

Instructions if any for students: Students have to try the ten questions and submit back in a word doc with the Roll No,Name ,Program,Output

Lab #1

- At the end of the lab students will be able to write java programs with basic datatypes and operators
- Students will have to submit the programs with output

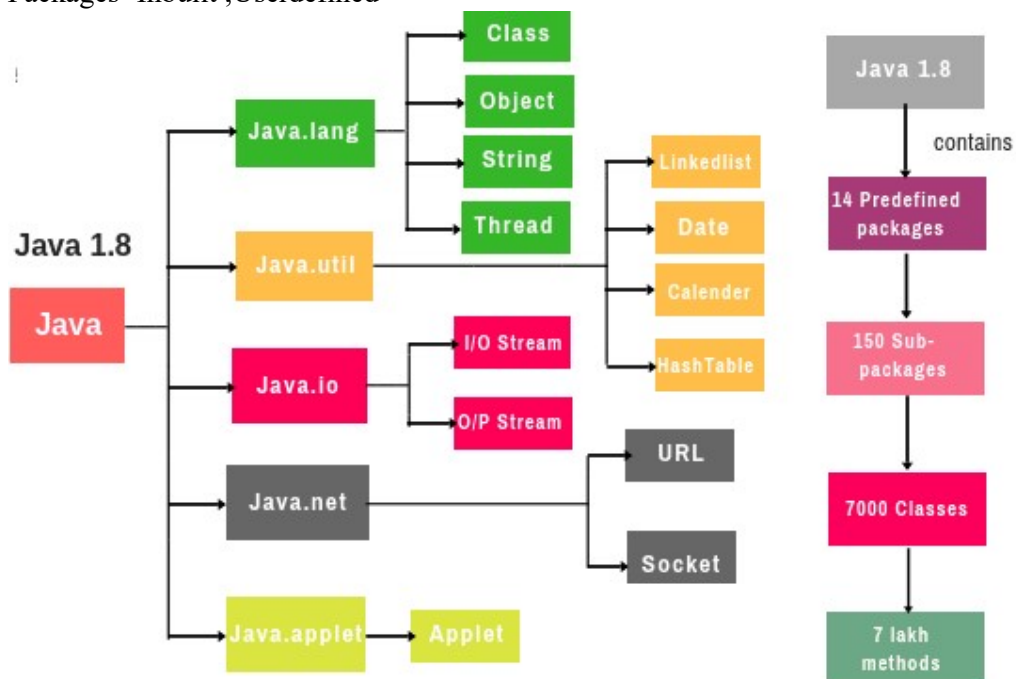
Topics to be covered

- Java Program structure
- Data Types
- Operators

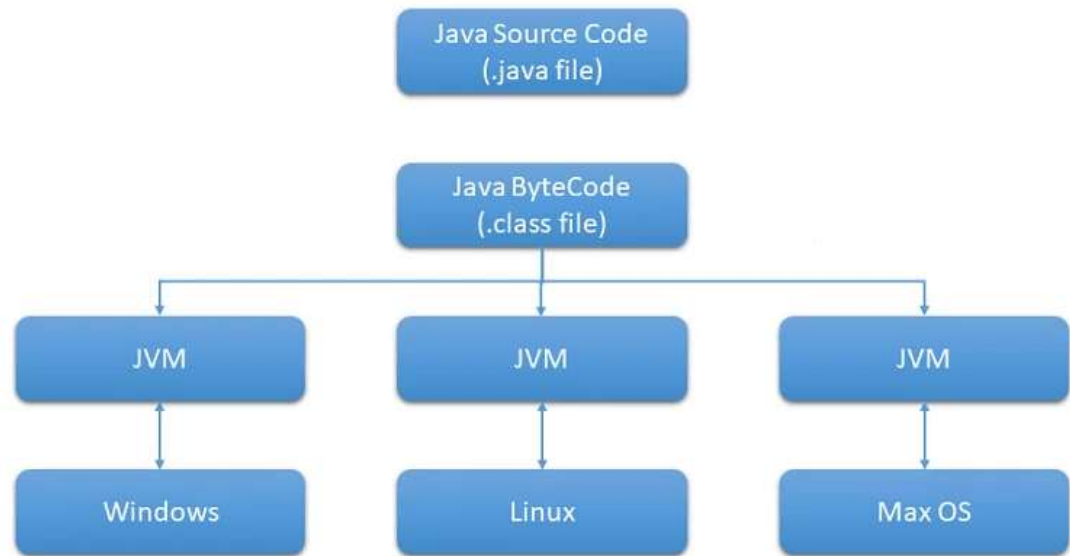
Java Programming Concepts

Features

1. Class Based
2. Object Oriented
3. Inheritance – Multiple Inheritance is replaced with interfaces
4. Hierarchy - Library, package, Class-Variables,Methods
5. Packages -Inbuilt ,Userdefined



6. Platform Independent



7. Running a Java program

a. Write the Java program

b. Place the comments within

```
/* Welcome to my first program*/
```

Or

```
//
```

Ex: //Welcome to my first program

Or

```
/** This class displays a text string at  
 * the console.  
 */
```

c. Compile the Java Program

Javac **FileName.java**

d. Run the Java Program

Filename should match the name of the class(Welcome.java)

General Syntax:

Import **packagename**;

Class **classname**

{

public static void main(String[] args){

Statements;

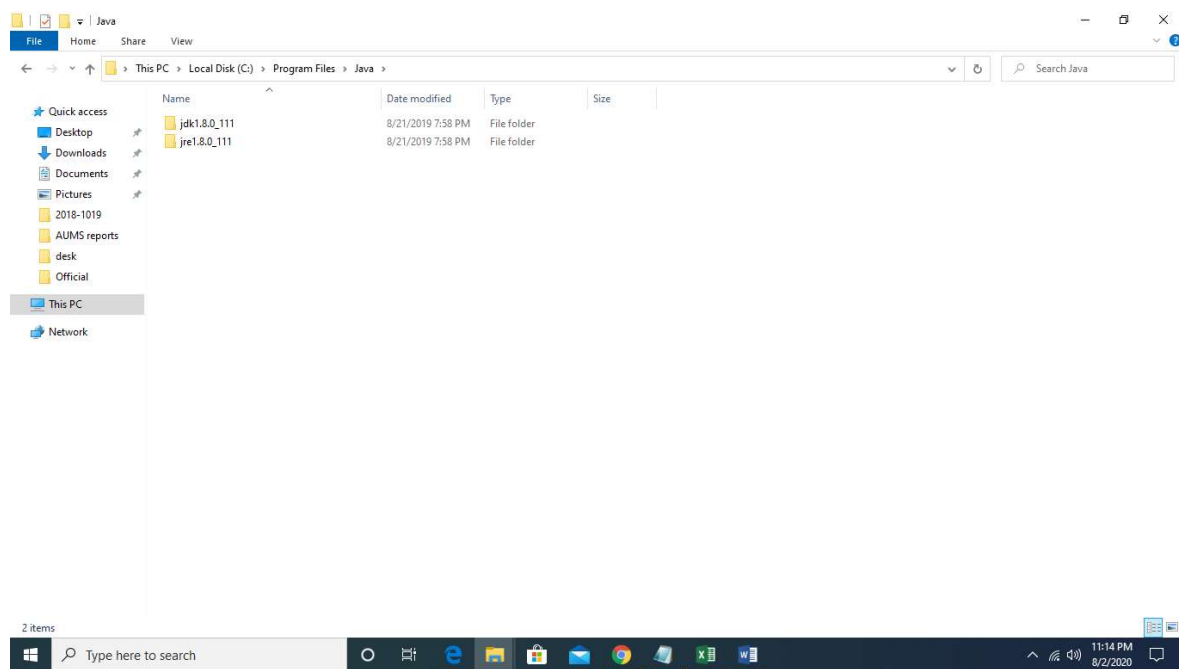
```
}
```

```
}
```

Example :

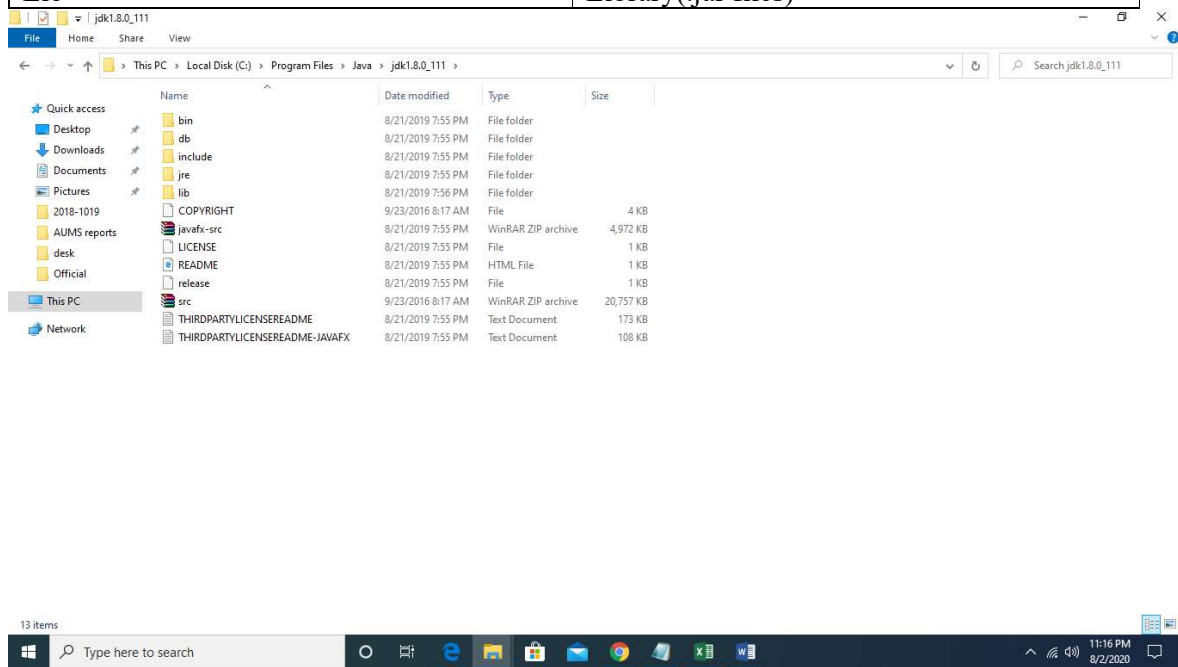
```
//Welcome Message  
class Welcome {  
    public static void main(String[] args){  
        System.out.println("I'm a Simple Program");  
    }  
}
```

Once Java is installed in system it will be available under Program Files:



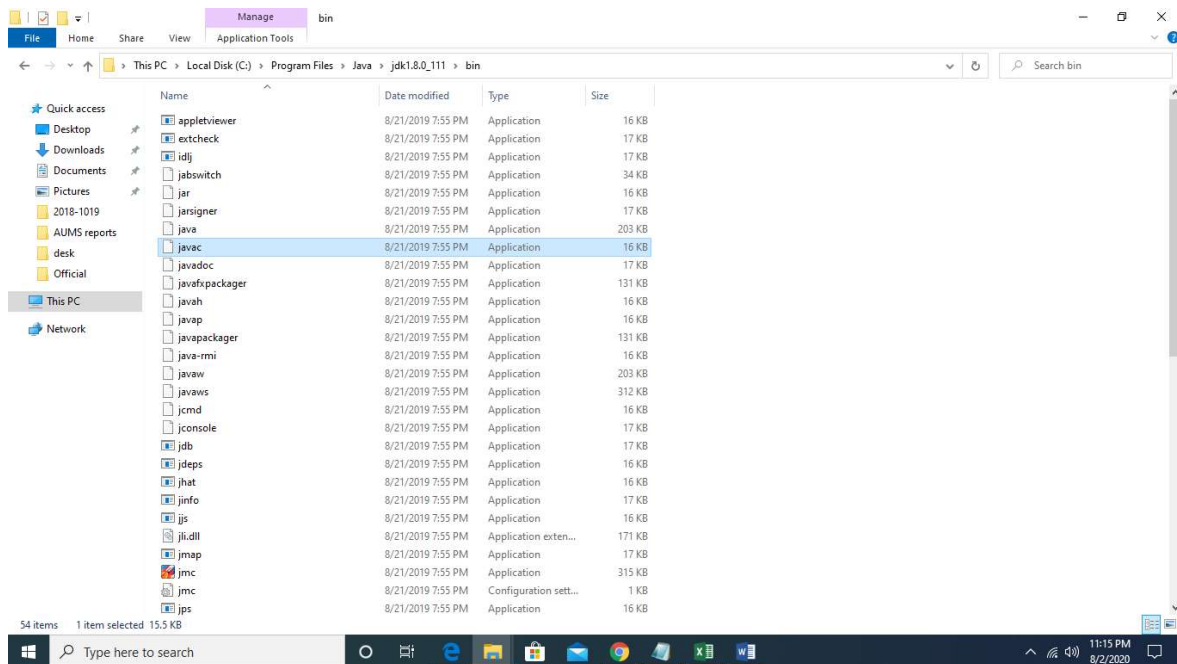
The folders have specific purpose:

Folder Name	Description
Bin	It contains the jar files
Db	Networking support
Include	Header files(.h files)
Jre	Java Runtime environment
Lib	Library(.jar files)

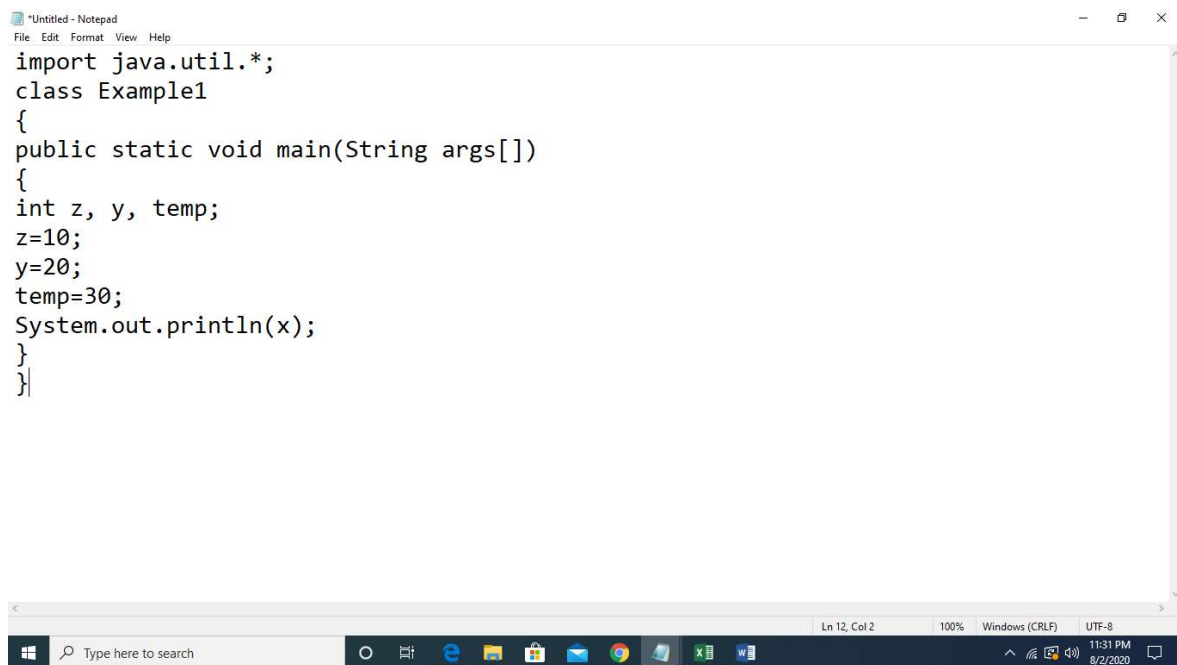


Folder Details

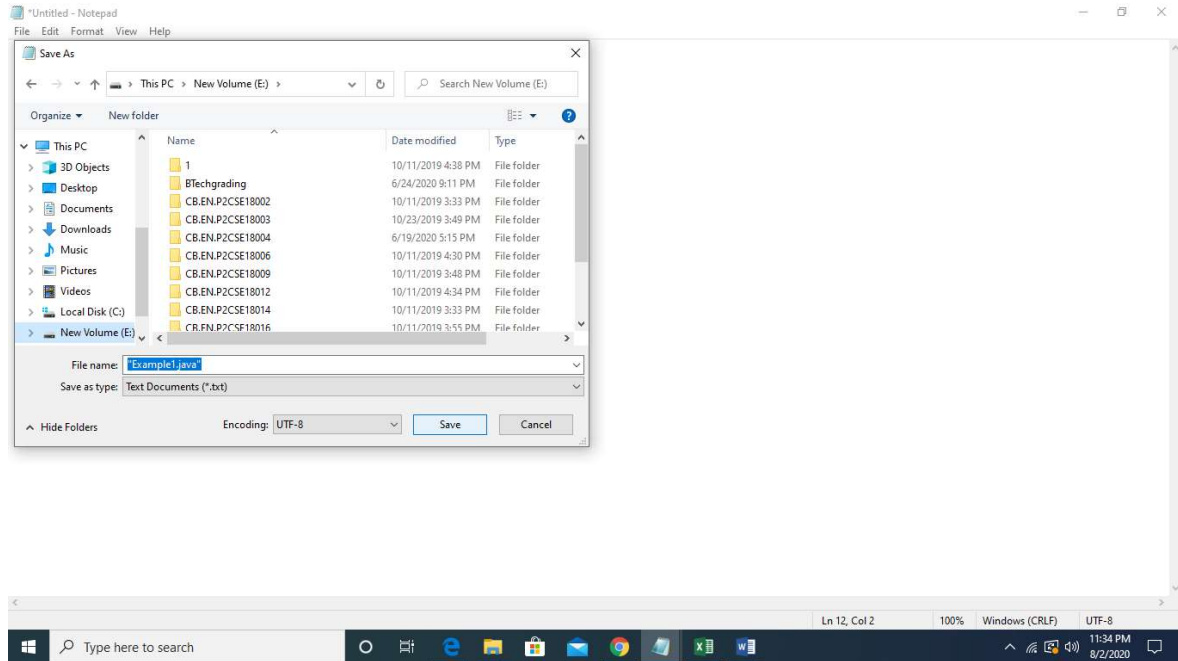
Screenshot of the bin folder



Let us try an example:



Save the file with extension java



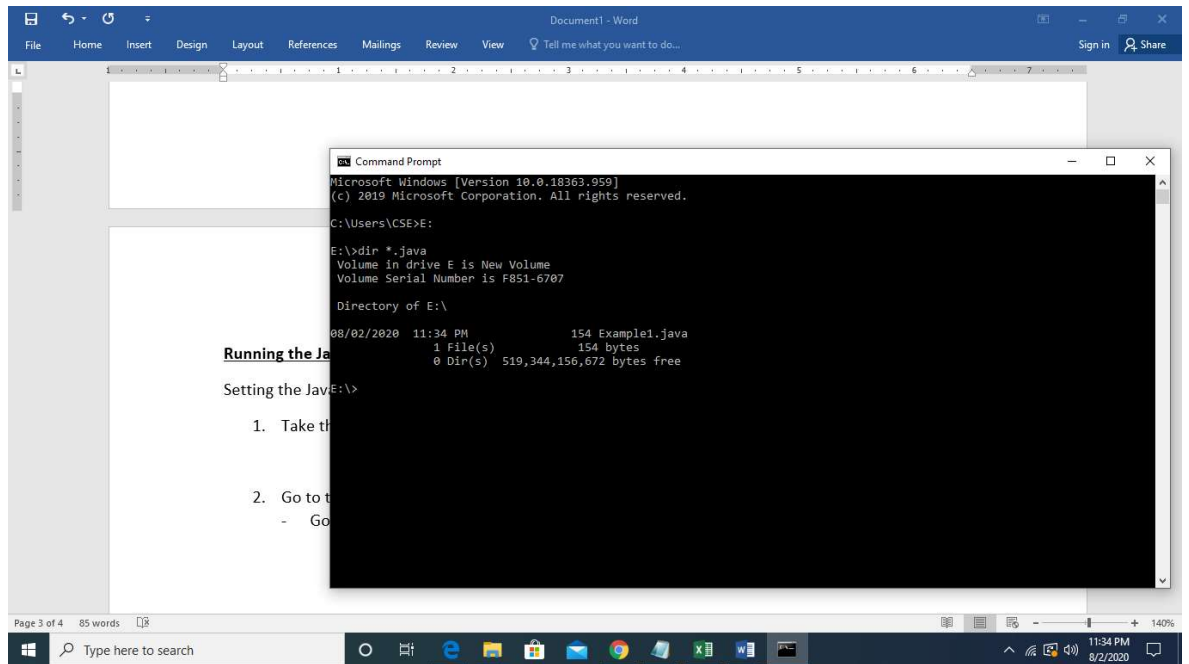
Running the Java Program

Setting the Java home path

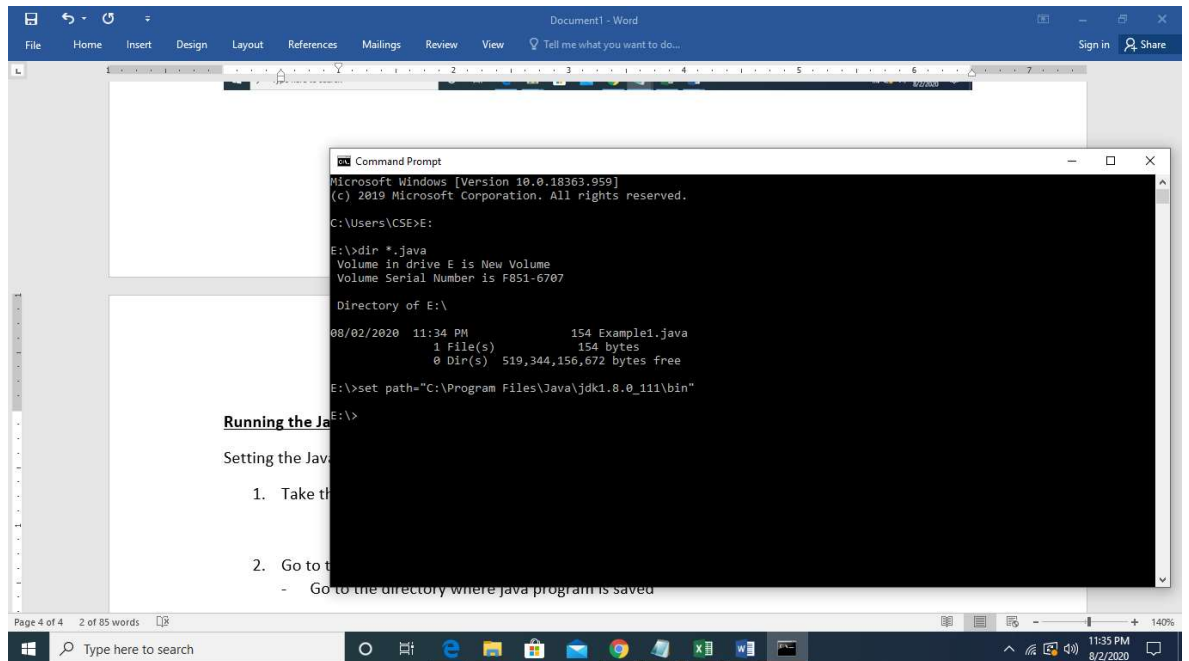
1. Take the path of the bin folder

C:\Program Files\Java\jdk1.8.0_111\bin

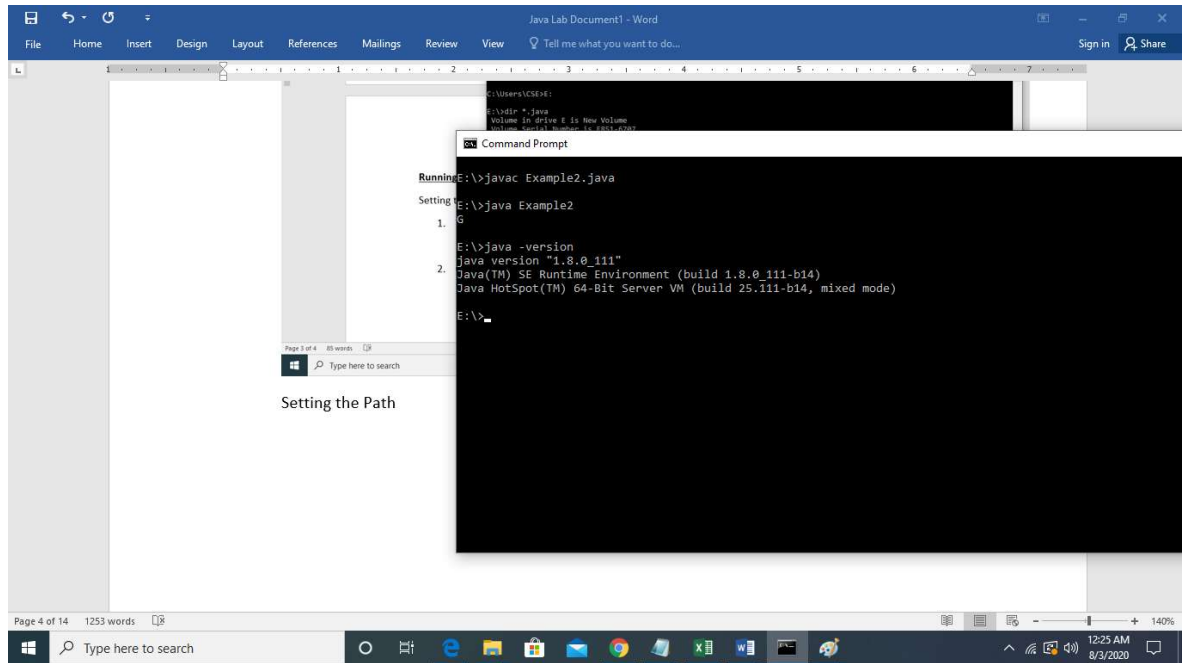
2. Go to the command prompt
 - Go to the directory where java program is saved



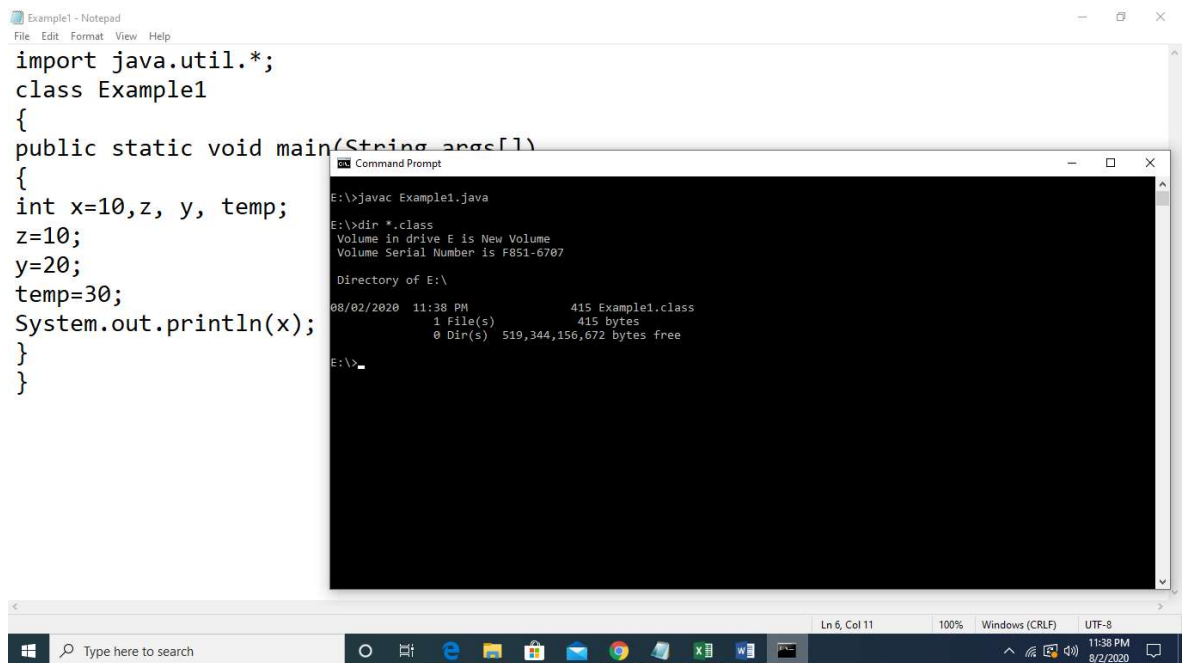
Setting the Path



Checking java version



Compiling the Java program

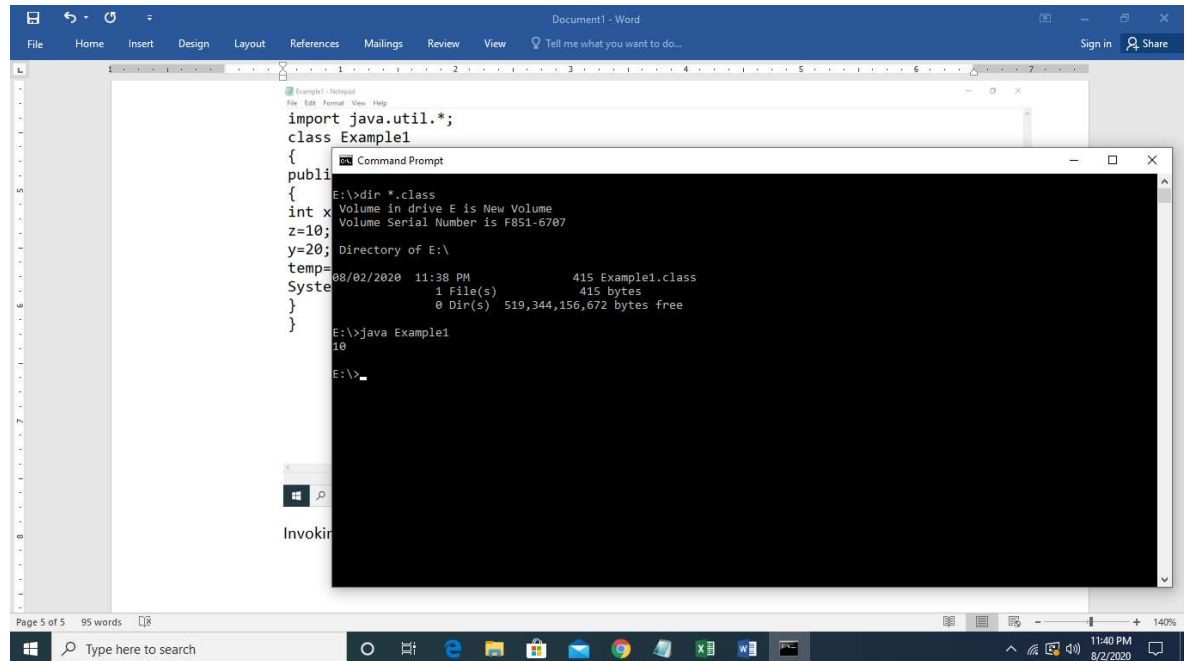


It creates a class file with name as the name of the class in program.

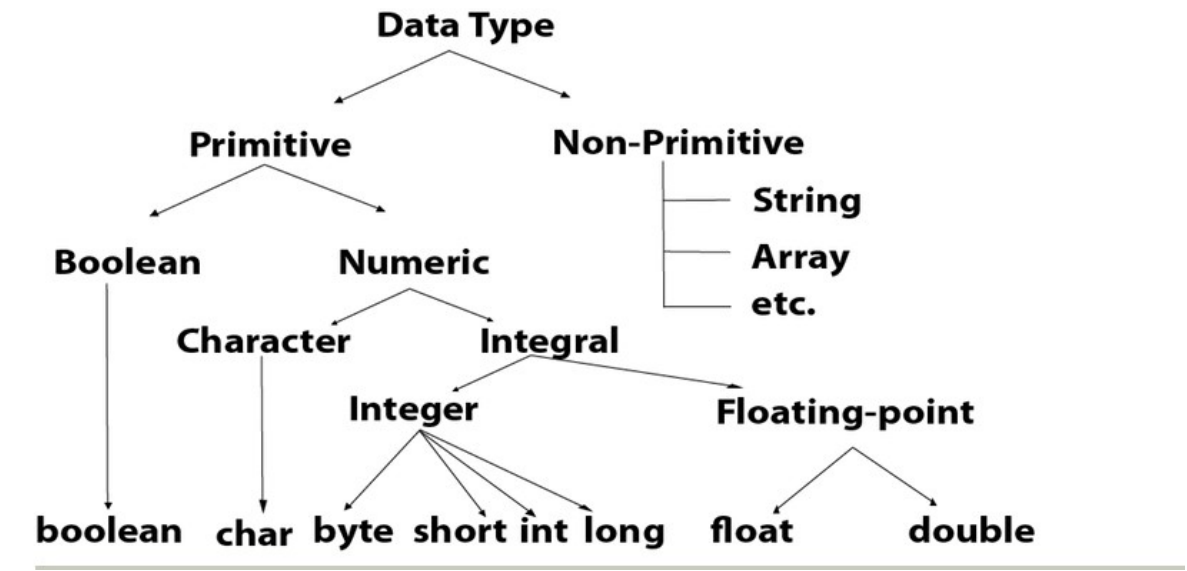
A Java class file is a file containing Java bytecode that can be executed on the Java Virtual Machine. A Java class file is usually produced by a Java compiler from Java programming language source files containing Java classes.

A class file consists of a stream of 8-bit bytes. All 16-bit, 32-bit, and 64-bit quantities are constructed by reading in two, four, and eight consecutive 8-bit bytes, respectively. Multibyte data items are always stored in big-endian order, where the high bytes come first.

Class files are input for a **JVM(Java Virtual Machine)** which will get interpreted by Just In Time Compiler to convert bytecode into machine code and execute it on Operating System to achieve functionality implemented in Java source code.



Java Datatypes



The following chart summarizes the default values for the above data types.

Data Type	Default Value (for fields)
Byte	0
Short	0
Int	0
Long	0L
Float	0.0f
Double	0.0d
Char	'\u0000'
String (or any object)	Null
boolean	False

Variables

1. Instance Variables
2. Class Variables
3. Local Variables
4. Parameters

Local variables are slightly different; the compiler never assigns a default value to an uninitialized local variable.

A *literal* is the source code representation of a fixed value; literals are represented directly in your code without requiring computation. As shown below, it's possible to assign a literal to a variable of a primitive type:

```
boolean result = true;
char capitalC = 'C';
byte b = 100;
short s = 10000;
int i = 100000;
```

There are eight primitive data types in Java. These are as follows:

1. Byte: A byte, is made up of 8 individual bits. Byte data types in Java have the following characteristics:

- **Minimum Value:** -128 (2^7)
- **Maximum Value:** 127 (2^7-1)
- **Default Value:** 0

Examples:

```
byte x = 56
```

```
byte y = 68
```

2. Short: A short is twice the size of a byte, i.e. it is made up of 16-bits. Its chief characteristics are:

- **Minimum Value:** -32,768 (2^{15})
- **Maximum Value:** 32,767 ($2^{15}-1$)
- **Default Value:** 0

Examples:

```
short x = 9870
```

```
short y = -635
```

Like bytes, short types are useful alternatives to int (see below) data types, particularly if your data falls within the specified range. As with byte, using short also improves code readability, besides saving memory.

3. Int: An integer is four times the size of a byte (i.e. it is made up of 32 bits). It is one of the most commonly used data types in Java.

- **Minimum Value:** -2,147,483,648 (2^{31})
- **Maximum Value:** 2,147,483,647 ($2^{31} - 1$)
- **Default Value:** 0

Examples:

```
int x = 150000
```

```
int y = -2004320
```

As the most easily understood data type, you will use int a lot in your code.

4. Long: A long data type is twice the size of an integer, i.e. it is made up of 64-bits. It's chief characteristics are:

- **Minimum Value:** -9,223,372,036,854,775,808 (2^{63})
- **Maximum Value:** 9,223,372,036,854,775,807 ($2^{63} - 1$)
- **Default Value:** 0

Examples:

```
long x = 6778005876543
```

```
long y = -554233254242
```

You'll use long only if you encounter data that doesn't fit within the int range (which will be rare).

5. Float: In programming, any decimal or fractional value is called a 'float'. If there is a decimal after the number, it will be classified as a float. In Java, a float is made up of 32-bits IEEE floating points*.

The minimum/maximum value of float is not the same as that of the int data type (despite both being made of 32-bits). The full range of float values is beyond the scope of this tutorial. For now, the only thing you need to know is that you'll use float (and double – see below) for saving decimal values.

Examples:

```
float x = 2.321
```

```
float y = 1.234
```

*The float value range depends on the IEEE standard classification for floating point numbers. You can read about it [here](#).

6. Double: Double is a data type that is twice the size of a float. I.e. it is made up of 64-bit IEEE floating points.

As with float, discussing the minimum/maximum value of double data type is beyond the scope of this article. What you should know is that double is a much more precise type than float. For all practical purposes, it is recommended that you use double instead of float for storing decimal values.

Examples:

```
double a = 1.245240
```

```
double y = 12.2232
```

7. Char: Char data type refers to a single 16-bit Unicode character. Unicode is a computer industry standard for representing text related data. This includes alphabets, symbols (\$, &, *, #, @, !, etc.), and special figures such as ¢, £, ¥, etc. The Unicode character set includes over 110,000 characters covering more than 100 language scripts.

In other words, any data besides numbers goes into the char data type.

Examples:

```
char name = 'J'
```

```
char country = 'U'
```

8. Boolean: Boolean is the smallest data type in Java, i.e. it is made up of only one bit. Thus, a Boolean data type can have only two values – 0 (or False) and 1 (or True).

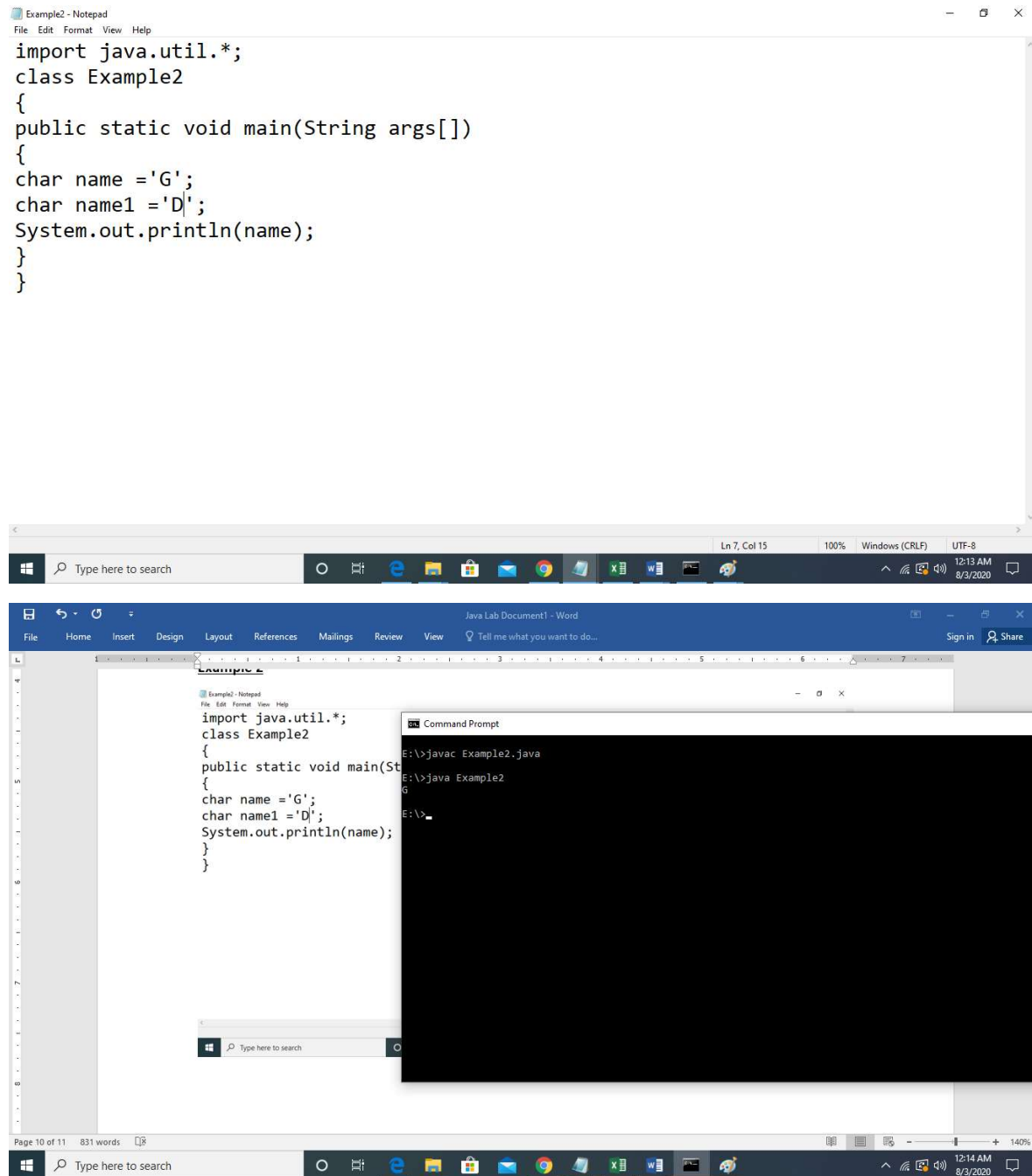
Example:

```
boolean x = true
```

```
boolean y = false
```

(**Tip:** 'True' and 'False' written above are **not** strings. Do not enclose them within quotes as we did with the char example above)

Example 2



Try it yourself and place the output:

Q.1.

Program 1:

```
public class CharExample2 {
    public static void main(String[] args) {
        char char1=65;
```

```
char char2=97;
```

```
System.out.println("char1: "+char1);
```

```
System.out.println("char2: "+char2);
```

```
}
```

```
}
```

Output

Q.2.

```
public class CharExample3 {
```

```
    public static void main(String[] args) {
```

```
        int num1=97;
```

```
        char char1=(char)num1;
```

```
        int num2=65;
```

```
        char char2=(char)num2;
```

```
        System.out.println("char1: "+char1);
```

```
        System.out.println("char2: "+char2);
```

```
    }
```

```
}
```

Output

Q3. Example for Unicode format

```
public class CharExample4 {  
    public static void main(String[] args) {  
  
        char char1='\u0061';  
        char char2='\u0041';  
  
        System.out.println("char1: "+char1);  
        System.out.println("char2: "+char2);  
  
    }  
}
```

Output

Self Study concept

Try it yourself

Using Underscore Characters in Numeric Literals

In Java SE 7 and later, any number of underscore characters (_) can appear anywhere between digits in a numerical literal. This feature enables you, for example, to separate groups of digits in numeric literals, which can improve the readability of your code.

For instance, if your code contains numbers with many digits, you can use an underscore character to separate digits in groups of three, similar to how you would use a punctuation mark like a comma, or a space, as a separator.

The following example shows other ways you can use the underscore in numeric literals:

```
long creditCardNumber = 1234_5678_9012_3456L;  
long socialSecurityNumber = 999_99_9999L;  
float pi = 3.14_15F;  
long hexBytes = 0xFF_EC_DE_5E;  
long hexWords = 0xCAFE_BABE;  
long maxLong = 0x7fff_ffff_ffff_ffffL;
```



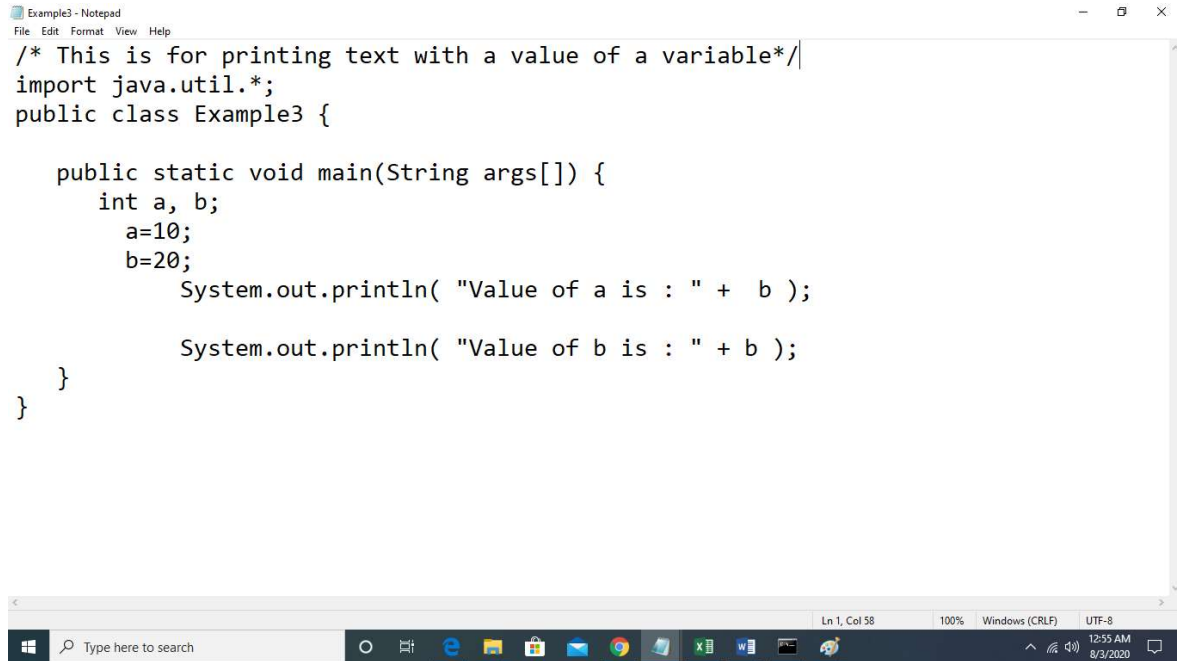
```
byte nybbles = 0b0010_0101;  
long bytes = 0b11010010_01101001_10010100_10010010;
```

You can place underscores only between digits; you cannot place underscores in the following places:

- At the beginning or end of a number
- Adjacent to a decimal point in a floating point literal
- Prior to an F or L suffix
- In positions where a string of digits is expected

The following examples demonstrate valid and invalid underscore placements (which are highlighted) in numeric literals:

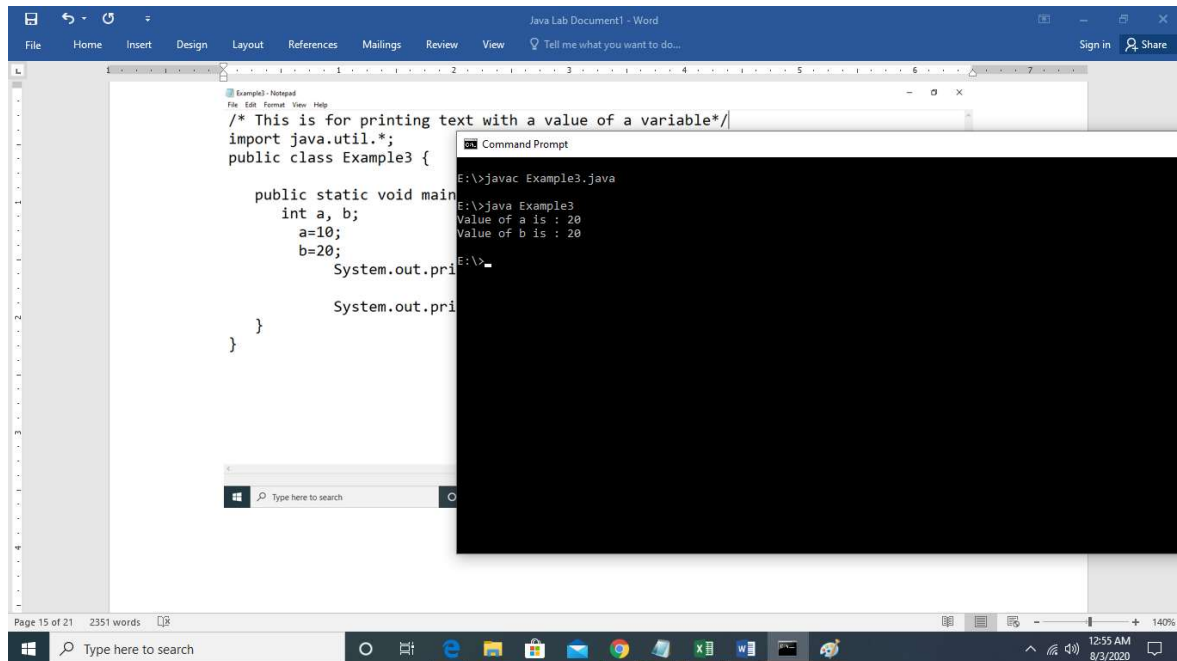
```
// Invalid: cannot put underscores  
// adjacent to a decimal point  
float pi1 = 3_.1415F;  
// Invalid: cannot put underscores  
// adjacent to a decimal point  
float pi2 = 3._1415F;  
// Invalid: cannot put underscores  
// prior to an L suffix  
long socialSecurityNumber1 = 999_99_9999_L;  
  
// OK (decimal literal)  
int x1 = 5_2;  
// Invalid: cannot put underscores  
// At the end of a literal  
int x2 = 52_;  
// OK (decimal literal)  
int x3 = 5_____2;  
  
// Invalid: cannot put underscores  
// in the 0x radix prefix  
int x4 = 0_x52;  
// Invalid: cannot put underscores  
// at the beginning of a number  
int x5 = 0x_52;  
// OK (hexadecimal literal)  
int x6 = 0x5_2;  
// Invalid: cannot put underscores  
// at the end of a number  
int x7 = 0x52_;
```



```
Example3 - Notepad
File Edit Format View Help
/* This is for printing text with a value of a variable*/
import java.util.*;
public class Example3 {

    public static void main(String args[]) {
        int a, b;
        a=10;
        b=20;
        System.out.println( "Value of a is : " + b );

        System.out.println( "Value of b is : " + b );
    }
}
```



```
Java Lab Document1 - Word
File Home Insert Design Layout References Mailings Review View Tell me what you want to do... Sign in Share

Example3 - Notepad
File Edit Format View Help
/* This is for printing text with a value of a variable*/
import java.util.*;
public class Example3 {

    public static void main
        int a, b;
        a=10;
        b=20;
        System.out.prin
        System.out.prin
    }
}

Command Prompt
E:\>javac Example3.java
E:\>java Example3
Value of a is : 20
Value of b is : 20
E:\>
```

Q4. Write a program to print the following details:

(use only variables. Do not use any inbuilt methods. Provide value for the fields in the program)

Roll No

Name

Address
Emailid
Mobile No
Parents Occupation
Todays Date

Q5. Write a program to print the following details:

(use only variables.Do not use any inbuilt methods)

RollNo
Name
English
Physics
Chemistry
Average
Grade

Operators

a. Arithmetic Operators

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator.	A + B will give 30
- (Subtraction)	Subtracts right-hand operand from left-hand operand.	A - B will give -10
* (Multiplication)	Multiplies values on either side of the operator.	A * B will give 200
/ (Division)	Divides left-hand operand by right-hand operand.	B / A will give 2

% (Modulus)	Divides left-hand operand by right-hand operand and returns remainder.	B % A will give 0
++ (Increment)	Increases the value of operand by 1.	B++ gives 21
-- (Decrement)	Decreases the value of operand by 1.	B-- gives 19

b. Relational

Operator	Description	Example
== (equal to)	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!= (not equal to)	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
> (greater than)	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
< (less than)	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>= (greater than or equal to)	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<= (less than or equal to)	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

c. Bitwise

a = 0011 1100

b = 0000 1101

Operator	Description	Example
& (bitwise and)	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
(bitwise or)	Binary OR Operator copies a bit if it exists in either operand.	(A B) will give 61 which is 0011 1101
^ (bitwise XOR)	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~ (bitwise compliment)	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number.
<< (left shift)	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>> (right shift)	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 1111
>>> (zero fill right shift)	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	A >>>2 will give 15 which is 0000 1111

d.Logical Operators

Operator	Description	Example
&& (logical and)	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false

(logical or)	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	(A B) is true
! (logical not)	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true

e. Assignment Operators

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator. It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A

<code>%=</code>	Modulus AND assignment operator. It takes modulus using two operands and assign the result to left operand.	<code>C %= A</code> is equivalent to <code>C = C % A</code>
<code><<=</code>	Left shift AND assignment operator.	<code>C <<= 2</code> is same as <code>C = C << 2</code>
<code>>>=</code>	Right shift AND assignment operator.	<code>C >>= 2</code> is same as <code>C = C >> 2</code>
<code>&=</code>	Bitwise AND assignment operator.	<code>C &= 2</code> is same as <code>C = C & 2</code>
<code>^=</code>	bitwise exclusive OR and assignment operator.	<code>C ^= 2</code> is same as <code>C = C ^ 2</code>
<code> =</code>	bitwise inclusive OR and assignment operator.	<code>C = 2</code> is same as <code>C = C 2</code>

f.Conditional Operator

Conditional operator is also known as the **ternary operator**. This operator consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide, which value should be assigned to the variable.

variable x = (expression) ? value if true : value if false

Try it yourself

Q6.

```
public class Test {  
    public static void main(String args[]) {  
        int a, b;  
        a = 10;  
        b = (a == 1) ? 20: 30;  
        System.out.println( "Value of b is : " + b );  
        b = (a == 10) ? 20: 30;  
        System.out.println( "Value of b is : " + b );  
    }  
}
```

Output:

Q7.

Given principal as P, No of Years as N, Rate of interest as R. Calculate

$$\text{Interest} = (P * N * R) / 100$$

Q8 . Write a program to declare numbers of all possible numeric datatypes . Assign a value to the variable and print the same.

Q9. Write a program to print following details:

Product id

Product Name

Product Description

Product Price

Model No

Q10. Write a program to have two numbers. Perform the sum and difference of two numbers.

References

1. <https://www.javatpoint.com/how-to-compile-and-run-java-program>
2. <https://www.oracle.com/java/technologies/compile.html>
3. <https://examples.javacodegeeks.com/core-java/java-class-file-example/>
4. <https://www.javatpoint.com/>
- 5.