

# COMP2026 Problem Solving Using Object Oriented Programming

## Laboratory 1

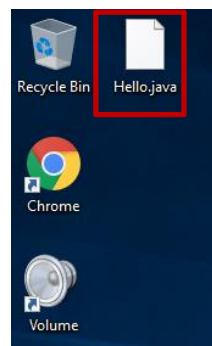
### Part A: Running Java Program in Command Prompt

#### 1. Introduction

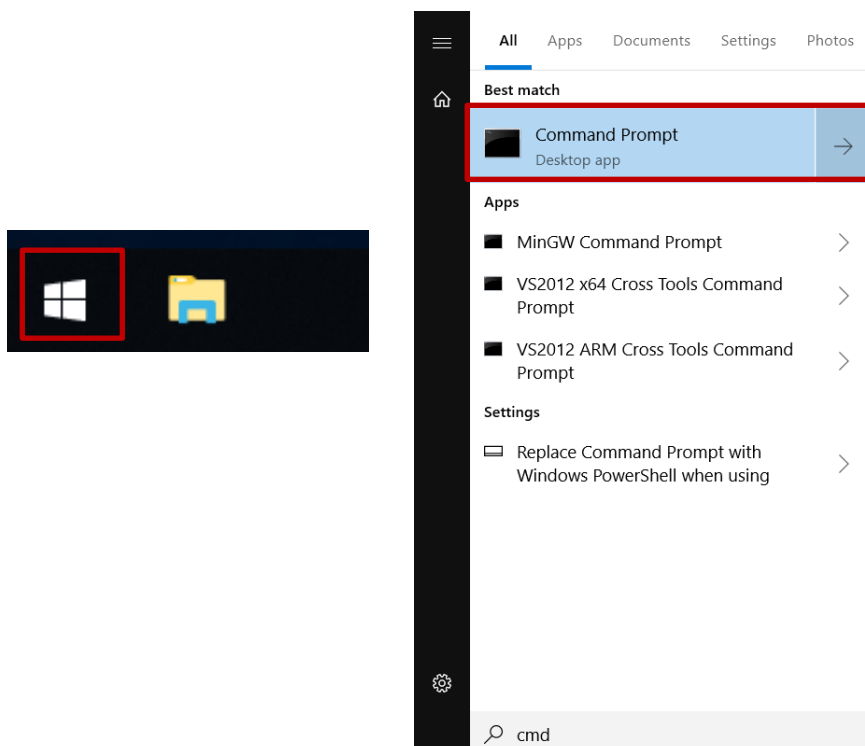
In this section, you will use the Java compiler **javac** to compile your Java programs and the Java interpreter **java** to run them.

#### 2. Starting up Command Prompt

Step 1: Put the program **Hello.java** on Desktop.



Step 2: Click the **Start** and type “**cmd**” to search for the Command Prompt Application.



Step 3: In the command prompt, type in “c:” and press **Enter** to navigate to the C drive.

```
Command Prompt
Microsoft Windows [Version 10.0.16299.461]
(c) 2017 Microsoft Corporation. All rights reserved.

m:\>c:

C:\>_
```

Step 4: Use **cd** command to navigate to the Desktop directory. In the command prompt, type in “**cd Users\login\_name\Desktop**” and press **Enter**.

Note: Type “**cd ~/Desktop**” for MacOS

```
Command Prompt
Microsoft Windows [Version 10.0.16299.461]
(c) 2017 Microsoft Corporation. All rights reserved.

m:\>c:

C:\>cd Users\sandylo\Desktop

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

Step 5: Type in “**javac Hello.java**” and press **Enter** to compile the program. If everything went well, you should see no error messages.

```
Command Prompt
Microsoft Windows [Version 10.0.16299.461]
(c) 2017 Microsoft Corporation. All rights reserved

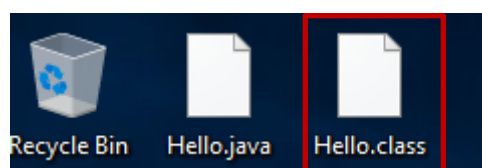
m:\>c:

C:\>cd Users\sandylo\Desktop

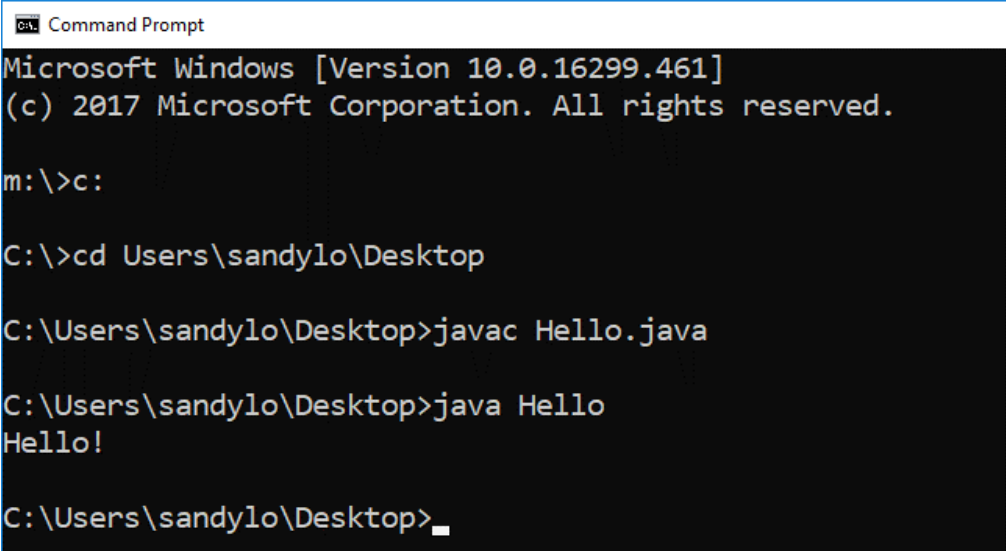
C:\Users\sandylo\Desktop>javac Hello.java

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

Step 6: After successfully compiled the Java source file, the class file is created. The class file contains Java bytecode that can be executed by JVM.



Step 7: Type in “**java Hello**” and press **Enter** to execute the program.



```
Command Prompt
Microsoft Windows [Version 10.0.16299.461]
(c) 2017 Microsoft Corporation. All rights reserved.

m:\>c:

C:\>cd Users\sandylo\Desktop

C:\Users\sandylo\Desktop>javac Hello.java

C:\Users\sandylo\Desktop>java Hello
Hello!

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

Useful commands:

Windows Command	MacOS Command	Description	Example
exit	exit	Close the window	exit
cd	cd	Change directory	cd MyFolder
dir	ls	List directories / files	dir ls
copy	cp	Copy file	copy src.txt new.txt cp src.txt new.txt
move	mv	Move file	move test.txt MyFolder mv test.txt MyFolder
mkdir	mkdir	Create a new directory	mkdir newfolder
del	rm	Delete a file	del test.txt rm test.txt

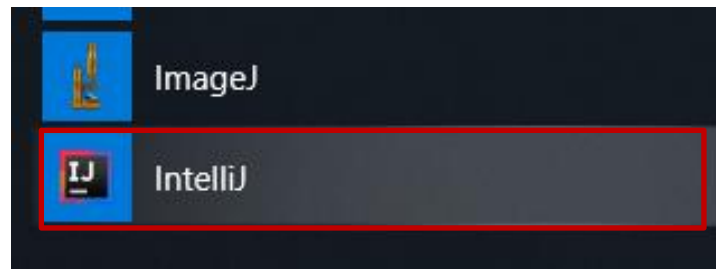
## Part B: Our Programming Environment – IntelliJ IDEA

### 1. Introduction

In this course, you are going to do Java programming using IntelliJ IDEA development environment. This session provides instructions for using it.

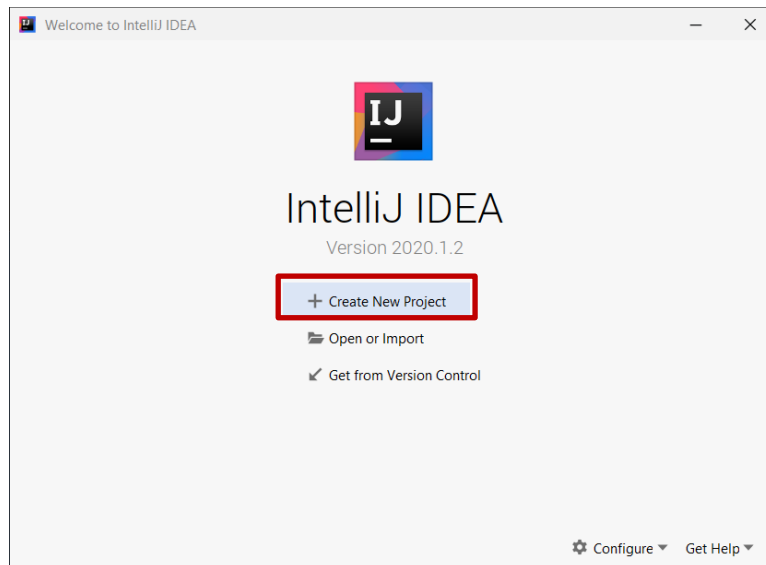
### 2. Starting up IntelliJ IDEA

Locate **IntelliJ** under the Windows menu.

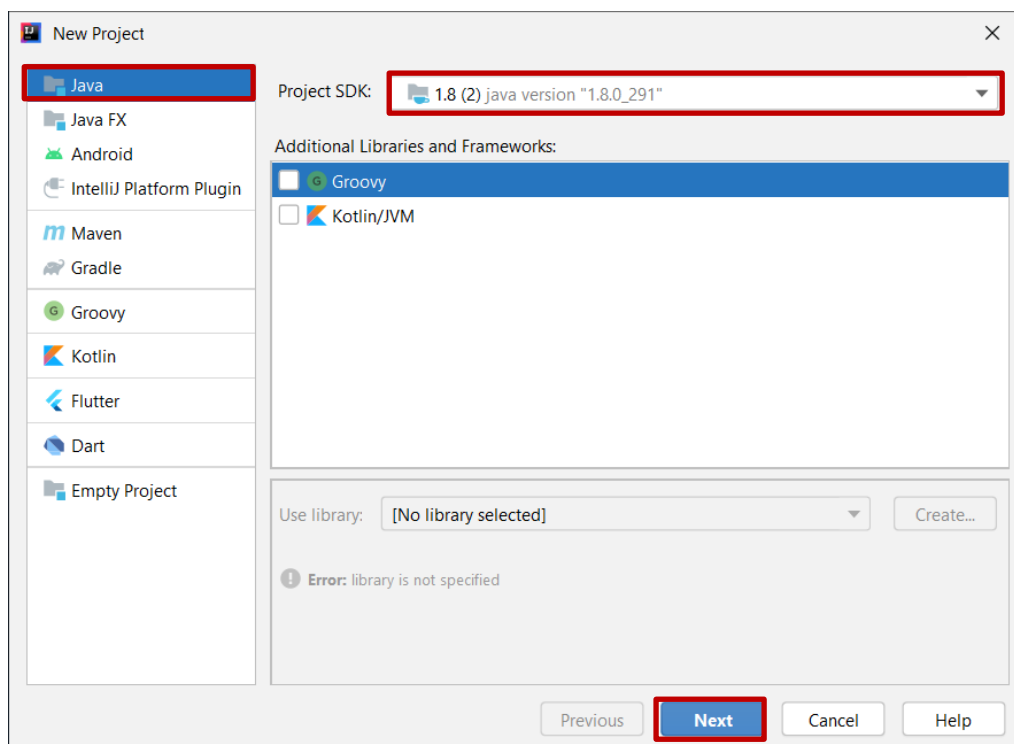


### 3. Creating a new Java Project

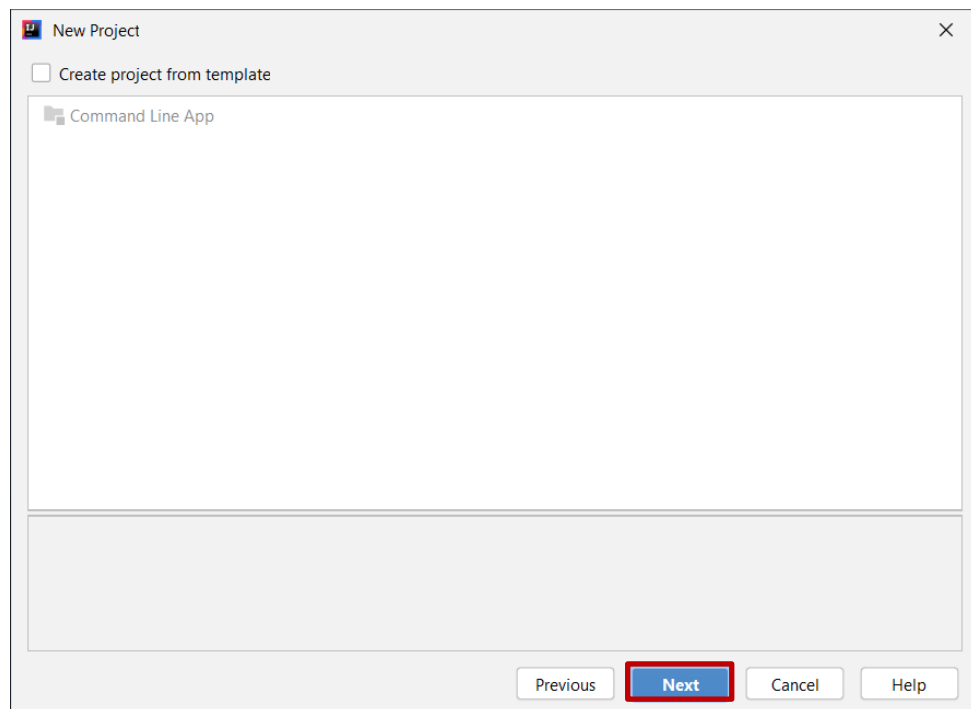
Step 1: Click “**Create New Project**” in the Welcome screen.



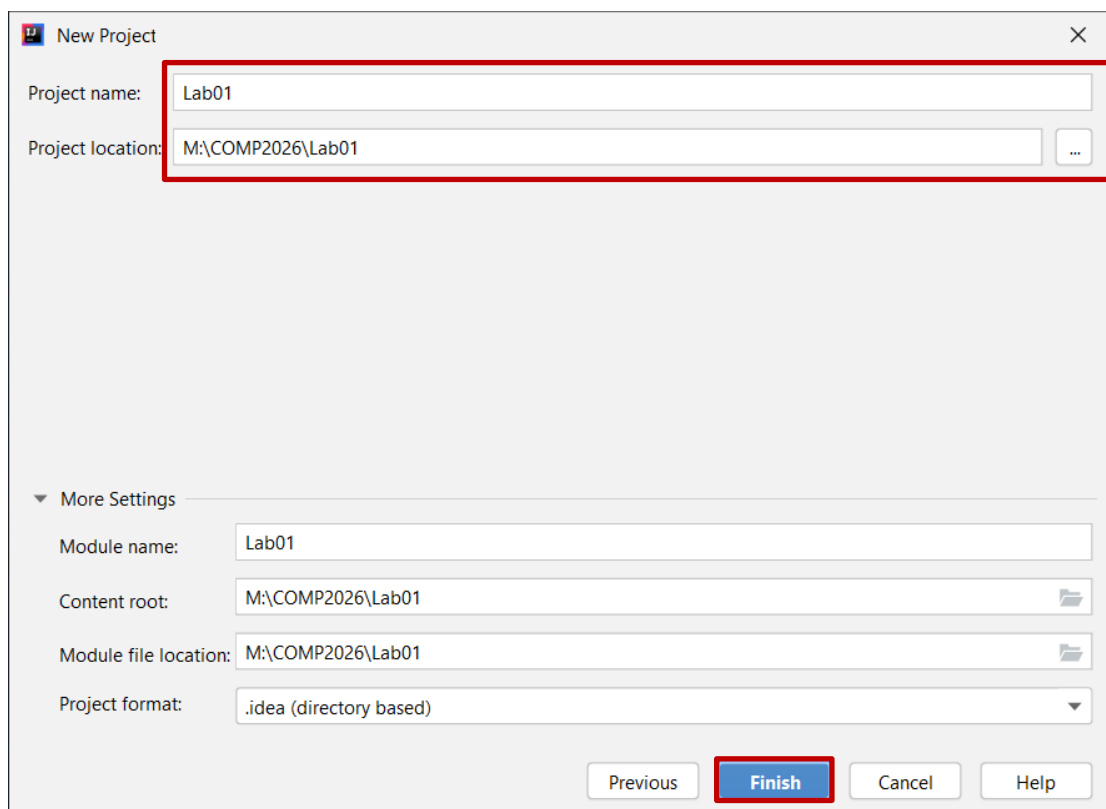
Step 2: Select **Java**, and then choose the preferred **Project SDK** and click **Next**.



Step 3: Click **Next** to continue.

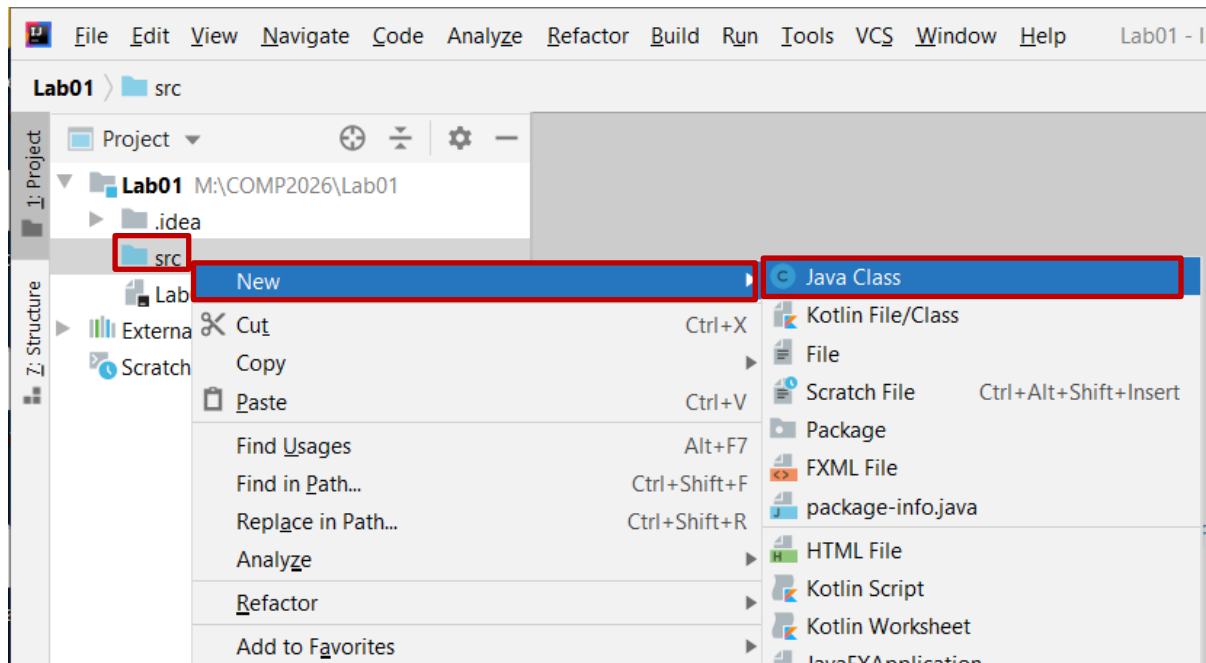


Step 4: Type in the project name “**Lab01**” and the project location “**M:\COMP2026\Lab01**” to create an empty project and then click **Finish**.

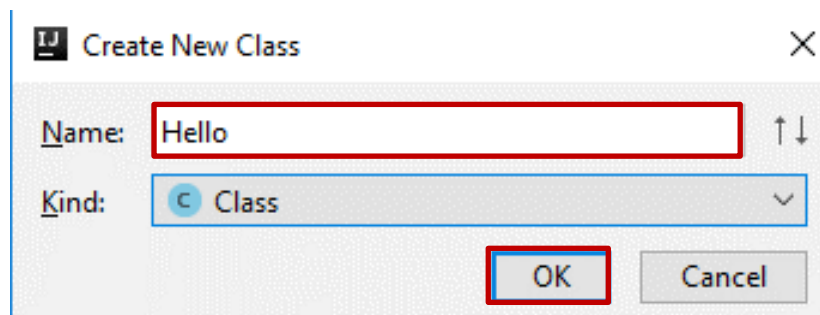


## 4. Creating new File

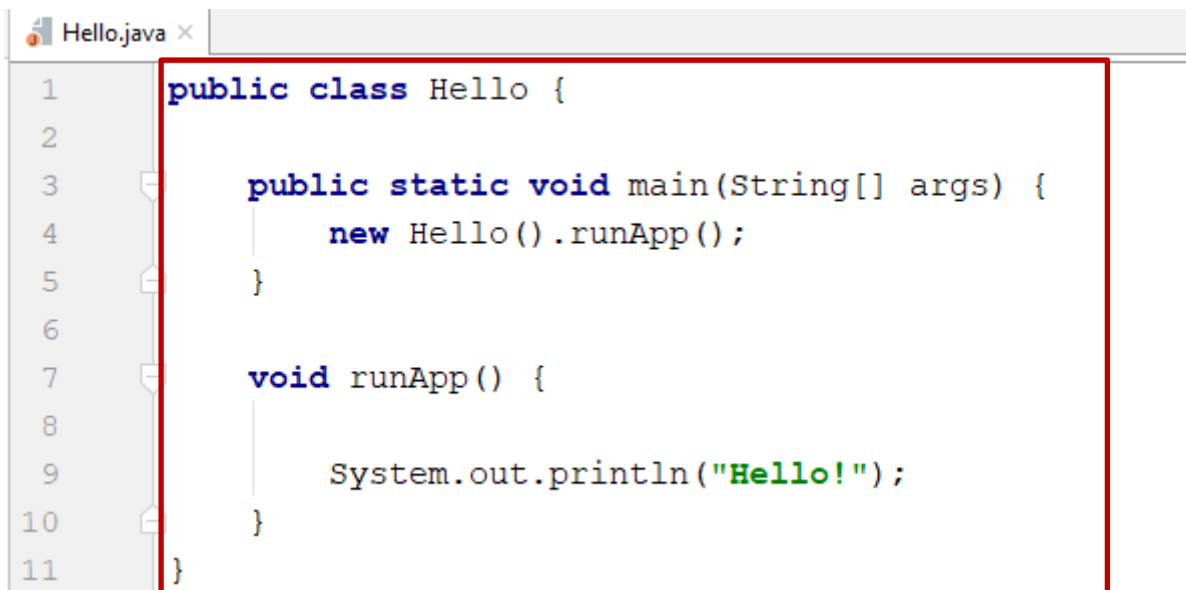
Step 1: To create a new Java source file to this project, you should **right-click** on the **src** folder in the Project Explorer, select **New** and choose **Java Class**.



Step 2: Type in the source file name ***“Hello”*** and click **OK**.

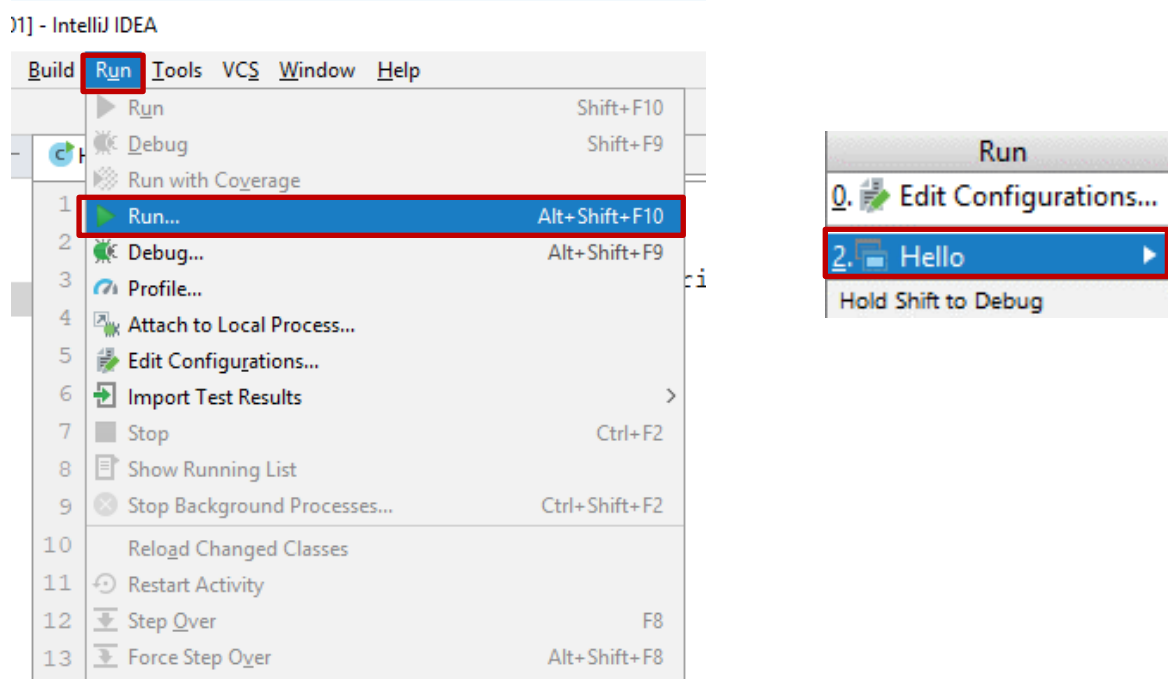


Step 3: Type in a simple **Hello** program.

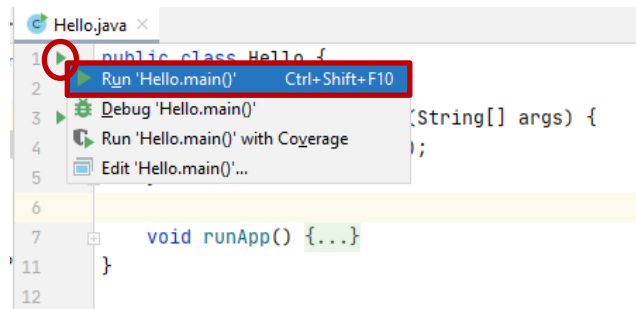
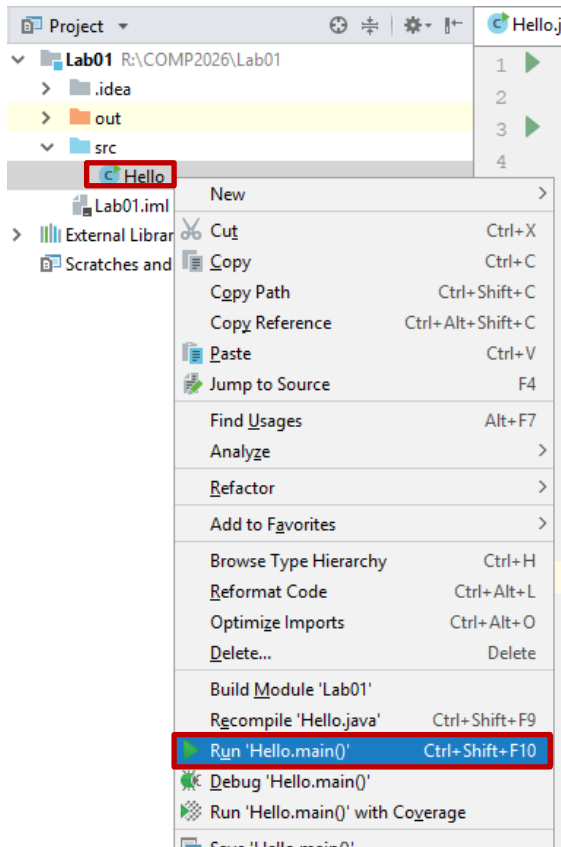


```
1 public class Hello {  
2  
3     public static void main(String[] args) {  
4         new Hello().runApp();  
5     }  
6  
7     void runApp() {  
8  
9         System.out.println("Hello!");  
10    }  
11 }
```

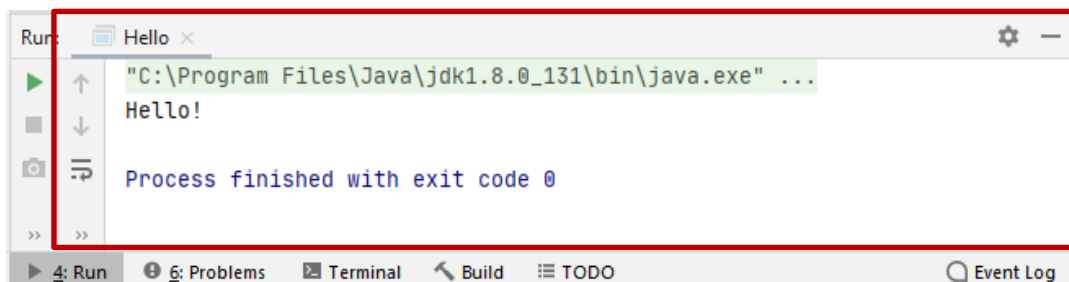
Step 4: Execute the program by clicking the **Run** under the **Run menu** and then select the **Hello** program.



Step 5: Alternatively, the program can also be executed by right-clicking program in the Project Explorer or clicking the green arrow in the gutter and then select **Run 'Hello.man()'**. You may also use the shortcut key **Ctrl+Shift+F10**.



Step 6: Result will be shown at the lower region of IntelliJ IDEA.



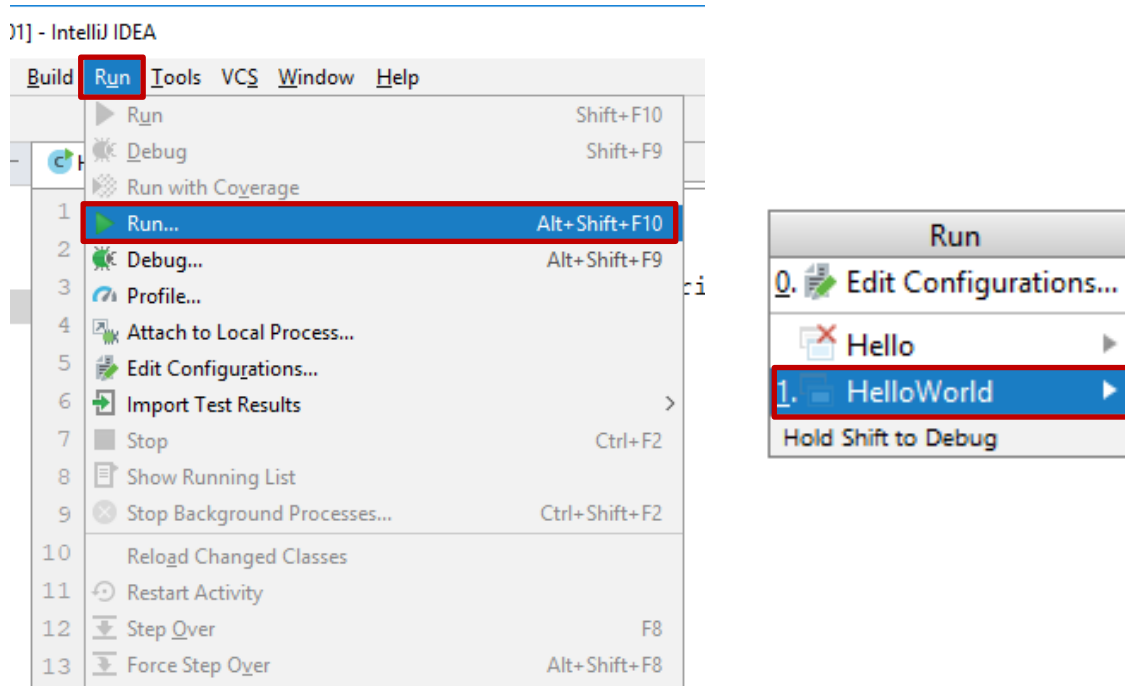


## 5. Renaming a Java Class

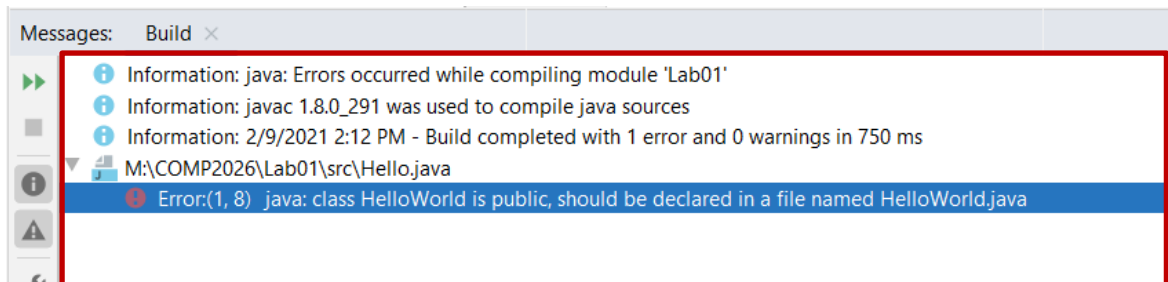
Step 1: Modify the content of **Hello.java** as follow.

```
1 public class HelloWorld {
2
3     public static void main(String[] args) {
4         new HelloWorld().runApp();
5     }
6
7     void runApp() {
8
9         System.out.println("Hello, World!");
10    }
11 }
12
```

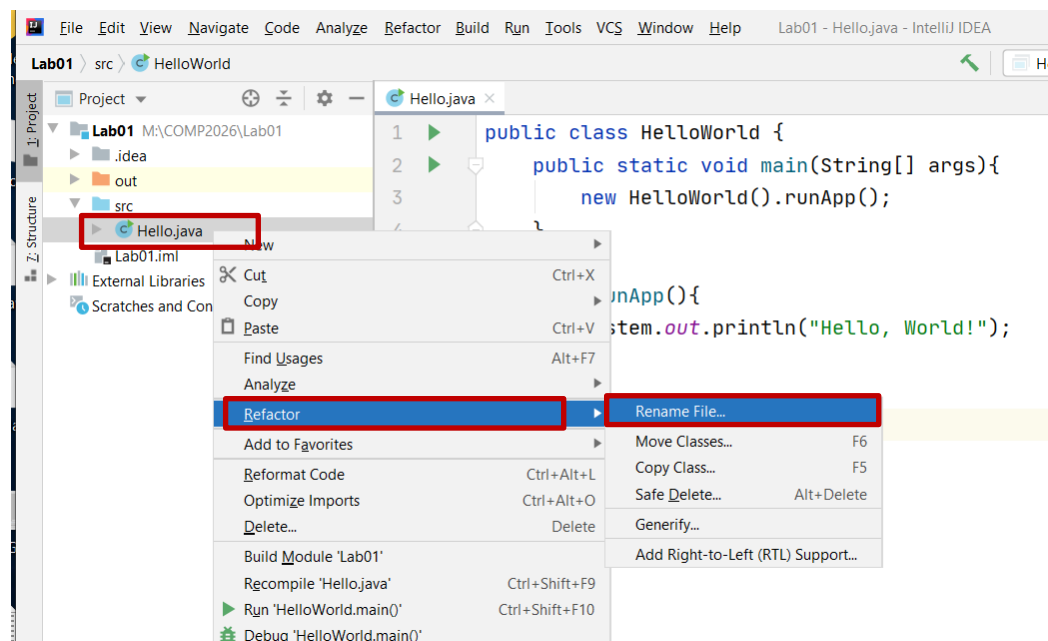
Step 2: Execute the program by clicking the **Run** under the **Run menu** and then select the **HelloWorld** program.



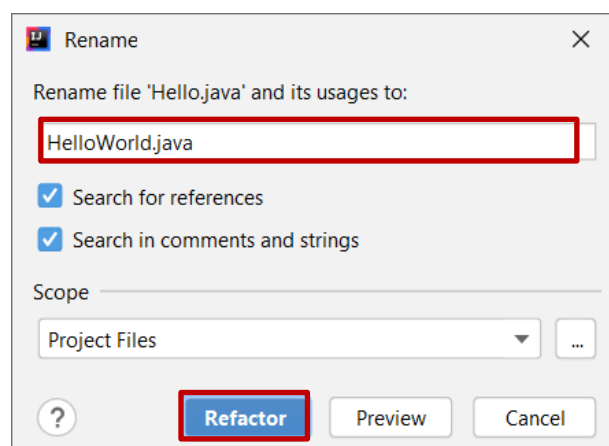
Step 3: Error found as the class name and file name are not the same.



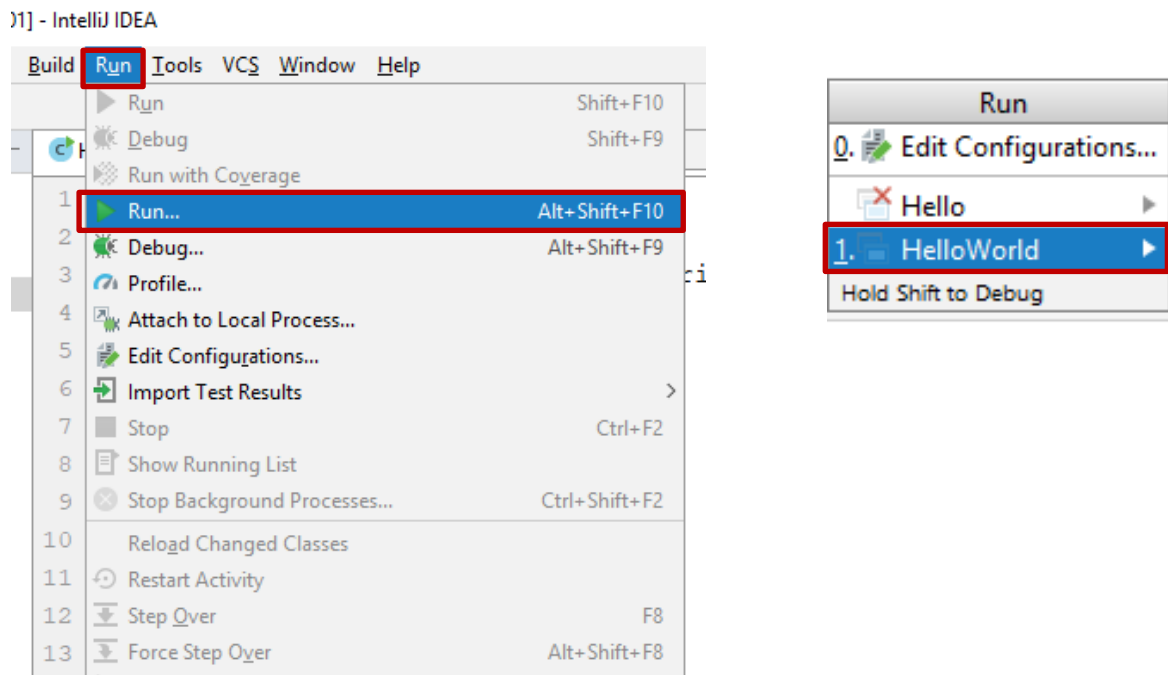
Step 5: Rename the java file by right-clicking the **Hello.java**, choose **Refactor** and then **Rename File....**



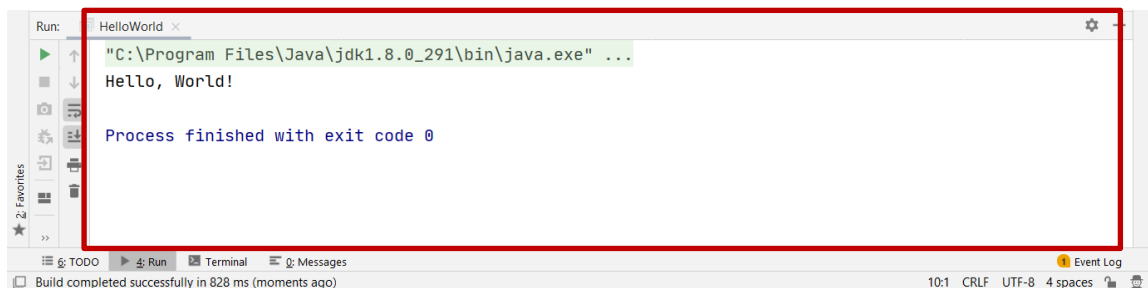
Step 6: Type in **"HelloWorld.java"** as the new file name and click **Refactor**.



Step 7: Execute the program by clicking the **Run** under the **Run** menu and then select the **HelloWorld** program.

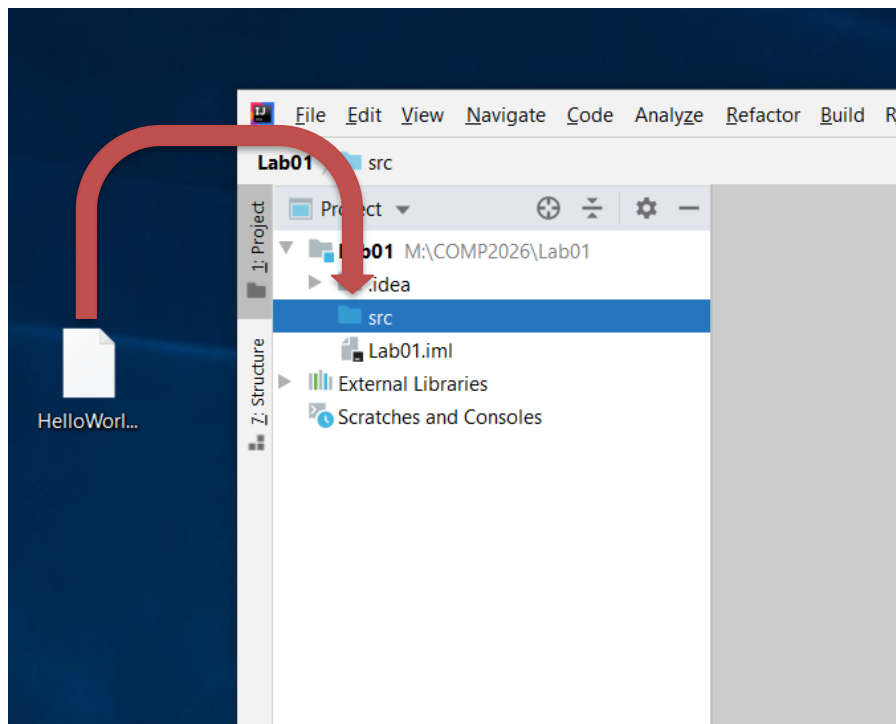


Step 8: Result will be shown at the lower region of IntelliJ IDEA.

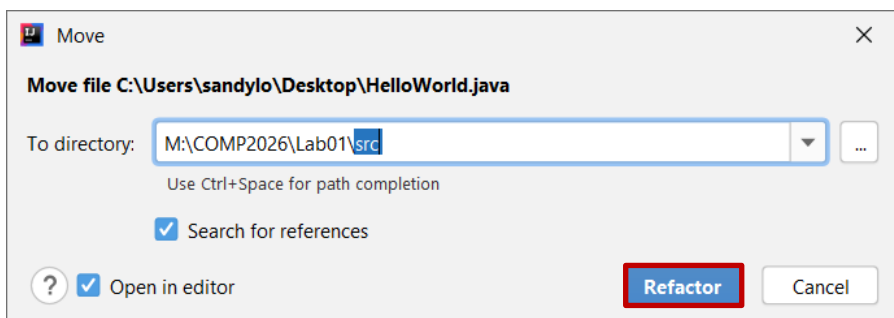


## 6. What if you already have a Java source file?

Step 1: Drag the Java source file into the **src** folder of the project.

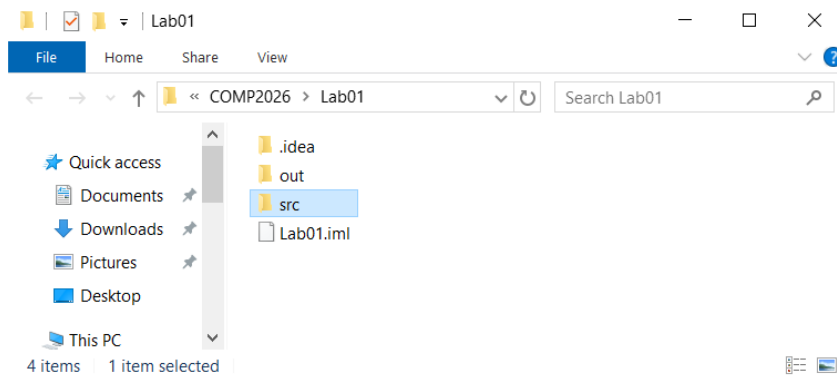
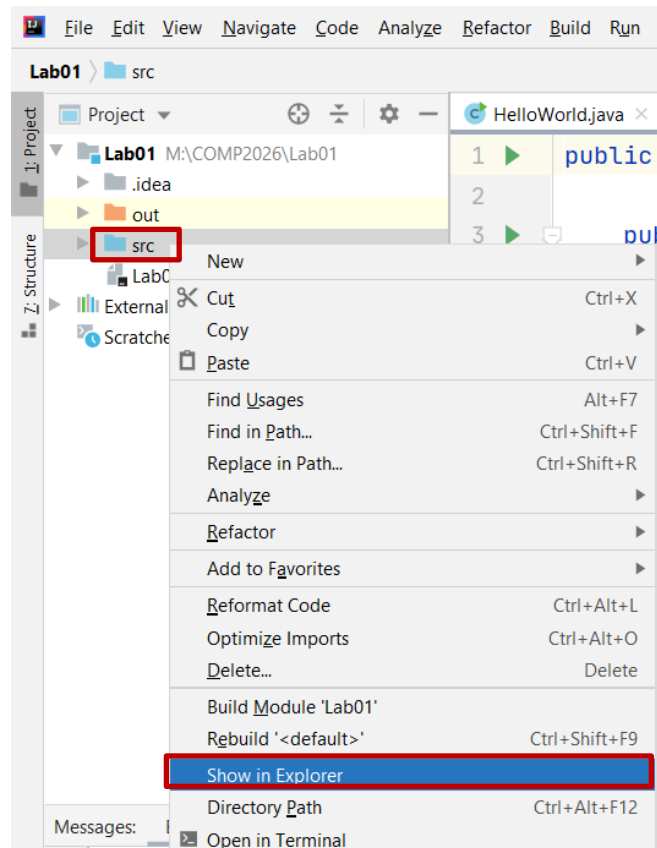


Step 2: Click **Refactor** to move the file into the **src** folder.



## 7. Where are my source files?

Right-click the **src** folder and select **Show in Explorer**.



## Part C Discovery Exercises

*\*Type your answers in XXXXXXXX\_lab01.docx*

### Task 1: Built-in Type

Execute the given `BuiltInType.java` program and fill in the following table.

Built-in Type	Size (in bits)	Range
byte		
short		
int		
long		
float		N/A
double		N/A

### Task 2: Escape Sequences

A character preceded by a backslash (\) is an **escape sequence** and has special meaning to the compiler. The following table shows the Java escape sequences:

Escape Sequence	Description
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a formfeed in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

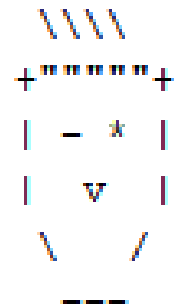
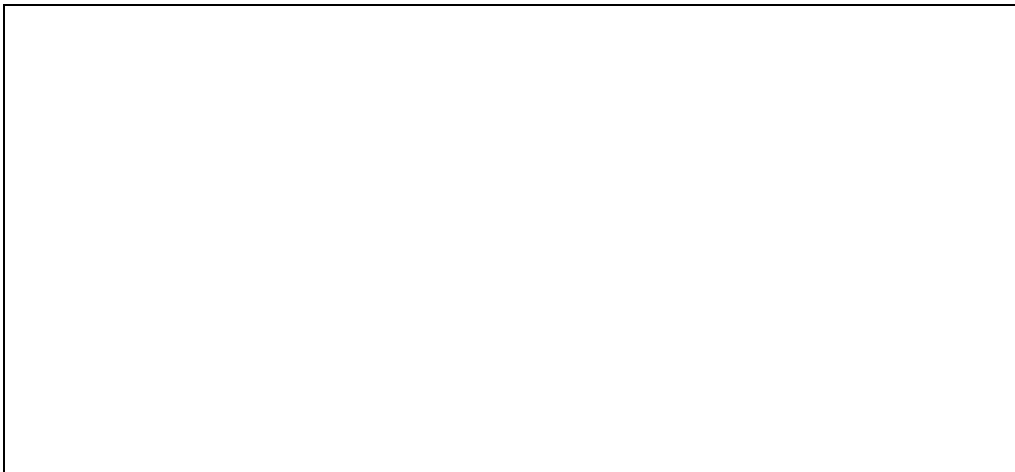
When an escape sequence is encountered in a print statement, the compiler interprets it accordingly. For example, the statement

```
System.out.println("Letter \"A\"");
```

prints

Letter "A"

Write the print statement(s) to print the pattern on the right.



### Task 3: Prefix and Postfix operators

When used in an assignment or print context, the prefix operator `++i` first increments `i` and then returns the value of `i`, whereas the postfix operator `i++` returns the value of `i` and then increments `i`.

```
...
int i = 3;
i++;           // i = 4
System.out.println(i); // prints 4
++i;          // i = 5
System.out.println(i); // prints 5
System.out.println(++i); // i = 6 and then prints 6
System.out.println(i++); // prints 6 and then i = 7
System.out.println(i);   // prints 7
...
```

Similarly, the prefix operator `--i` first decrements `i` and then returns the value of `i`, whereas the postfix operator `i--` returns the value of `i` and then decrements `i`.

```
...
int i = 3;
i--;           // i = 2
System.out.println(i); // prints 2
--i;          // i = 1
System.out.println(i); // prints 1
System.out.println(--i); // i = 0 and then prints 0
System.out.println(i--); // prints 0 and then i = -1
System.out.println(i);   // prints -1
...
```

After the following code executes, what are the values of **x** and **y**?

```
x = 6;  
y = ++x;
```

**x** = \_\_\_\_\_. **y** = \_\_\_\_\_.

After the following code executes, what are the values of **x** and **y**?

```
x = 6;  
y = x++;
```

**x** = \_\_\_\_\_. **y** = \_\_\_\_\_.

Fill in the blanks **with prefix or postfix operator** to produce the output specified.

```
...  
int i = 5;  
System.out.println(i++);           // prints 5  
System.out.println(  );           // prints 5  
System.out.println(  );           // prints 6  
System.out.println(  );           // prints 6  
System.out.println(  );           // prints 8  
System.out.println(  );           // prints 8  
System.out.println(  );           // prints 6  
System.out.println(  );           // prints 6  
System.out.println(i);             // prints 7  
...
```

#### Task 4: Formatted output

In addition to the **print()** and **println()** methods, the **printf()** method is also frequently used for displaying text that needs to be formatted. Formatting here refers to how data is to be displayed. For example, the number of decimal digits to be displayed, the alignment of the String, etc.

The **printf()** method requires a format String and a set of other arguments whose number depends on the format String. The format String contains specifier(s) that specifies the type of data that is going to be displayed and the formatting details. For example,

```
...  
double num1 = 10.0/3;  
double num2 = 10.0/6;  
  
System.out.println("The numbers are " + num1 + " and " + num2);  
System.out.printf("The numbers are %.2f and %.2f", num1, num2);  
...
```



prints

```
The numbers are 3.333333333333335 and 1.666666666666667
The numbers are 3.33 and 1.67
```

Format specifiers include flags, width, precision, and conversion characters in the following sequence:

**%[flags][width][.precision] conversion-character**

*Note: square brackets denote optional parameters*

**Flags:**

Flag	Description
–	Left-justify (default is to right-justify)
+	Output a plus (+) or minus (-) sign for a numeral value
,	Include locale-specific grouping characters
0	Forces numerical values to be zero-padded (default is blank padding)
	<u>Space</u> will display a minus sign if the number is negative or a space if it is positive

**Width** specifies the field width for outputting the argument and represents the *minimum* number of characters to be written to the output. Include space for expected commas and a decimal point in the determination of the width for numerical values.

**Precision** specifies the number of digits of precision when outputting floating-point values or the length of a substring to extract from a string. Numbers are rounded to the specified precision.

**Common Conversion-characters:**

Conversion-character	Description
c	Display a character
d	Display an integer number (base 10) [byte, short, int, long]
f	Display a floating-point number [float, double]
s	Display a string of characters

Examples:

Specifier	Result
<b>%8d</b>	Integer, right-aligned, 8-space-wide field
<b>%-6d</b>	Integer, left-aligned, 6-space-wide field
<b>%.2f</b>	Floating-point number, rounded to nearest hundredth
<b>%16.3f</b>	Floating-point number, rounded to nearest thousandth, 16-space-wide field
<b>%-9s</b>	String, left-aligned, 9-space-wide field

Complete the following `printf()` statements to print the specified output.

```
...
int n = 12345;
double pi = Math.PI;
String str = "Hello";

System.out.printf("[%d]\n", n);
System.out.printf(    );
System.out.printf(    );
System.out.printf(    );
System.out.printf(    );
System.out.printf(    );

System.out.printf("[%f]\n", pi);
System.out.printf(    );
System.out.printf(    );
System.out.printf(    );

System.out.printf("[%s]\n", str);
System.out.printf(    );
System.out.printf(    );

...
```

prints

```
[12345]
[   12345]
[12345   ]
[000012345]
[   12,345]
[  +12,345]
[3.141593]
[3.142]
[   3.142]
[  +3.142]
[Hello]
[   Hello]
[Hello   ]
```

## Task 5: ASCII Code

Computer store data in binary format. Every piece of information, including characters, numbers, and even program instructions, is stored as a sequence of 0's and 1's. For example, a lowercase 'a' is represented by 1100001 (97 in decimal) and a 'b' is encoded as 1100010 (98 in decimal). This encoding is used to identify a character's ASCII code (American Standard Code for Information Interchange). Every character that appears on your keyboard has its own 7-bit ASCII sequence or code.

# ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	\$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	.	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

Image source: <https://qph.fs.quoracdn.net/main-qimg-40eb504e356af0832441ef95091eba8e>

The following code fragment can print the decimal ASCII value of a given character, e.g. 'a'.

```
char c = 'a';
System.out.println((int)c);
```

The following code fragment can print the character representation of a given decimal ASCII value.

```
int asciiCode = 97;
System.out.println((char)asciiCode);
```

Using the ASCII Table above, answer the following questions.

Fill in the following table.

Letter	Lowercase ASCII Value	Uppercase ASCII Value	Lowercase ASCII Value – Uppercase ASCII Value
A	97	65	
B			
C			
D			

What is the different between the decimal ASCII values of lowercase letter and uppercase letter?

Ans: \_\_\_\_\_.

What is the range of the decimal ASCII values of lowercase letter a to z?

Ans: \_\_\_\_\_.

What is the range of the decimal ASCII values of uppercase letter A to Z?

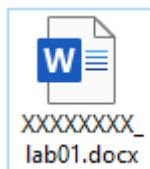
Ans: \_\_\_\_\_.

What is the range of the decimal ASCII values of digit 0 to 9?

Ans: \_\_\_\_\_.

## Part D Submitting Exercises

Submit **XXXXXXXX\_lab01.docx** to Moodle where **XXXXXXXX** is your **Student ID number**.



## References

- [1] Bravaco, R., & Simonson, C. (2009). *Java programming: From the ground up*. Dubuque, IA: McGraw-Hill.
- [2] Dean, J., & Dean, R. (2008). *Introduction to programming with Java: A problem solving approach*. Boston: McGraw-Hill.
- [3] Farrell, J. (2012). *Java programming*. Boston, MA: Course Technology Cengage Learning
- [4] Forouzan, B. A., & Gilberg, R. F. (2007). *Computer science: A structured programming approach using C (3rd ed.)*. Boston, MA: Thomson Course Technology.
- [5] Gaddis, T. (2016). *Starting out with Java (6th ed.)*. Pearson.
- [6] Liang, Y. D. (2013). *Introduction to Java programming: Comprehensive version*. (8<sup>th</sup> ed.). Pearson.
- [7] Schildt, H. ( 2006). *Java a beginner's guide*. New York: McGraw Hill.
- [8] Schildt, H., & Skrien, D. J. (2013). *Java programming: A comprehensive introduction*. New York: McGraw-Hill.
- [9] Wu, C. T. (2010). *An introduction to object-oriented programming with Java*. Boston: McGraw Hill Higher Education
- [10] Xavier, C. (2011). *Java programming: A practical approach*. New Delhi: Tata McGraw Hill.
- [11] yet another insignificant Programming Notes. (n.d.). Retrieved from <https://www3.ntu.edu.sg/home/ehchua/programming>
- [12] Zakhour, S., Kannan, S., & Gallardo, R. (2013). *The Java tutorial: A short course on the basics (5th ed.)*.