To revise the related concepts, you may find the following readings useful:

Topic	Readings
Branching (e.g. if-then-else)	**Course Materials:** Lecture 1 – P.32 **Online Materials:** If-then-else: https://www.w3schools.com/java/java_conditions.asp
Loops (e.g. while, for, break, continue)	**Course Materials:** Lecture 1 – P.33-37 **Online Materials:** While Loop: https://www.w3schools.com/java/java_while_loop.asp For Loop: https://www.w3schools.com/java/java_for_loop.asp Break / Continue: https://www.w3schools.com/java/java_break.asp

3. Array Operations

To revise the related concepts, you may find the following readings useful:

Topic	Readings
1D Array	**Course Materials:** Lecture 3 – P.17-24 Lecture 1 – P.35-37 **Online Materials:** https://www.w3schools.com/java/java_arrays.asp
Multi-Dimensional Array	**Course Materials:** Lecture 3 – P.25-28 **Online Materials:** https://www.w3schools.com/java/java_arrays_multi.asp

4. **Write User-defined Class Methods**

In Java, users declare functions as class methods inside a class.

```
import java.io.*;

public class example {

    // Create a checkAge() method with an integer variable called age
    static boolean checkAge(int age) {
        if (age < 18) {
            return false;
        }
        else {
            return true;
        }
    }

    public static void main(String[] args) {
        // Call the checkAge method and pass along an age of 20
        boolean check;
        check = checkAge(20);

        if (check == true) {
            System.out.println("Access granted - You are old enough!");
        }
        else {
            System.out.println("Access denied - You are not old enough!");
        }
    }

}
```

Sample Run:

```
Access granted - You are old enough!
```

You may also find the following reading useful to know more about the topic:
https://www.w3schools.com/java/java_methods_return.asp

5. String Class Member Functions

Introduction

In Java, the String class is very important to represent character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

For example,

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

Important Class Member Functions:

Here are some important member functions of the String class:

(a) String Property: length(), chartAt(), indexOf(), equals()

(b) String Manipulations: substring(), concat(), toLowerCase(), toUpperCase(), trim(), replace()

```
import java.io.*;
public class example {
    public static void main(String[] args) {

        String s = "COMP2396";

        // length: Returns the number of characters in the String.
        System.out.println("String length = " + s.length());

        // charAt: Returns the character at ith index.
        System.out.println("Character at 3rd position = " + s.charAt(3));


        // indexOf: Returns the index of the first occurrence of specified string,
        //          starting at the specified index.
        String s1 = "Learn Share Learn";
        System.out.println("Index of a = " + s1.indexOf('a', 3));

        // equals: Checking equality of Strings
        Boolean out = "Chim".equals("chim");
        System.out.println("Checking Equality = " + out);

        // substring: Returns the substring from i to j-1 index.
        System.out.println("Substring = " + s1.substring(2, 5));

        // concat: Concatenates string2 to the end of string1.
        String s2 = "Dr. ";
        String s3 = "TW Chim";
        System.out.println("Concatenated string = " + s2.concat(s3));
```

```
        // toLowerCase: Converting cases
        String word1 = "TW";
        System.out.println("Changing to lower Case = " + word1.toLowerCase());

        // toUpperCase: Converting cases
        String word2 = "tw";
        System.out.println("Changing to UPPER Case = " + word2.toUpperCase());

        // trim: Trimming the word
        String word3 = " Learn Share Learn ";
        System.out.println("Trim the word = " + word3.trim());

        // replace: Replacing characters
        String str1 = "Dr Thim";
        System.out.println("Original String = " + str1);
        String str2 = "Dr Thim".replace('T' ,'C') ;
        System.out.println("Replaced T with C -> " + str2);

    }
}
```

Sample Run:

```
String length = 8
Character at 3rd position = P
Index of a  = 8
Checking Equality  false
Substring  = arn
Concatenated string  = Dr. TW Chim
Changing to lower Case tw
Changing to UPPER Case TW
Trim the word Learn Share Learn
Original String Dr Thim
Replaced T with C -> Dr Chim
```

You may also find the following reading useful to know more about the String class:
https://www.geeksforgeeks.org/string-class-in-java/

6. Math Class Member Functions

The usage of Math class member functions is frequent in assignments / examinations.

```
import java.io.*;

public class example {
    public static void main(String[] args) {

       int i1 = Math.max(5, 10);
       int i2 = Math.min(5, 10);
       int i3 = (int)(Math.sqrt(64));
       int i4 = (int)(Math.random() * 101);  // 0 to 100

       System.out.println("max() output: " + i1);
       System.out.println("min() output: " + i2);
       System.out.println("sqrt() output: " + i3);
       System.out.println("random() output: " + i4);

    }
}
```

Sample Run:

```
max() output: 10
min() output: 5
sqrt() output: 8
random() output: 68
```

You may also find the following reading useful to know more about the Math class:
https://www.javatpoint.com/java-math

7. Accept User Input & Console Output

Example 1 – Basics

In this example, we are using the BufferedReader stream with the InputStreamReader stream for reading the line by line data from the keyboard.

We use readLine() to read a line of text. The readline() throws an IOException if an I/O error occurs. You should either add throws declaration or use try/catch to surround it.

```
import java.io.*;

public class example {
    public static void main(String args[]) throws Exception {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader inData = new BufferedReader(isr);

        System.out.println("Enter your name:");

        String name = inData.readLine();
        System.out.println("Welcome " + name);
    }
}
```

Sample Run:

```
Enter your name:
Chan
Welcome Chan
```

Example 2 – Reading Input Continuously

In this example, we read and sum the input numbers until the user enters an negative number.

```
import java.io.*;
public class example2 {
    public static void main(String[] args) throws IOException {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader inData = new BufferedReader(isr);

        String str = inData.readLine();
        Double n = Double.parseDouble(str);

        double sum = 0;
        while (n >= 0) {
            sum += n;
            str = inData.readLine();
            n = Double.parseDouble(str);
        }

        System.out.println("Sum of the numbers: " + sum);
    }
}
```

Sample Run:

```
2
10
7
-1
Sum of the numbers: 19.0
```

Example 3 – Breaking Down Input Line into an Array

In this example, we first read an array of data delimited by a space character, then place it into an array in the program.

```
import java.io.*;
import java.util.Arrays;

public class example3 {
    public static void main(String[] args) throws IOException {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader inData = new BufferedReader(isr);

        String str = inData.readLine();
        int[] data = Arrays.stream(str.split(" ")).mapToInt(Integer::parseInt).toArray();

        for (int i: data) {
             System.out.println(i);
        }
    }
}
```

Sample Run:

```
5 6 7 8
5
6
7
8
```

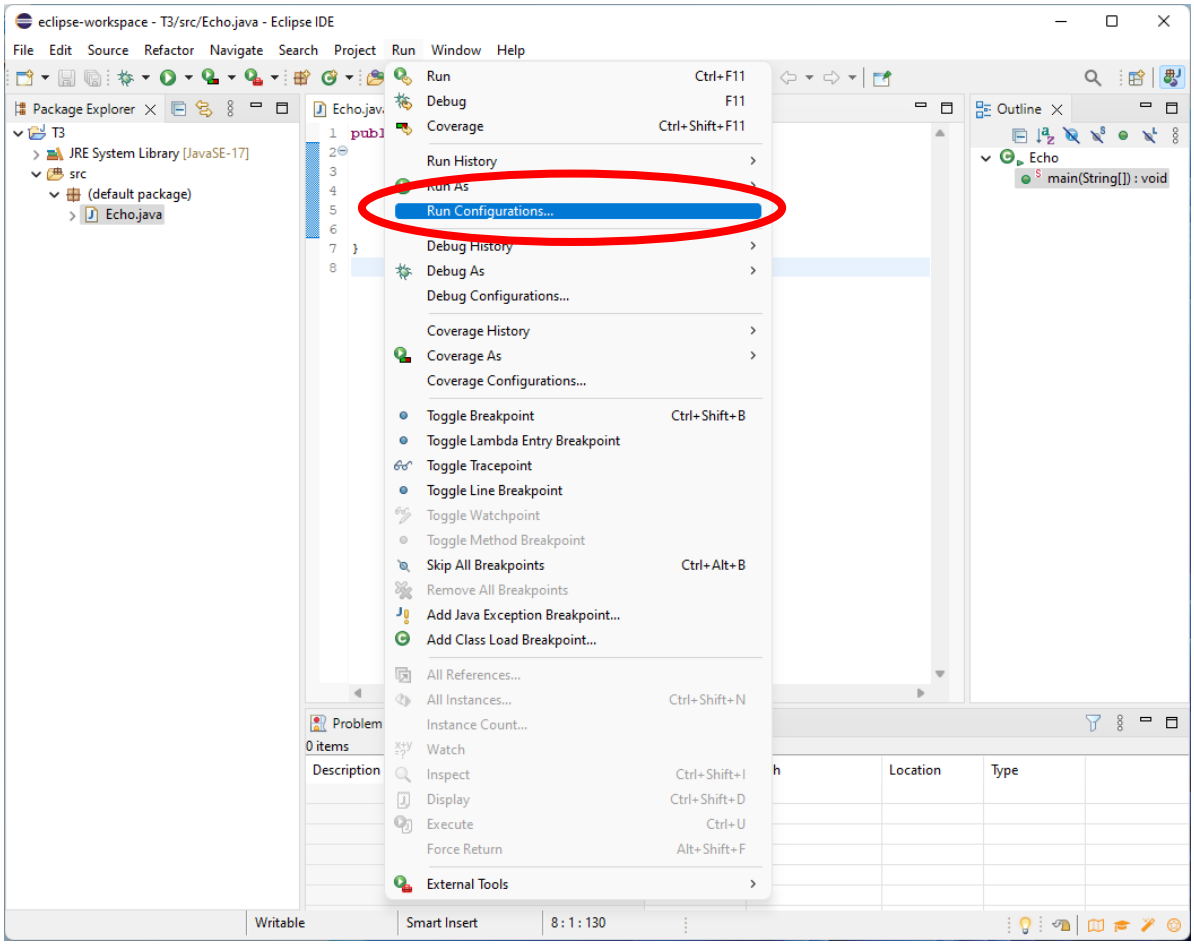
Part 2: Passing Arguments to main() Method

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.

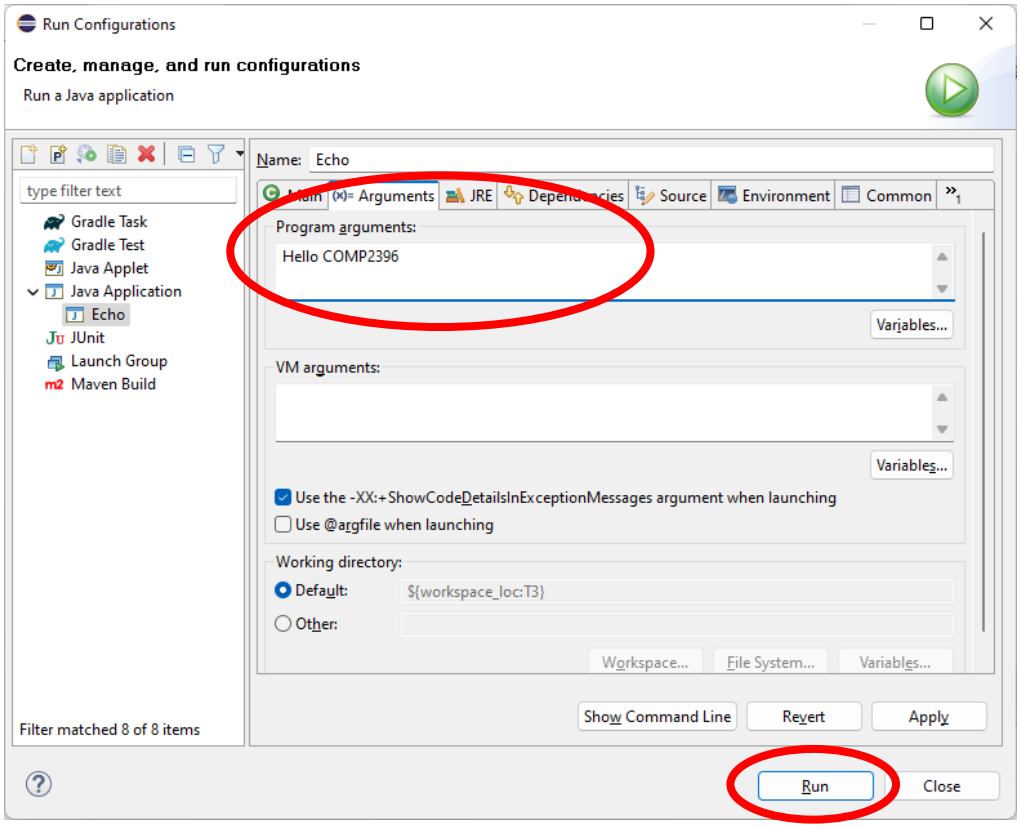
This program reads the command line arguments and prints them to the console.

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

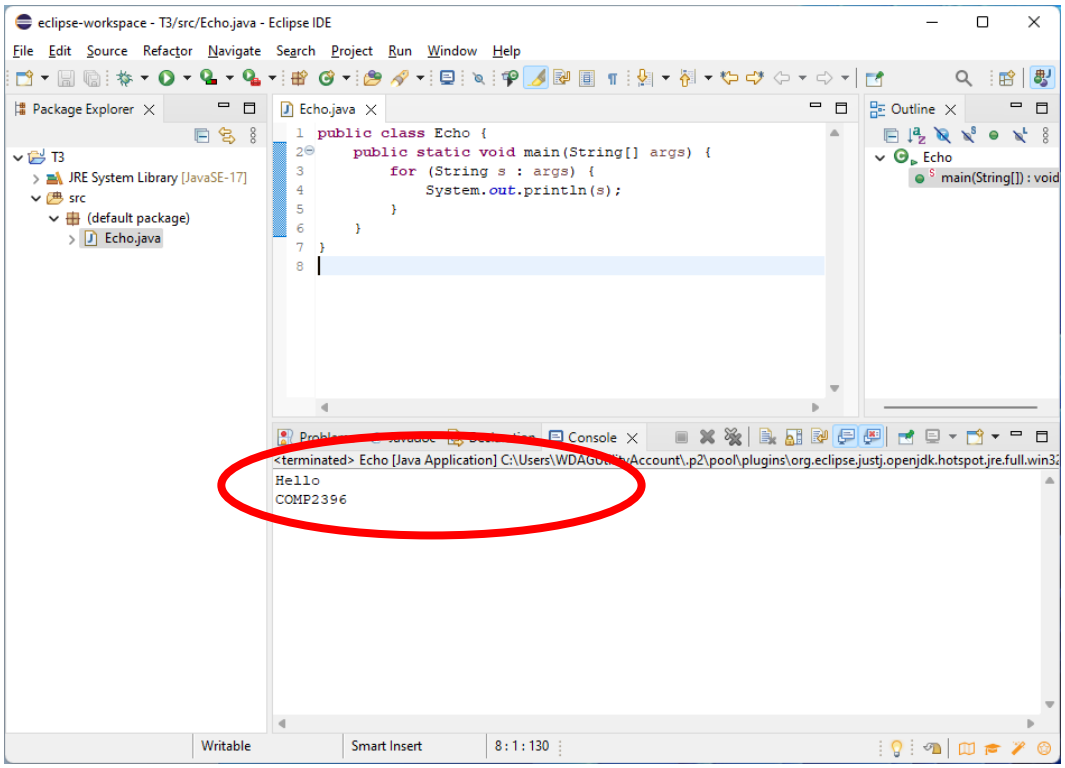
1. Click “Run” -> “Run Configurations…”



2. Navigate to “Java Application”, then select the class you are passing arguments to. In “program arguments”, type the argument to be passed. Then click “Run”



3. You can see the pogram can read the arguments.



Part 3: API Documemtation - Javadoc

Why should we learn it?
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
Documenting your code is important to help others understand it. API documentation is a technical content deliverable, containing instructions about how to use and integrate with your classes. It's a reference manual containing all the information required to work with the classes, with details about the functions, classes, return types, arguments.

Example:
The official API documentations of the Java 20
https://docs.oracle.com/en/java/javase/20/docs/api/index.html

Popular Way to Write API Documentations - Javadoc
Javadoc is a tool for generating API documentation in HTML format from **doc comments** in source code.

Javadoc comments structure look very similar to a regular multi-line comment, but the key difference is the extra asterisk at the beginning:

```
// This is a single line comment

/*
 * This is a regular multi-line comment
 */

/**
 * This is a Javadoc comment
 */
```

Javadoc comments may be placed above any class, method, or field which we want to document.

These comments are commonly made up of two sections:

1. The description of what we're commenting on.
2. The standalone block tags (marked with the "@" symbol) which describe specific meta-data.

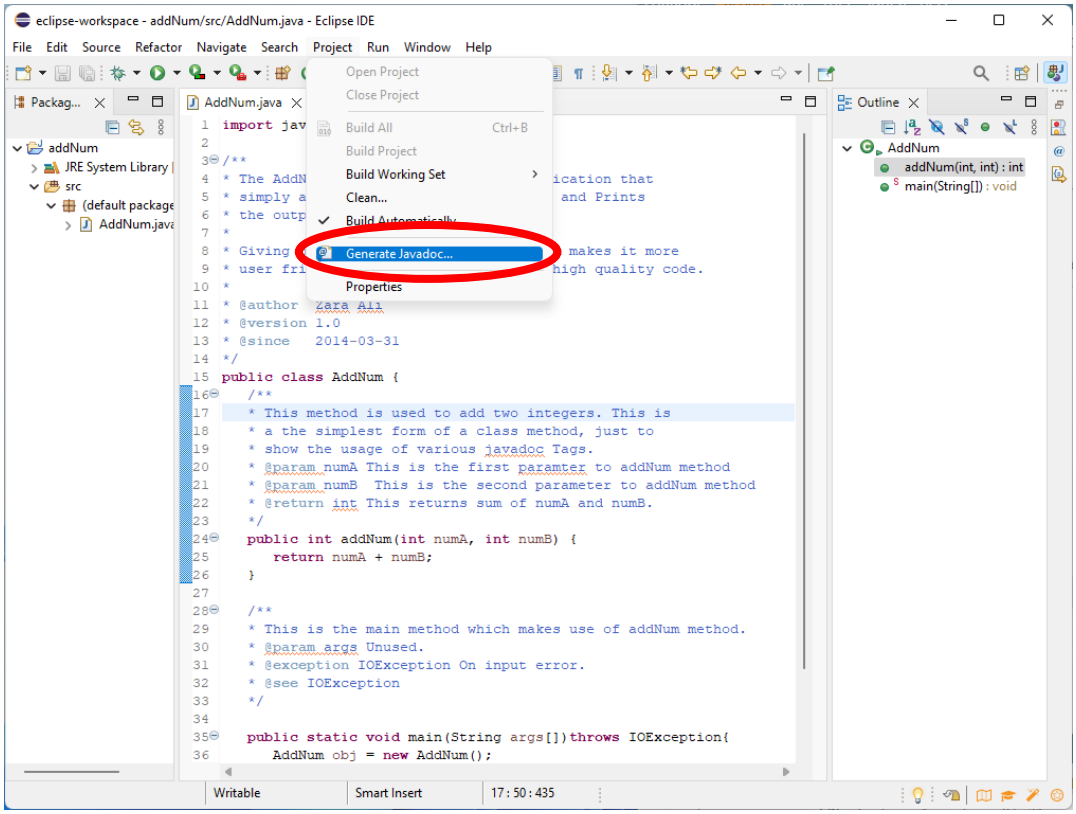
Tags (In order)	Applicable to	Description
@author (required)	classes and interfaces only	Identifies the author.
@version (required)	classes and interfaces only	Specifies the version of aclass.
@param	methods and constructors only	Documents a parameter
@return	methods only	Documents a method's return value.
@exception		Identifies an exception thrown by a method or constructor.
@see		Specifies a link to anothertopic.
@since		States the release when a specific change was introduced
@serial		Documents a default serializable field
@deprecated		Specifies that a program element is deprecated

Example

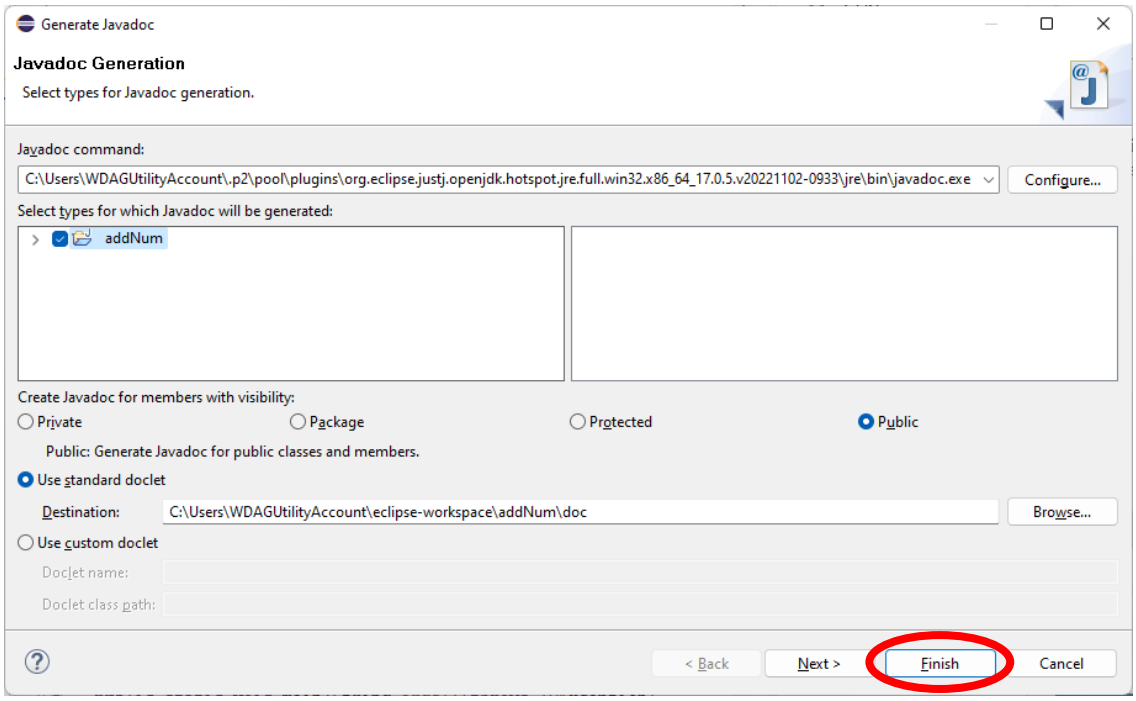
```
import java.io.IOException;

/**
* The AddNum program implements an application that
* simply adds two given integer numbers and Prints
* the output on the screen.
*
* Giving proper comments in your program makes it more
* user friendly and it is assumed as a high quality code.
*
* @author  Zara Ali
* @version 1.0
* @since   2014-03-31
*/
public class AddNum {

   /**
   * This method is used to add two integers. This is
   * a the simplest form of a class method, just to
   * show the usage of various javadoc Tags.
   * @param numA This is the first paramter to addNum method
   * @param numB This is the second parameter to addNum method
   * @return int This returns sum of numA and numB.
   */
   public int addNum(int numA, int numB) {
      return numA + numB;
   }

   /**
   * This is the main method which makes use of addNum method.
   * @param args Unused.
   * @exception IOException On input error.
   * @see IOException
   */
   public static void main(String args[]) throws IOException{
      AddNum obj = new AddNum();
      int sum = obj.addNum(10, 20);

      System.out.println("Sum of 10 and 20 is :" + sum);
   }
}
```

Generating Javadoc in Eclipse

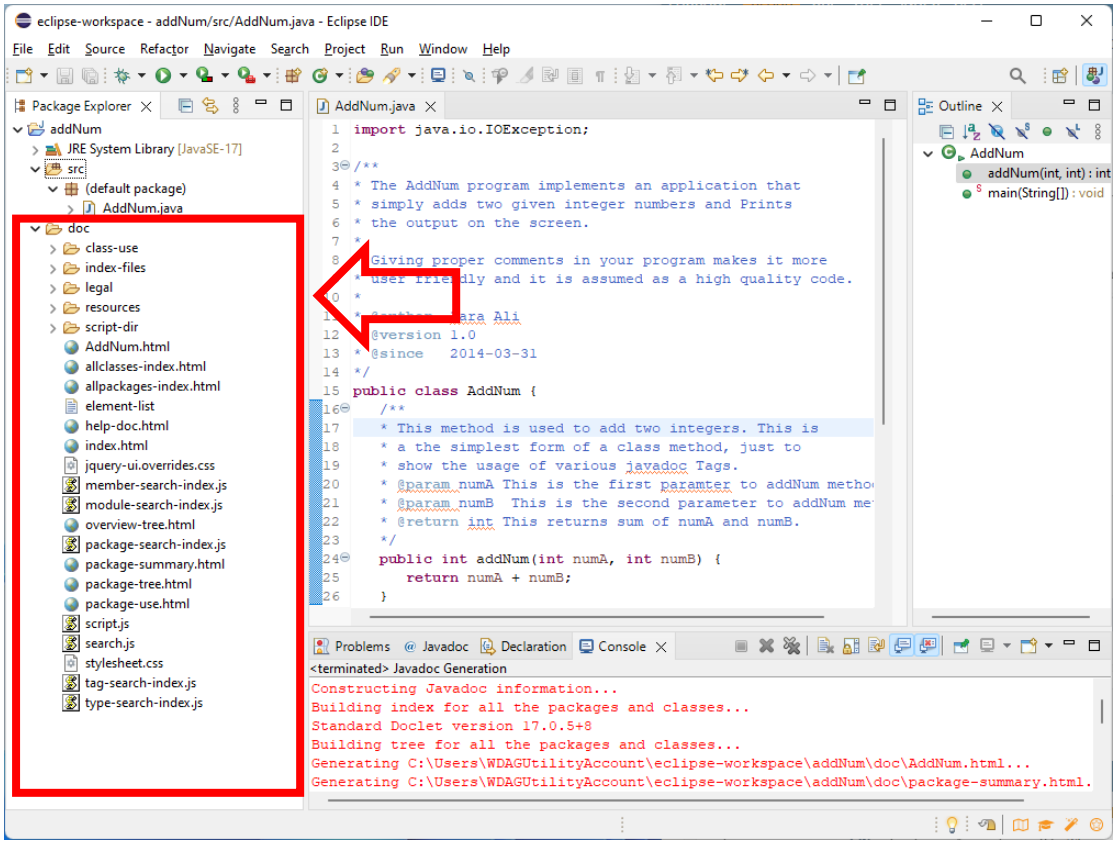
1. Click “Project” -> “Generate Javadoc”



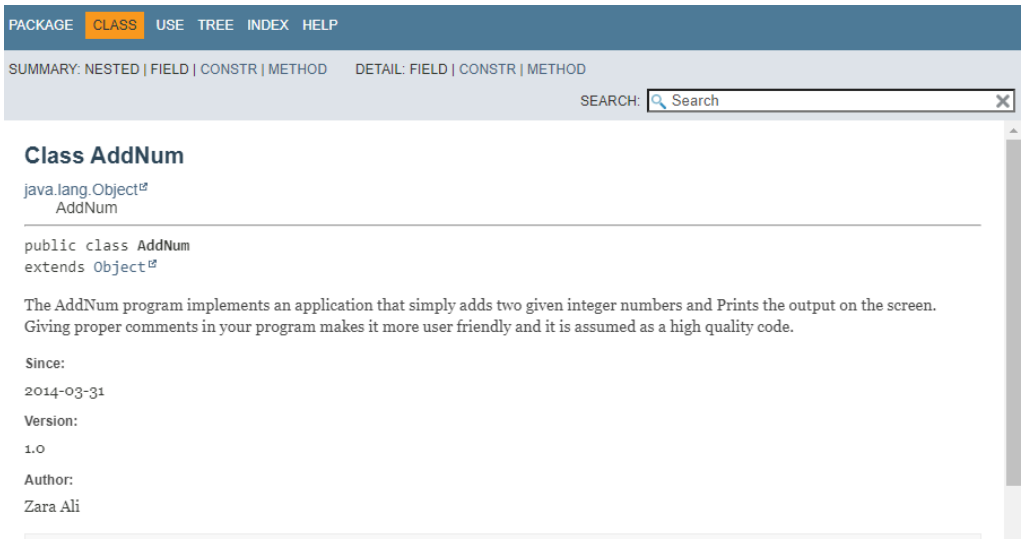
2. Click “Finish”



3. You can find the Javadoc files from Packet Explorer



4. Open “index.html”. You can see the API documemetion in a webpage, which is similar to the Java API Specifiction online.



Reference:

1. Introduction to Javadoc - https://www.baeldung.com/javadoc
2. Java - Documentation Comments - https://www.tutorialspoint.com/java/java_documentation.htm

Part 4: Take Home Exercises

The exercises below serves as self practice and no submission is required. However, you may test your program by VPL on Moodle to check it's correctness!

1. For any HKID number, the check digit can be calculated using the following algorithm:

   - For the first character, treat 'A'as 1, 'B'as 2, …, and 'Z'as 26, to obtain 7 numbers. Multiply the first number by 8, the second number by 7, …, and the 7^{th} number by2. Divide the sum of these products by 11 to obtain a remainder.
   - Subtract this remainder from 11 to obtain the check digit. If the check digit obtained is 10, treat the check digit as 'A'.If the check digit obtained is 11, treat the check digit as '0'.

   Write a Java program Question1.java which takes a HKID from user input, and output the check digit.

   Sample Test Cases:

   Test case 1:
   Input:
   A123456

   Output:
   3

   Explanation:
   We treat 'A'as 1 to obtain 7 numbers 1, 1, 2, 3, 4, 5, 6. The remainder is calculated as following:

   Check digit = $11 - (1 \times 8 + 1 \times 7 + 2 \times 6 + 3 \times 5 + 4 \times 4 + 5 \times 3 + 6 \times 2) \% 11$
   $= 11 - 85 \% 11 = 11 - 8 = 3$

2. For any positive integer n, define d(n) to be n plus the sum of the digits of n. The number n is called a generator of d(n). For example, d(75) = 75 + 7 + 5 = 87.

   Some numbers have more than one generator. For example, 101 has two generators, 91 and 100. A number with no generators is a self number. There are 13 self numbers less than 100: 1, 3, 5, 7, 9, 20, 31, 42, 53, 64, 75, 86, and 97.

   Write a Java program Question2.java which takes an input N from user input, and output all positive self numbers less than N in increasing order.

   Sample Test Cases:

   Test case 1:
   Input:
   30

   Output:
   1
   3
   5
   7
   9

   Test case 2:
   Input:
   35

   Output:
   1
   3
   5
   7
   9
   20
   31

Sample Solution:

1. Question1.java

```
import java.io.*;

public class Question1 {
    public static char CheckDigit(String HKID) {
        int sum = (HKID.charAt(0) - 64) * 8;
        for (int i = 1; i <= 6; i ++) {
            sum += (HKID.charAt(i) - 48) * (8 - i);
        }

        int check = 11 - sum % 11;
        if ((check >= 0) && (check <= 10)) {
            return (char)(check + 48);
        }
        else {
            return 'A';
        }
    }

    public static void main(String[] args) throws IOException {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader inData = new BufferedReader(isr);

        String input = inData.readLine();
        System.out.println(CheckDigit(input));
    }
}
```

2. Question2.java

```
import java.io.*;

public class Question2 {
    public static int d(int n) {
        int sum = n;
        while (n >= 10) {
            sum += n % 10;
            n /= 10;
        }
        sum += n;

        return sum;
    }

    public static boolean IsSelfNumber(int n) {
        for (int i = 1; i < n; i++) {
            if (d(i) == n) {
                return false;
            }
        }

        return true;
    }

    public static void main(String[] args) throws IOException {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader inData = new BufferedReader(isr);

        String input = inData.readLine();
        int n = Integer.parseInt(input);

        for (int i = 1; i < n; i++) {
            if (IsSelfNumber(i)) {
                 System.out.println(i);
            }
        }
   }
}
```