

1.Installing java

Currently I am using ubuntu(Linux).Now I will describe how to install java on my pc.

1. For downloading Java Open web browser Type URL: <https://www.oracle.com/java/technolog...> to go to Oracle download page. This will lead JAVA JDK download page Click on button “ JDK download ” for Java SE update 4. Accept oracle license agreement Find and click on the correct jdk download link right for your operating system to download Save the file to disk.

2. Install Java

I will have to perform this for installing java.

```
$ sudo dpkg -i jdk-14_linux-x64_bin.deb
```

```
$ sudo update-alternatives --install /usr/bin/java java /usr/lib/jvm/jdk-14/bin/java 1
```

```
$ sudo update-alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk-14/bin/javac 1
```

```
$ java --version
java 14 2020-03-17 Java(TM) SE Runtime Environment (build 14+36-1461)
Java HotSpot(TM) 64-Bit Server VM (build 14+36-1461, mixed mode, sharing)
```

```
$ javac --version
javac 14
```

3.After installing java on my system I prefer to use a IDE.for that I install intelliJ idea.For install intelliJ

I will simply perform this command:

```
$ sudo snap install intellij-idea-community --classic
```

After that my system is ready with java environment!!

2.JRE and JDK

JRE:The *Java Runtime Environment* is a software layer that runs on top of a computer's operating system software and provides the *class libraries* and other resources that a specific java program needs to run.

JDK:The *Java Development Kit* is a set of tools for developing Java applications. Developers choose JDKs by Java version and by package or edition—Java Enterprise Edition (Java EE), Java Special Edition (Java SE), or Java Mobile Edition (Java ME). Every JDK always includes a compatible JRE, because running a Java program is part of the process of developing a Java program.

**Difference between JDK and JRE

JDK is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed. The difference between JDK and JRE is that JDK is the software development kit for java while JRE is the place where you run your programs.

3.Connect JDK with the environment:

After installing The JDK software on computer in the default location; for example, at /usr/jdk/jdk1.6.0_02. You can change this location.

- To set JAVA_HOME, do one of the following:
- For Korn and bash shells, run the following commands:

```
export JAVA_HOME=jdk-install-dir
```

```
export PATH=$JAVA_HOME/bin:$PATH
```

- For the bourne shell, run the following commands:

```
JAVA_HOME=jdk-install-dir
```

```
export JAVA_HOME
```

```
PATH=$JAVA_HOME/bin:$PATH
```

```
export PATH
```

- For the C shell, run the following commands:

```
setenv JAVA_HOME jdk-install-dir
```

```
setenv PATH $JAVA_HOME/bin:$PATH
```

```
export PATH=$JAVA_HOME/bin:$PATH
```

- Change the permissions to enable you to run the GlassFish ESB Installer by running the following command:

```
chmod 755 JavaCAPS.bin
```

4.

A package in Java is used to group related classes. Think of it as a folder in a file directory. We use packages to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories:

- Built-in Packages (packages from the Java API)
- User-defined Packages (create your own packages)

5.

Comments in Java are the statements that are not executed by the compiler and interpreter. It can be used to provide information or explanation about the variable, method, class or any statement. It can also be used to hide program code for a specific time.

Use `// text` when you want to comment a single line of code. Use `/* text */` when you want to comment multiple lines of code. Use `/** documentation */` when you would want to add some info about the program that can be used for automatic generation of program documentation.

6.

Public: It is an *Access modifier*, which specifies from where and who can access the method. Making the *main()* method public makes it globally available. It is made public so that JVM can invoke it from outside the class as it is not present in the current class.

Void: It is a keyword and used to specify that a method doesn't return anything. As *main()* method doesn't return anything, its return type is void. As soon as the *main()* method terminates, the java

program terminates too. Hence, it doesn't make any sense to return from main() method as JVM can't do anything with the return value of it.

main: It is the name of Java main method. It is the identifier that the JVM looks for as the starting point of the java program. It's not a keyword.

String[] args: It stores Java command line arguments and is an array of type java.lang.String class. Here, the name of the String array is args but it is not fixed and user can use any name in place of it.

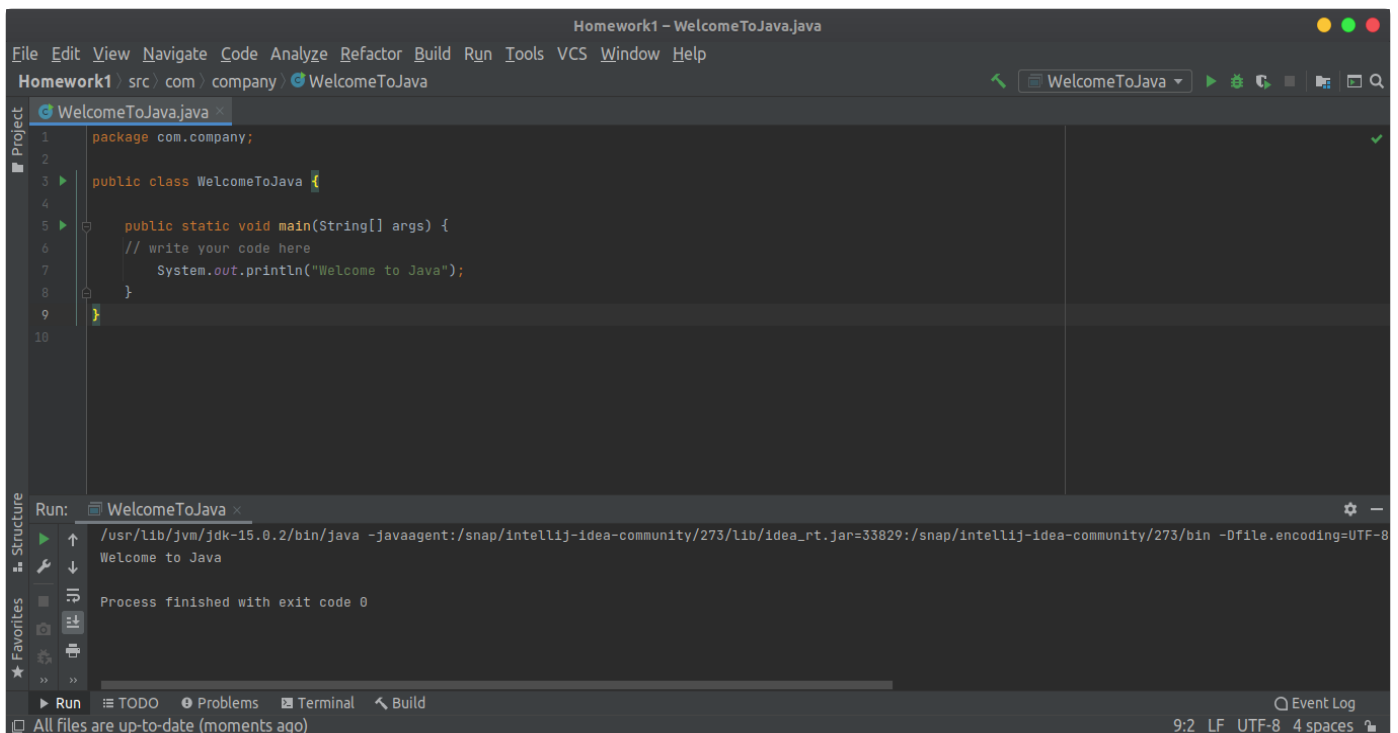
Apart from the above mentioned signature of main, you could use public static void main(String args[]) or public static void main(String... args) to call the main function in java. The main method is called if it's formal parameter matches that of an array of Strings.

7.The input of a Java compiler is a Java source code file and the output is a Java class file.

8.

JVM:Java Virtual Machine is a engine that provides runtime environment to drive the Java Code or applications. It converts Java bytecode into machines language. JVM is a part of Java Run Environment (JRE). In other programming languages, the compiler produces machine code for a particular system

9.Screenshot for java program



The screenshot displays an IDE window titled "Homework1 - WelcomeToJava.java". The main editor shows the following Java code:

```
1 package com.company;
2
3 public class WelcomeToJava {
4
5     public static void main(String[] args) {
6         // write your code here
7         System.out.println("Welcome to Java");
8     }
9 }
10
```

Below the code editor, the "Run" tab is active, showing the command used to execute the program:

```
/usr/lib/jvm/jdk-15.0.2/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=33829:/snap/intellij-idea-community/273/bin -Dfile.encoding=UTF-8
```

The output of the program is "Welcome to Java". Below the output, it states "Process finished with exit code 0".

The bottom status bar indicates "All files are up-to-date (moments ago)" and "9:2 LF UTF-8 4 spaces".

Homework1 - WelToCS.java

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Homework1 > src > com > company > WelToCS

WelToCS (1)

```
1 public class WelToCS {
2
3     public static void main(String[] args) {
4         // write your code here
5         System.out.println("Welcome to Computer Science");
6     }
7 }
8
9
```

Run: WelToCS (1)

```
/usr/lib/jvm/jdk-15.0.2/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=41703:/snap/intellij-idea-community/273/bin -Dfile.encoding=UTF-8
Welcome to Computer Science

Process finished with exit code 0
```

Run | TODO | Problems | Terminal | Build | Event Log

Build completed successfully in 1 sec, 601 ms (moments ago) 9:2

Homework1 - CrazyForCoding.java

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Homework1 > src > com > company > CrazyForCoding

CrazyForCoding

```
1 public class CrazyForCoding {
2
3     public static void main(String[] args) {
4         // write your code here
5         System.out.println("Crazy for Coding");
6     }
7 }
8
9
```

Run: CrazyForCoding

```
/usr/lib/jvm/jdk-15.0.2/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=41335:/snap/intellij-idea-community/273/bin -Dfile.encoding=UTF-8
Crazy for Coding

Process finished with exit code 0
```

Run | TODO | Problems | Terminal | Build | Event Log

Build completed successfully in 1 sec, 818 ms (moments ago) 9:2 LF UTF-8 4 spaces