



National University of Computer & Emerging Sciences,  
Karachi



Computer Science Department  
Spring 2022, Lab Manual – 01

Course Code: CL-1004	Course : Object Oriented Programming Lab
Instructor(s) :	Abeer Gauher, Hajra Ahmed, Syed Zain ul Hassan

## **LAB - 1**

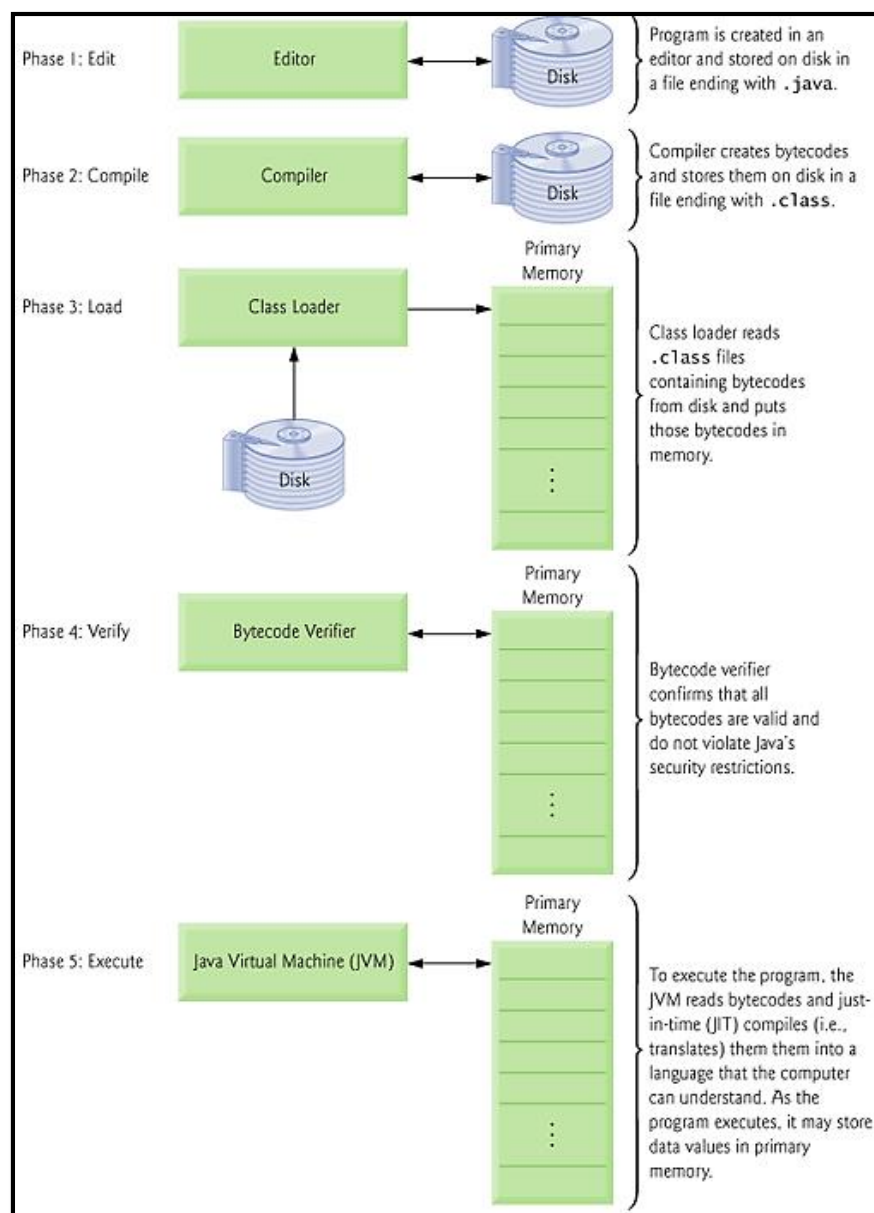
# **INTRODUCTION TO JAVA**

# JAVA

Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. James Gosling initiated the Java language project in June 1991.

Java can be used to create two types of programs: applications and applets. An application is a program that runs on your computer, under the operating system of that computer. An applet is an application designed to be transmitted over the Internet and executed by a Java compatible Web browser.

## TYPICAL JAVA DEVELOPMENT ENVIRONMENT



### **Phase 1: Creating a Program**

Phase 1 consists of editing a file with an editor program. You type a Java program (typically referred to as source code) using the editor, make any necessary corrections and save the program. A file name ending with the .java extension indicates that the file contains java source code.



### **Phase 2: Compiling a Java Program into Bytecodes**

The Java compiler compiles a program. If the program compiles successfully, the compiler produces a .class file. The Java compiler translates Java source code into bytecodes. Bytecodes are executed by the Java Virtual Machine (JVM).



### **Phase 3: Loading a Program into Memory**

The JVM places the program in memory to execute it—this is known as loading. The JVM's class loader takes the .class files containing the program's bytecodes and transfers them to primary memory.



### **Phase 4: Bytecode Verification**

The classes are loaded, the bytecode verifier examines their bytecodes to ensure that they're valid and do not violate Java's security restrictions.



### **Phase 5: Execution**

The JVM executes the program's bytecodes, thus performing the actions specified by the program.

## Difference between JDK, JRE, and JVM

### **JVM**

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed.

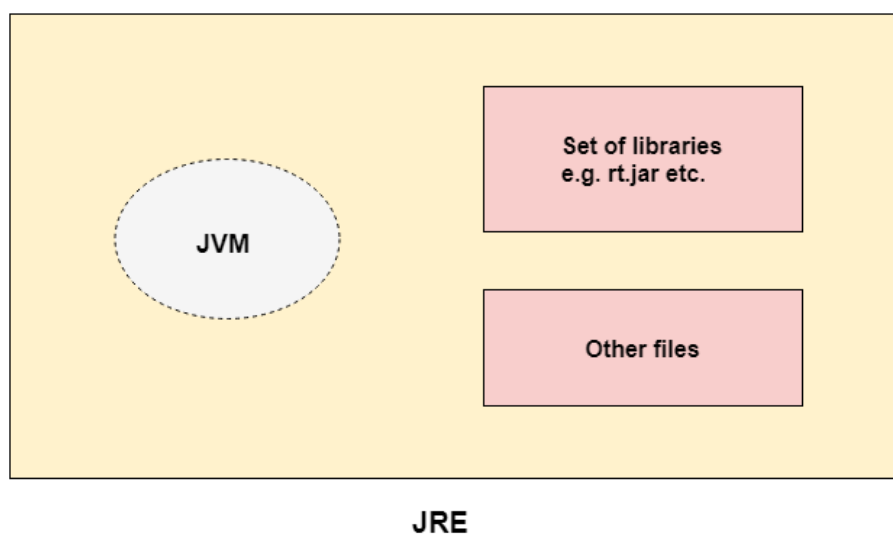
JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each OS is different from each other. However, Java is platform independent.

The JVM performs the following main tasks:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

### **JRE**

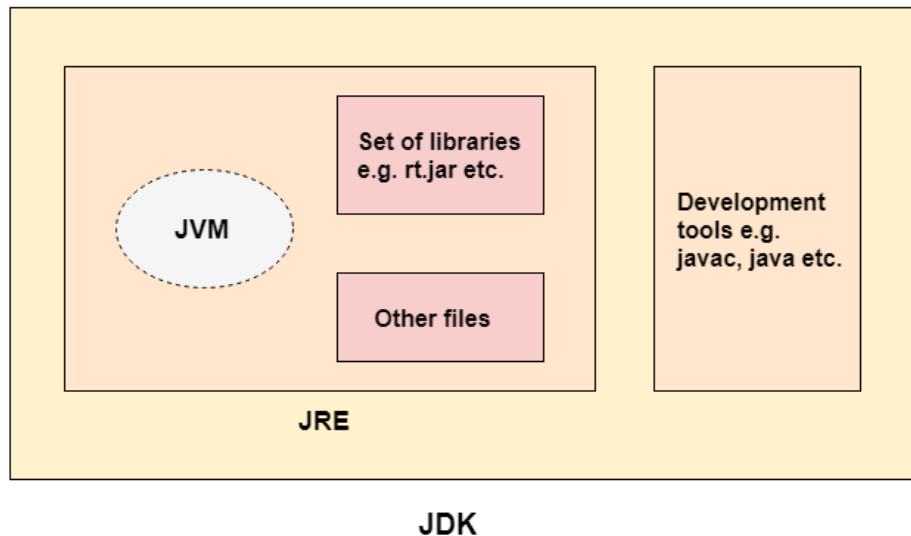
JRE is an acronym for Java Runtime Environment. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.



### **JDK**

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.



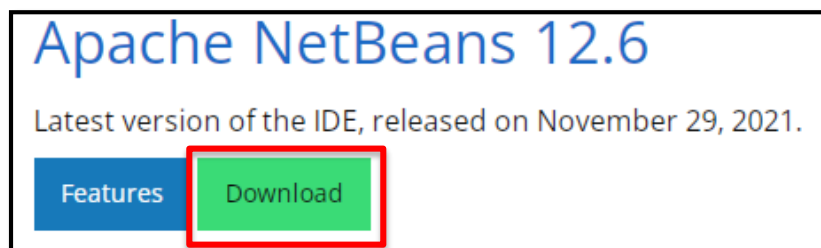
### IDE:

NetBeans is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, macOS, and Linux. In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML5, and JavaScript.

### INSTALLATION GUIDE TO NETBEANS:

To download the latest version of netbeans, use the following link:

<https://netbeans.apache.org/download/index.html>



Click on Download and then select the desired option based on your OS:

- [Apache-NetBeans-12.6-bin-windows-x64.exe \(SHA-512, PGP ASC\)](#)
- [Apache-NetBeans-12.6-bin-linux-x64.sh \(SHA-512, PGP ASC\)](#)
- [Apache-NetBeans-12.6-bin-macosx.dmg \(SHA-512, PGP ASC\)](#)

If Windows is selected, you will be redirected to another page. Click on the link to download the .exe file.

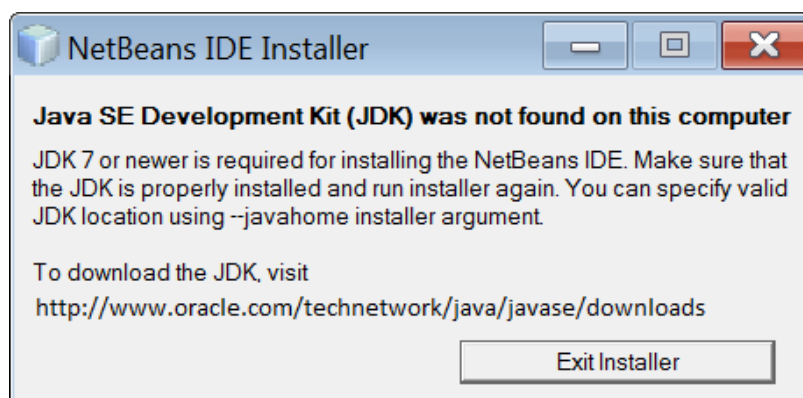
We suggest the following site for your download:

<https://dlcdn.apache.org/netbeans/netbeans-installers/12.6/Apache-NetBeans-12.6-bin-windows-x64.exe>

Run the .exe file to setup Netbeans on your systems.

If incase, you are prompted that JDK is not found on your computer, please visit the following link to download it on your computer.

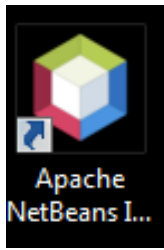
<https://www.oracle.com/java/technologies/downloads/#jdk17-windows>



Download the Installer file based on your OS. Run the .exe file to setup JDK on your computer and then install NetBeans.

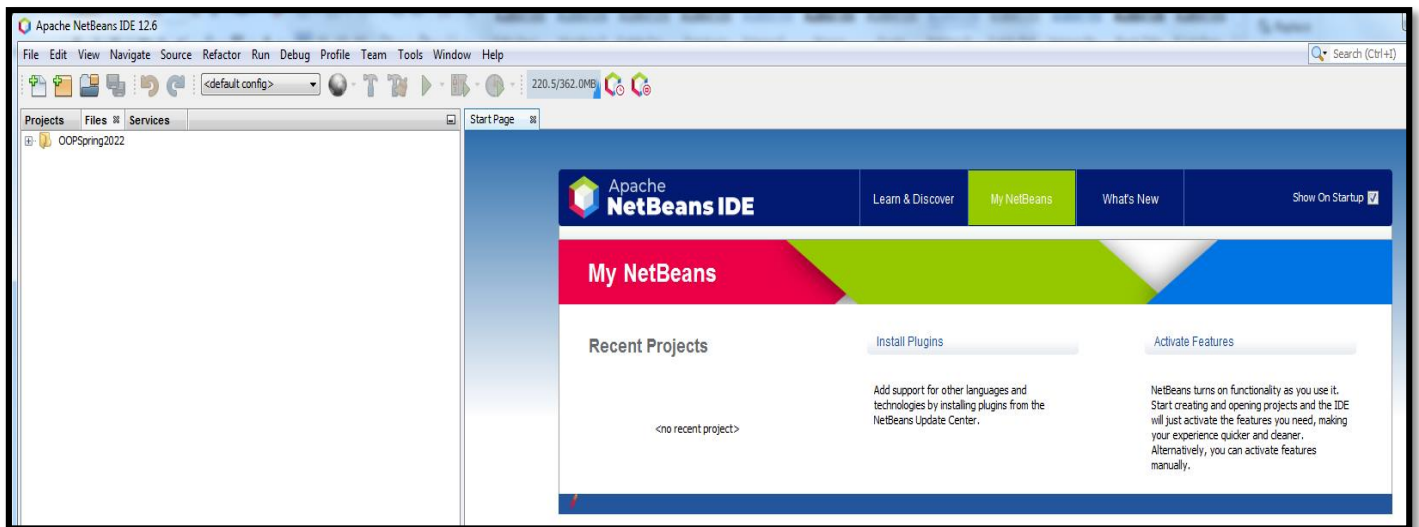
Java SE Development Kit 17.0.2 downloads		
Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications and components using the Java programming language.		
The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.		
Linux	macOS	Windows
Product/file description	File size	Download
x64 Compressed Archive	171.34 MB	<a href="https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip">https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip</a> (sha256 <a href="#">↗</a> )
x64 Installer	152.43 MB	<a href="https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe">https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe</a> (sha256 <a href="#">↗</a> )
x64 MSI Installer	151.32 MB	<a href="https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi">https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi</a> (sha256 <a href="#">↗</a> )

## USING NETBEANS:



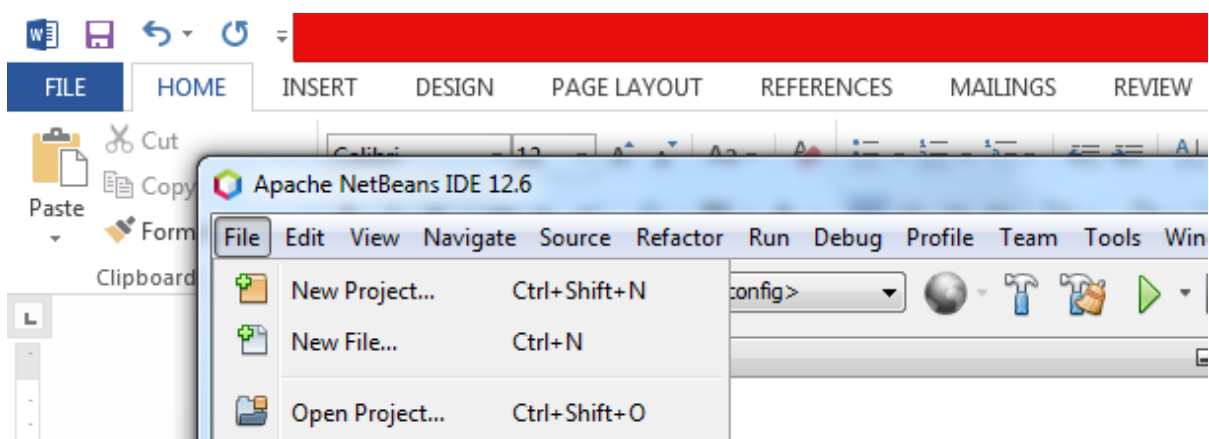
Use the shortcut on your Desktop to open NetBeans.

This is the first screen that appears when you open NetBeans.



## CREATING A PROJECT IN NETBEANS

Click on File – New Project



## Java Build Tools

### Maven

Apache Maven is a powerful software project management tool used in the Java development environment to manage and build projects as well as to maintain dependencies. Maven uses an XML (pom.xml) for project configuration.

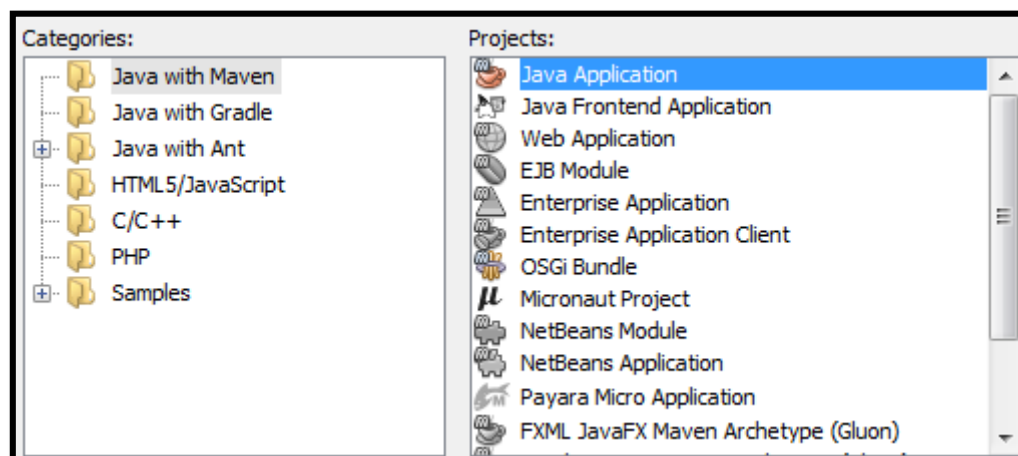
### Gradle

Gradle is a modern automation tool used in software development for project build automation. Gradle has its own domain-specific language (DSL) based on a Groovy (build.gradle) or Kotlin (build.gradle.kts) code.

### Ant

Apache Ant is the predecessor of Apache Maven. First released in 2000, Ant was developed as a replacement for a build tool Make, which was used widely in software development in the past. Using an XML file, Ant is used to automatize build tasks.

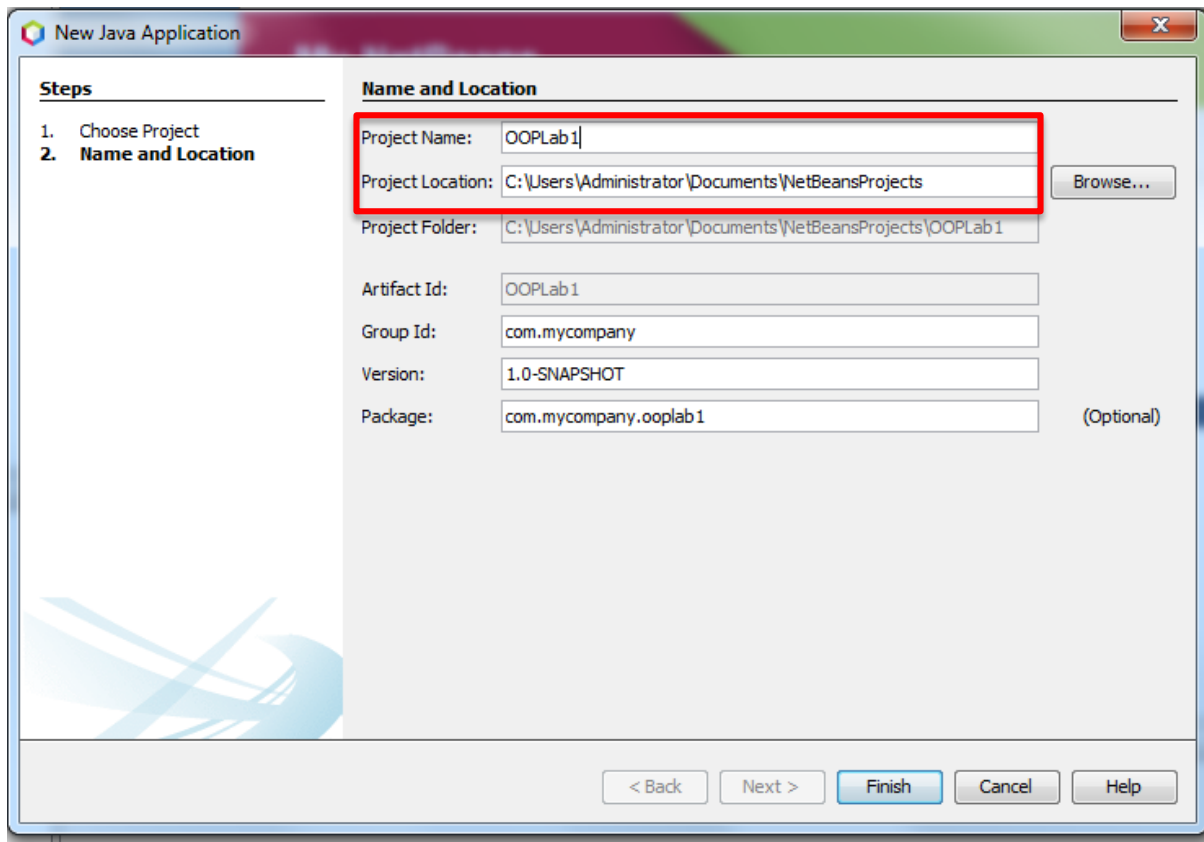
Click on Java Maven, then Java Application



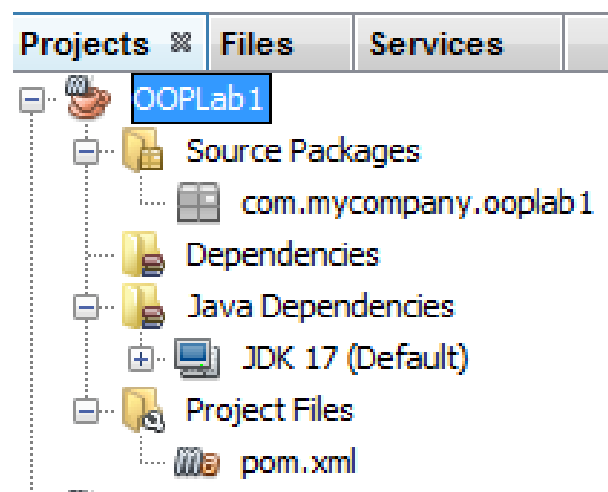


Write down your Project Name.

You can also choose a project location. Here we are using the default location.

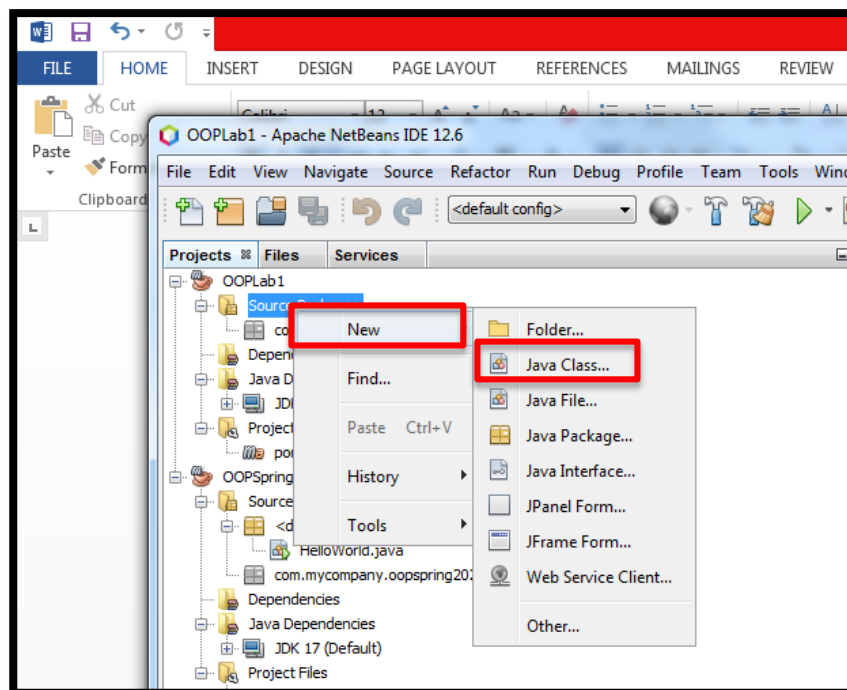


Once you click Finish, a project directory will be automatically created as seen in the Projects panel.

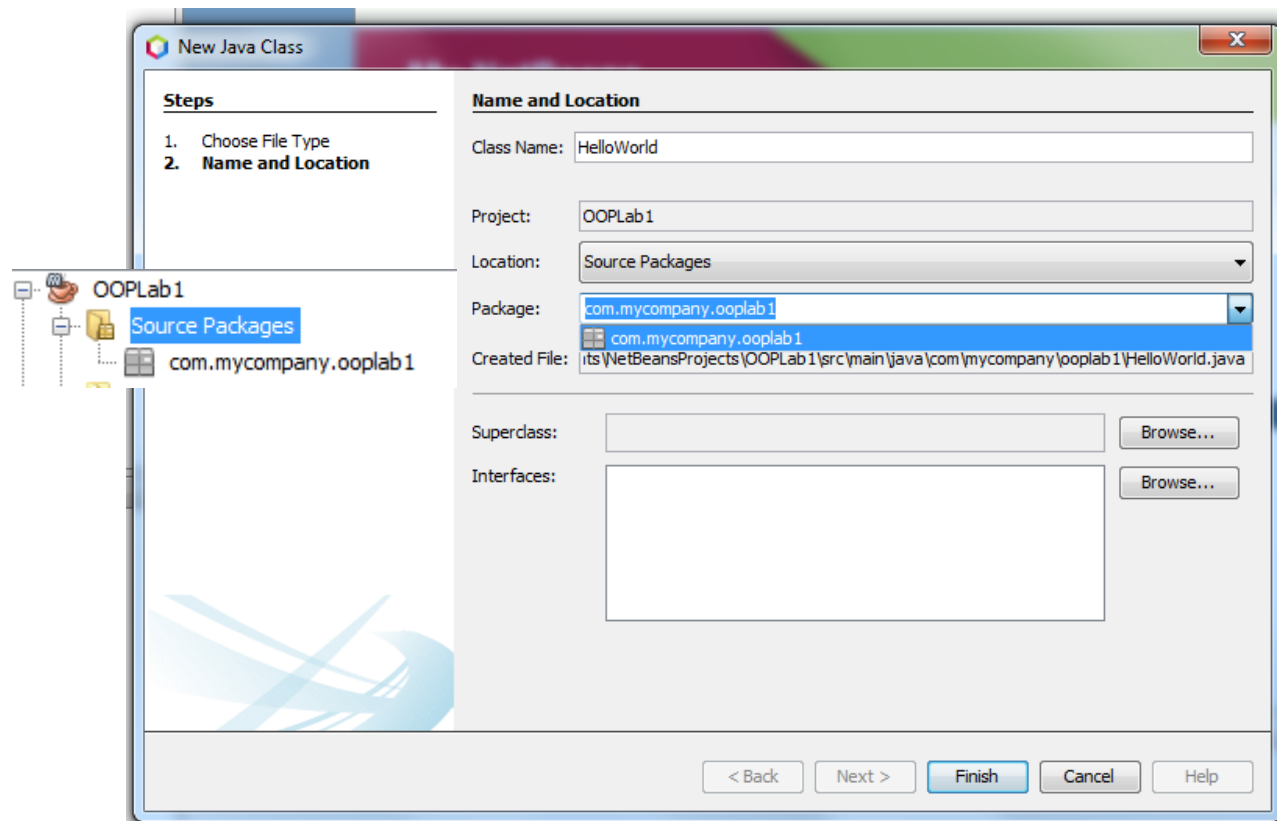


## CREATING A JAVA CLASS

Right click on source packages – New – Java Class



Once you click on Java Class, a window pops up. Write down a class name and choose the correct package that is present in source packages.



After Finish, a class is created.



Now let's create a simple program.

// MyFirstProgram **In Java, any line starting with // is a comment.**

public class HelloWorld { **In Java, every application begins with a class definition. In the program, HelloWorld is the name of the class. Every Java application has a class definition, and the name of the class should match the filename in Java.**

public static void main(String[] args) { **This is the main method.**

- **class keyword is used to declare a class in Java.**
- **public keyword is an access modifier that represents visibility.**
- **static is a keyword. The advantage of the static method is that there is no need to create an object to invoke the static method.**
- **The main() method is executed by the JVM, so it doesn't require creating an object to invoke the main() method. So, it saves memory.**
- **void is the return type of the method. It means it doesn't return any value.**
- **main represents the starting point of the program.**
- **String[] args means an array of sequence of characters (Strings) that are passed to the "main" function.**
- **When you execute a Java program via the command line: java MyProgram "This is just a test" Therefore, the array will store: ["This", "is", "just", "a", "test"]**

System.out.println("Hello, World!"); **System is a class, out is an object of the PrintStream class, println() is a method of the PrintStream class.**

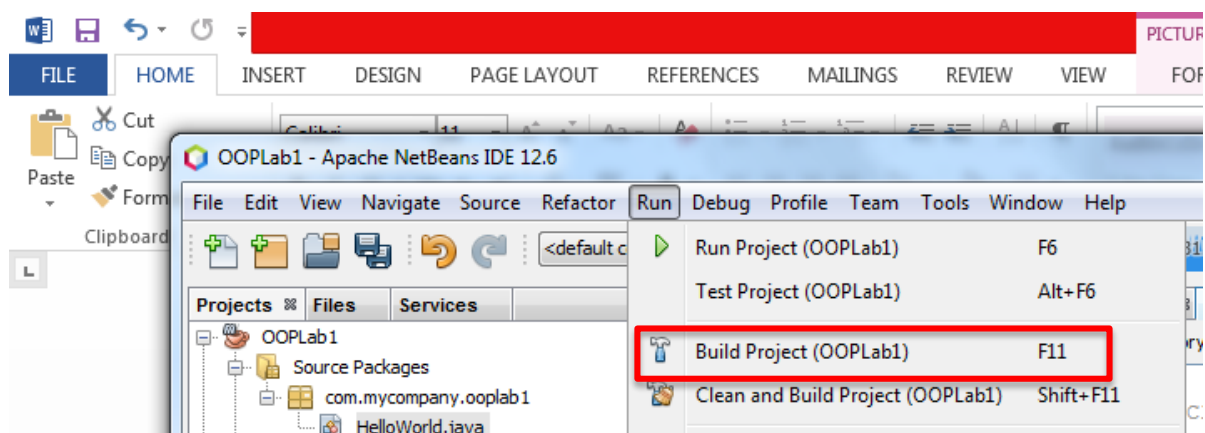
} }

## EXECUTING YOUR PROGRAM

```
// MyFirstProgram
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Build your project.

Click on Run – Build Project (OOP Lab1)



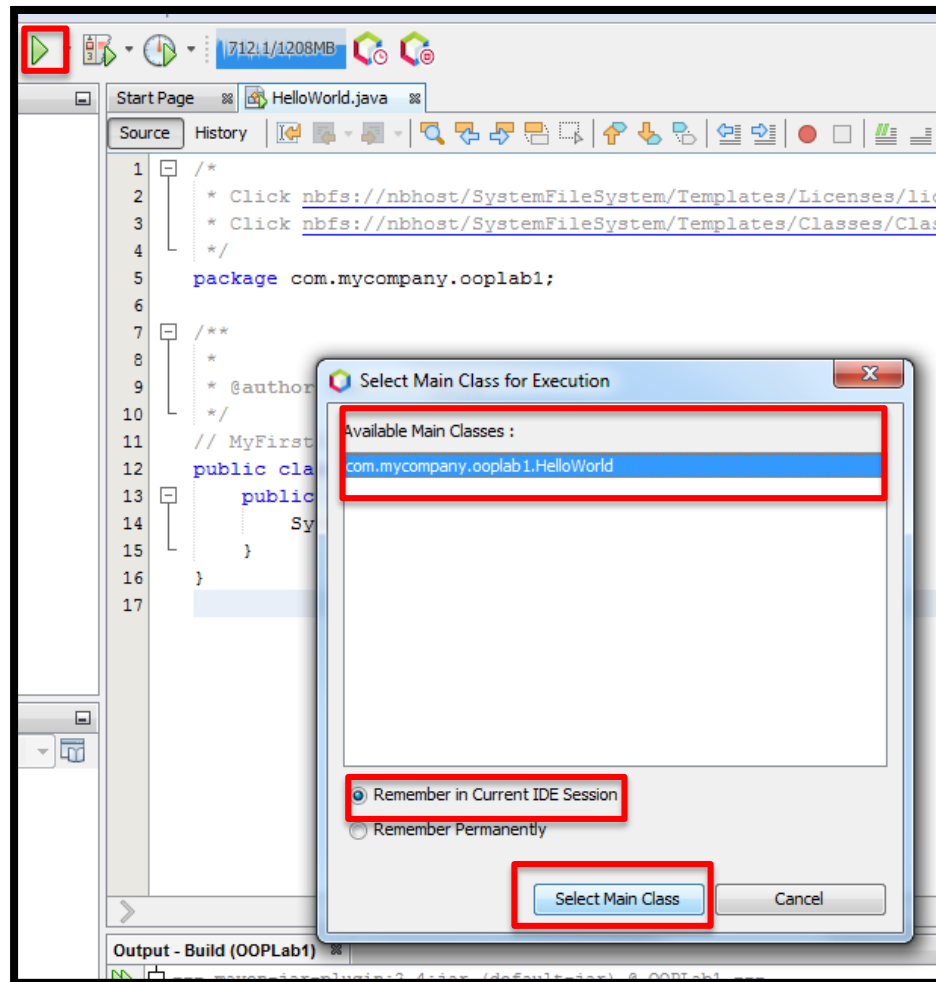
If Build is successful, the output screen displays the following message:

```
Output - Build (OOPLab1) %
--- maven-jar-plugin:2.4:jar (default-jar) @ OOPLab1 ---
Building jar: C:\Users\Administrator\Documents\NetBeansProjects\OOPLab1\tar
--- maven-install-plugin:2.4:install (default-install) @ OOPLab1 ---
Installing C:\Users\Administrator\Documents\NetBeansProjects\OOPLab1\targe
Installing C:\Users\Administrator\Documents\NetBeansProjects\OOPLab1\pom.x

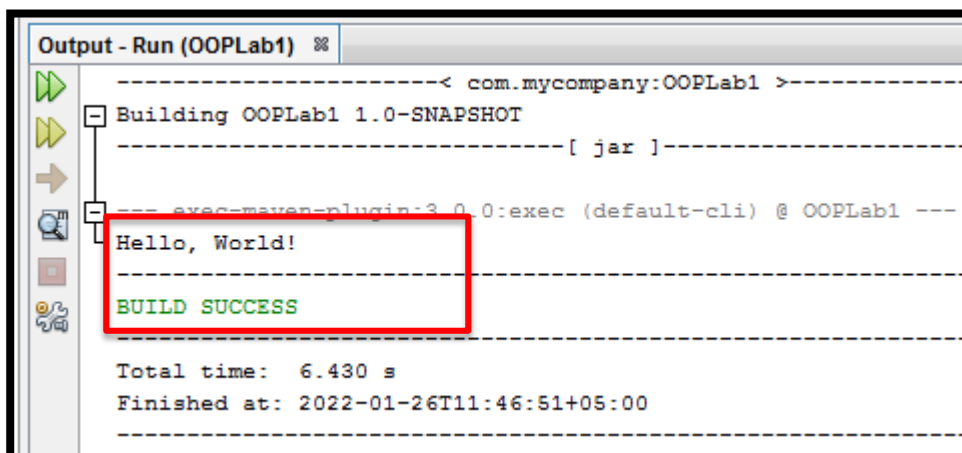
BUILD SUCCESS

Total time: 39.901 s
Finished at: 2022-01-26T11:27:55+05:00
```

To run your file click on the green arrow, and then select the main class and remember in Current IDE Session.



The output is displayed in the output window.



## DATA TYPES

Data types are divided into two groups:

Primitive data types - includes byte, short, int, long, float, double, boolean and char

Non-primitive data types - such as String, Arrays and Classes

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

```
public class Output {  
    public static void main(String[] args) {  
  
        int myNum = 5;           // Integer (whole number)  
        float myFloatNum = 5.99f; // Floating point number  
        char myLetter = 'D';     // Character  
        boolean myBool = true;   // Boolean  
        String myText = "Hello"; // String  
        byte bNum = 100;         // Byte  
        short sNum = 5000;       // Short  
        long lNum = 15000000000L; // Long  
  
        System.out.println("You entered integer" + " " + myNum);  
        System.out.println("You entered float" + " " + myFloatNum);  
        System.out.println("You entered character" + " " + myLetter);  
        System.out.println("You entered boolean" + " " + myBool);  
        System.out.println("You entered string" + " " + myText);  
        System.out.println("You entered Byte" + " " + bNum);  
        System.out.println("You entered short" + " " + sNum);  
        System.out.println("You entered long" + " " + lNum);  
  
    }  
}
```

Output - Run (OOPLab1) %

```
--- exec-maven-plugin:3.0.0:run  
You entered integer 5  
You entered float 5.99  
You entered character D  
You entered boolean true  
You entered string Hello  
You entered Byte 100  
You entered short 5000  
You entered long 15000000000  
-----  
BUILD SUCCESS
```

## INPUT AND OUTPUT IN JAVA

### Java output:

In Java, you can simply use

**System.out.println(); or**

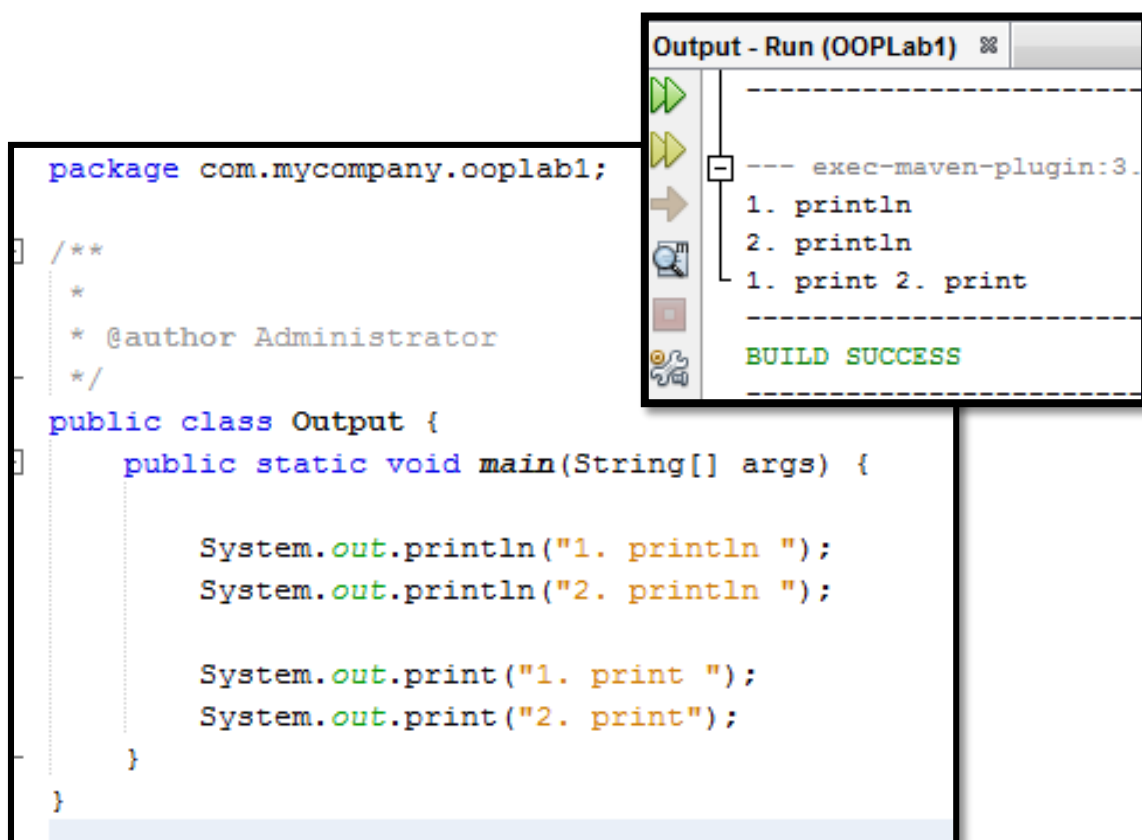
**System.out.print(); or**

**System.out.printf();**

to send output to standard output (screen). Here, System is a class out is a public static field: it accepts output data.

Difference between println(), print() and printf()

- print() - It prints string inside the quotes.
- println() - It prints string inside the quotes similar like print() method. Then the cursor moves to the beginning of the next line.
- printf() - It provides string formatting (similar to printf in C/C++ programming).



The screenshot displays an IDE with a Java source file on the left and an output window on the right. The source file, located at `package com.mycompany.ooplal1;`, contains a `public class Output` with a `main` method. The `main` method uses `System.out.println` and `System.out.print` to output a sequence of strings. The output window, titled "Output - Run (OOPLab1)", shows the execution results, including a Maven build success message and the printed output: "1. println", "2. println", "1. print 2. print".

```
package com.mycompany.ooplal1;

/**
 *
 * @author Administrator
 */
public class Output {
    public static void main(String[] args) {

        System.out.println("1. println ");
        System.out.println("2. println ");

        System.out.print("1. print ");
        System.out.print("2. print");

    }
}
```

Output - Run (OOPLab1) ✖

```
--- exec-maven-plugin:3.
1. println
2. println
1. print 2. print

BUILD SUCCESS
```

## FORMATTING OUTPUT WITH PRINTF() IN JAVA

Conversion characters are only valid for certain data types. Here are some common ones:

Specifier	Explanation
%c	Format characters
%d	Format decimal (integer) numbers (base 10)
%e	Format exponential floating-point numbers
%f	Format floating-point numbers
%i	Format integers (base 10)
%o	Format octal numbers (base 8)
%s	Format string of characters
%u	Format unsigned decimal (integer) numbers
%x	Format numbers in hexadecimal (base 16)
%n	add a new line character

### Float and Double Formatting

To format a float number, we'll need the f format:

**System.out.printf("%f%n", 5.1473);** which will output: **5.147300**

To control the precision: **System.out.printf("%5.2f%n", 5.1473);** Here we define the width of our number as 5, and the length of the decimal part is 2: **'5.15'**.

### Integer Formatting

The printf() method accepts all the integers available in the language — byte, short, int, long, and BigInteger if we use %d:

**System.out.printf("simple integer: %d%n", 10000L);**

With the help of the d character, we'll have this result: **simple integer: 10000**

### String Formatting

To format a simple string, we'll use the %s combination. Additionally, we can make the string uppercase:

```
printf("%s' %n", "Java");
```

```
printf("%S' %n", "JAVA");
```

And this is the output:

'Java'

'JAVA'



### Char Formatting

The result of %c is a Unicode character:

```
System.out.printf("%c%n", 's');
```

```
System.out.printf("%C%n", 's');
```

The capital letter C will uppercase the result:

s

S

### Input from user in Java

Java Scanner Class

Java Scanner class allows the user to take input from the console. It belongs to java.util package.

Syntax

***Scanner sc = new Scanner(System.in);***

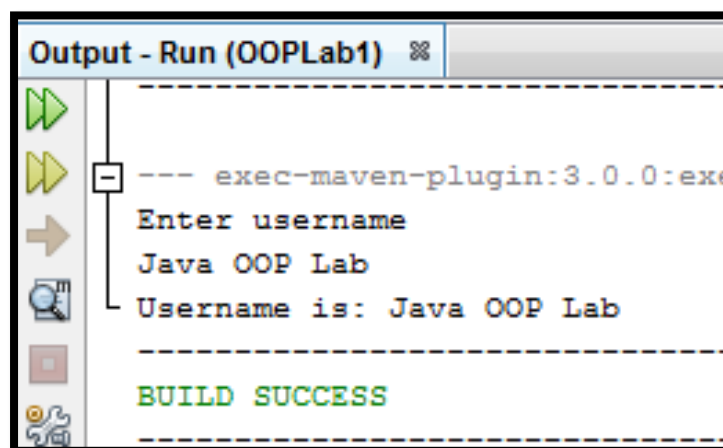
The above statement creates a constructor of the Scanner class having System.in as an argument. It means it is going to read from the standard input stream of the program. The java.util package should be imported while using Scanner class.

Methods of Java Scanner Class

Method	Description
<b>int nextInt()</b>	It is used to scan the next token of the input as an integer.
<b>float nextFloat()</b>	It is used to scan the next token of the input as a float.
<b>double nextDouble()</b>	It is used to scan the next token of the input as a double.
<b>byte nextByte()</b>	It is used to scan the next token of the input as a byte.
<b>String nextLine()</b>	Advances this scanner past the current line.
<b>boolean nextBoolean()</b>	It is used to scan the next token of the input into a boolean value.
<b>long nextLong()</b>	It is used to scan the next token of the input as a long.
<b>short nextShort()</b>	It is used to scan the next token of the input as a Short.
<b>BigInteger nextBigInteger()</b>	It is used to scan the next token of the input as a BigInteger.
<b>BigDecimal nextBigDecimal()</b>	It is used to scan the next token of the input as a BigDecimal.

### Example: Input your name

```
package com.mycompany.ooplal1;  
import java.util.Scanner; // import the Scanner class  
  
class Input1 {  
    public static void main(String[] args) {  
        Scanner myObj = new Scanner(System.in);  
        String userName;  
  
        // Enter username and press Enter  
        System.out.println("Enter username");  
        userName = myObj.nextLine();  
  
        System.out.println("Username is: " + userName);  
    }  
}
```



```
Output - Run (OOPLab1) %  
--- exec-maven-plugin:3.0.0:exe  
Enter username  
Java OOP Lab  
Username is: Java OOP Lab  
-----  
BUILD SUCCESS  
-----
```

### Example: Input integers

```
package com.mycompany.ooplal1;
import java.util.Scanner; // import the Scanner class

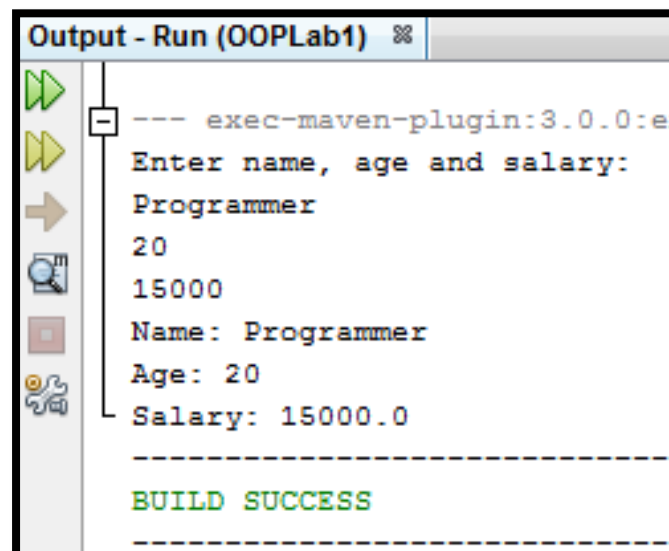
class Input1 {
    public static void main(String[] args) {
        Scanner myObj = new Scanner(System.in);

        System.out.println("Enter name, age and salary:");

        // String input
        String name = myObj.nextLine();

        // Numerical input
        int age = myObj.nextInt();
        double salary = myObj.nextDouble();

        // Output input by user
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Salary: " + salary);
    }
}
```



```
Output - Run (OOPLab1)
--- exec-maven-plugin:3.0.0:ex
Enter name, age and salary:
Programmer
20
15000
Name: Programmer
Age: 20
Salary: 15000.0
-----
BUILD SUCCESS
-----
```

### Example: Adding two numbers

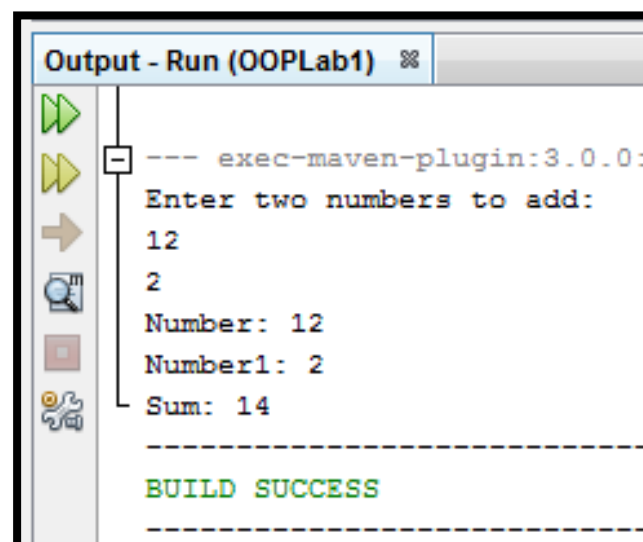
```
package com.mycompany.ooplal1;
import java.util.Scanner; // import the Scanner class

class Input1 {
    public static void main(String[] args) {
        Scanner myObj = new Scanner(System.in);

        System.out.println("Enter two numbers to add:");

        // Numerical input
        int number = myObj.nextInt();
        int number1 = myObj.nextInt();
        int sum = number + number1;

        // Output input by user
        System.out.println("Number: " + number);
        System.out.println("Number1: " + number1);
        System.out.println("Sum: " + sum);
    }
}
```



## **LAB#01 EXERCISES**

### **QUESTION#1**

Write a program that calculates how long it takes to drive from Karachi to Lahore at 75 mile per hour (Use 3000 miles as the approximate distance between two cities).

### **QUESTION#2**

- a) Write an application that accepts two doubles, multiple these together and display the product.
- b) Write a Program to print the area of a triangle.

### **QUESTION#3**

Write a Java program that works as a simple calculator for the +, -, \*, / operations. Take two integer numbers from the user and perform all the four operations.

### **QUESTION#4**

Write a Java program that prints all real solutions to the quadratic equation  $ax^2 + bx + c = 0$ . Read in a, b, c and use the quadratic formula.

### **QUESTION#5**

Write a Java program that inputs three integers from the user and displays the sum, average, and product of these numbers.

### **QUESTION#6**

Write a Java program that inputs from the user the radius of a circle as an integer and prints the circle's diameter, circumference and area using the floating-point value 3.14159 for  $\pi$ .

### **QUESTION#7**

Write a Java program that takes as input your name, student ID, current courses registered for and displays all the information.