Java

Module 1

Introduction to Java

Author
Srinivas Dande

1.1 Introduction

Java is Simple, High Level, Platform independent, Architecture-Neutral , Secure, Robust, Multithreaded, Distributed, and Object Oriented programming language.

1.2 Java Characteristics / Features / Buzzwords

- **Simple**
  - Java is easy to learn because most of the complex or confusing features of C and C++ like pointers, operator overloading etc are not provided in Java.
- **Secure**
  - Java programs run within the JVM which protects from unauthorized or illegal access to system resources.
- **Platform Independent / Portable**
  - Java program can be executed on any kind of machine containing any CPU or any operating system.
- **Architecture-neutral**
  - Since Java applications can run on any kind of CPU so it is architecture – neutral.
- **Robust**
  - Java is robust because of following:
    - Strong memory management
    - No Pointers
    - Exception handling and Type checking
    - Platform Independent
- **Multithreaded**
  - Java supports multithreaded programming, which allows us to write programs that do many things simultaneously.
- **Distributed**
  - Using RMI and EJB we can create distributed applications in java.
- **Object-oriented**
  - Java follows object oriented programming model that helps to break the complex code into easy to understand objects.

1.3 Programming Languages

- **Language**
  - It is the medium of communication.

- **Program**
  - It is a set of instructions for a device to perform some specific task.

- **Programming Language**
  - It contains set of grammatical rules or syntax to write a program for instructing machine to perform specific tasks.

1.3.1 Types of Programming Languages

1. Low Level Programming Language
2. Middle Level Programming Language
3. High Level Programming Language

1.3.1.1 Low Level Programming Languages

- Binary digits(0,1) will be used to write the instructions in Low Level Language.
- Instructions written in binary language or native language will be understood by the machine directly.
- No translator program is required.

Problem:

- It is very complex to write, modify and debug the instructions written in Binary Language.
- Binary code or Native code is platform dependent.

1.3.1.2 Middle Level Programming Languages

- Assembly codes like ADD, SUB, MUL, DIV, MOV etc. will be used to write the instructions in Middle Level Language.
- Instructions written in assembly language will not be understood by the machine directly.
- Assembler is a program that should be used to convert Assembly language to Machine language.

Problem:

- The code written in Assembly language will be processor dependent.

1.3.1.3 High Level Programming Languages

- Human understandable codes like English words will be used to write the instructions in High Level Language.
- Instructions written in High Level Language will not be understood by the machine directly.
- Compiler or Interpreter is a program that should be used to convert High Level Language to Machine language.

Advantages:

- It is easy to write, modify and debug the instructions written using High Level Language.

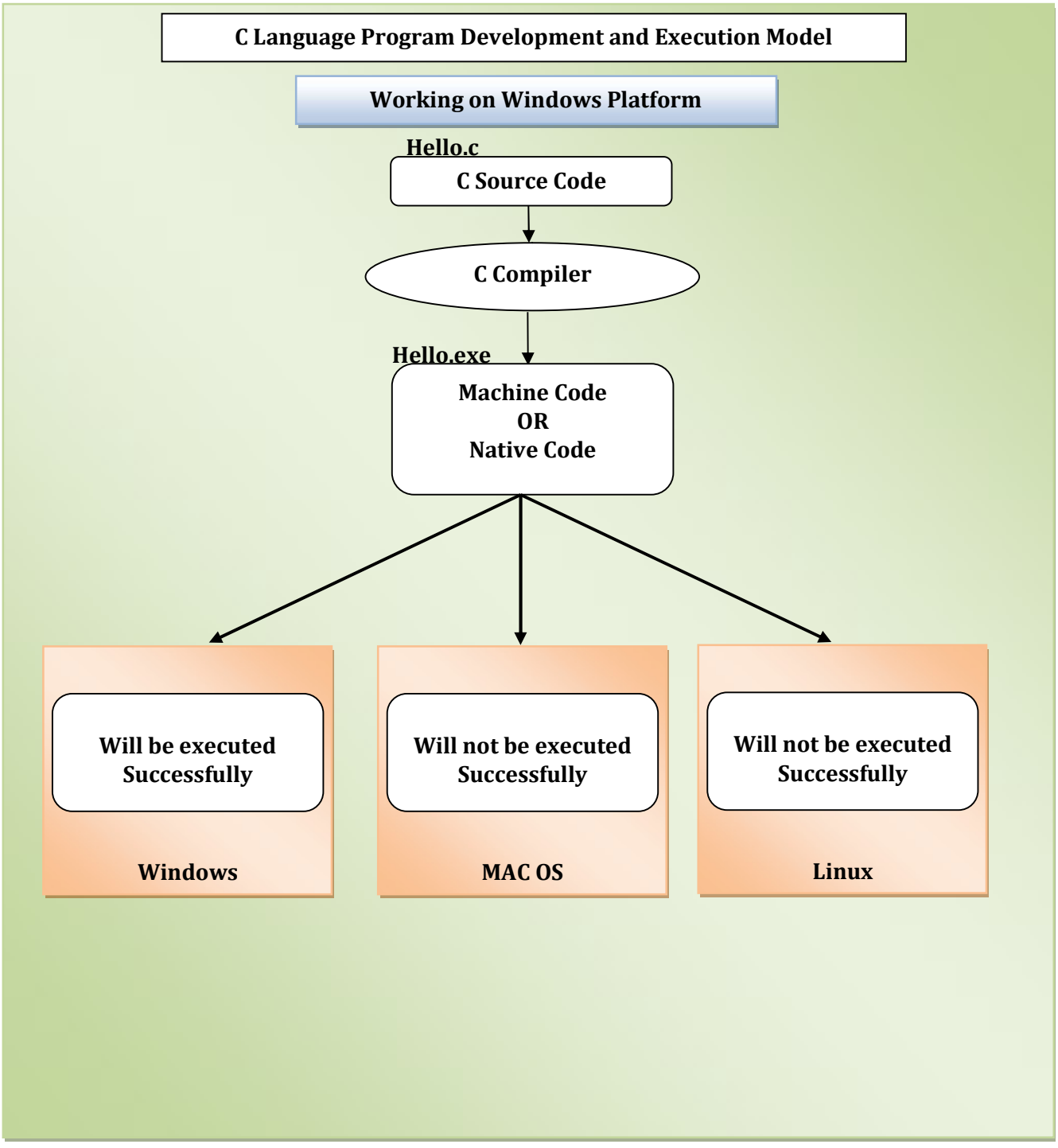

C Language Program Development and Execution Model
Working on Windows Platform
Hello.c
C Source Code
C Compiler
Hello.exe
Machine Code
OR
Native Code
Will be executed Successfully
Windows
Will not be executed Successfully
MAC OS
Will not be executed Successfully
Linux

Diagram - 1

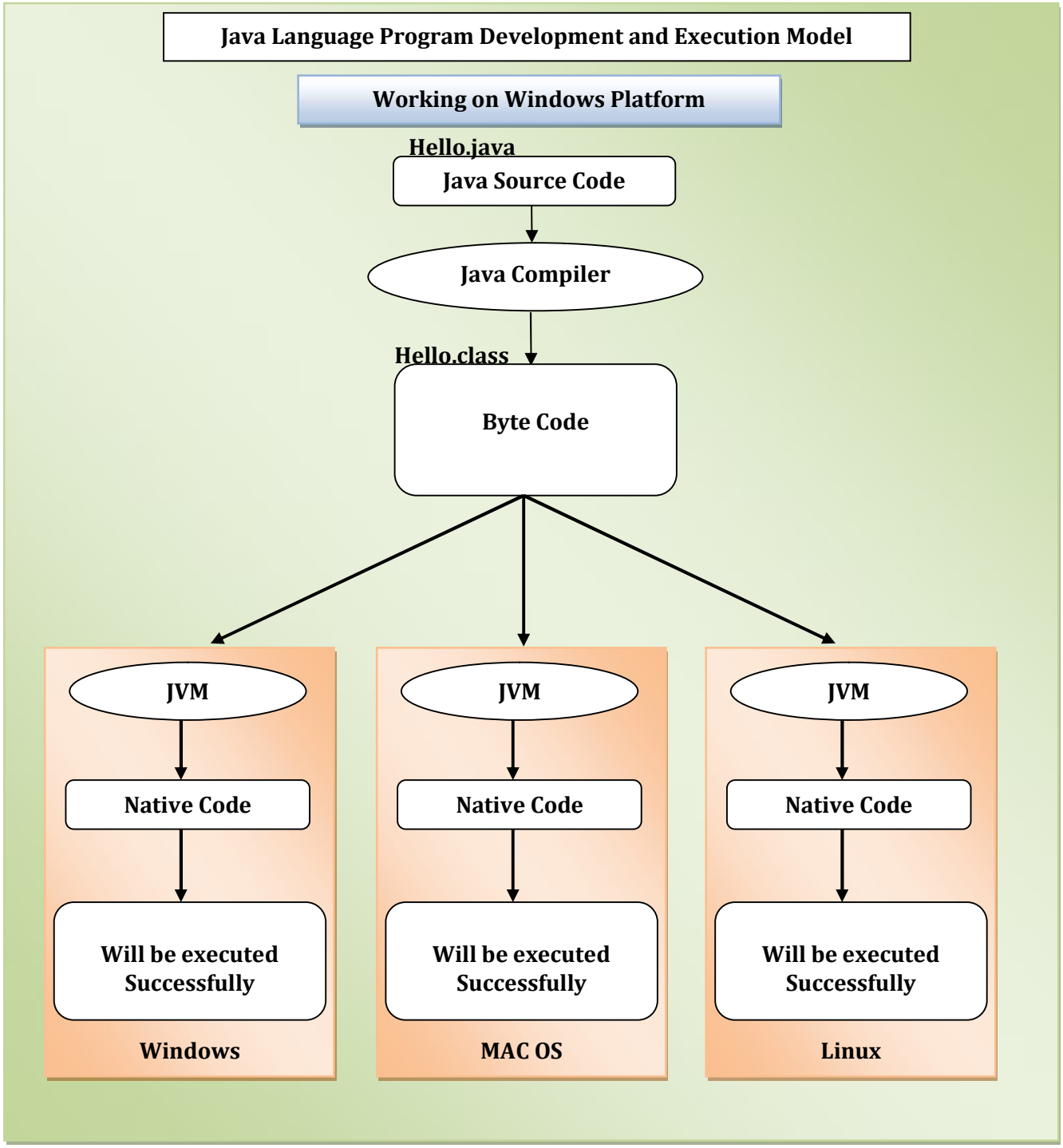

Java Language Program Development and Execution Model
Working on Windows Platform
Hello.java
Java Source Code
Java Compiler
Hello.class
Byte Code
JVM
Native Code
Will be executed Successfully
Windows
JVM
Native Code
Will be executed Successfully
MAC OS
JVM
Native Code
Will be executed Successfully
Linux

Diagram - 2

Q1) What is Source Code?

Ans:

Human understandable code written using High Level Programming language is called as Source Code.

Q2) What is Native Code or Binary Code or Machine Code?

Ans:

Machine understandable code written using Low Level Programming language is called as Native Code.

Q3) What is Byte Code?

Ans:

JVM understandable code generated by Java Compiler is called as Byte Code. Byte code is also called as Magic Value.

Q4) Why C and C++ are Platform Dependent?

Ans:

When you compile C or C++ program on one Operating System then compiler generates that Operating System understandable native code.This native code generated on one OS will not run on other OS directly.

Q5) Why Java is Platform Independent?

Ans:

When you compile Java program on one Operating System then Java compiler generates Byte Code. You can run that Byte code on any OS which has the JVM.
JVM is understanding the Byte code and generating native code for the corresponding OS.
Because of Byte code and JVM, Java is Platform Independent.
You can use - Write Once Run Anywhere (WORA).

Q6) I have written one program in C or C++ and another program in Java. Which program will be executed fastly and why?

Ans:

C or C++ program execution is faster than Java program.
Because compiled code of C or C++ program contains binary code and that binary code will be executed directly whereas compiled code of Java program contains byte code and that byte code will be converted to native code first and then native code will be executed.

Q7) What is Java Compiler?

Ans:

Java Compiler is a program developed in C or C++ programming language with the name "javac".

It is responsible for the following tasks:

- It will check syntactical or grammatical errors of the programs.
- It converts source code to byte code.

Q8) What is Java Interpreter?

Ans:

Java Interpreter is a program developed in C or C++ programming language with the name "java".

It is responsible for the following tasks:

- It will convert byte code to native code line by line.
- It will execute that native code.

Note: Because of converting and executing line by line, Java program execution was slower in initial versions of java.

Q9) What is JIT Compiler?

Ans:

JIT (Just-In-Time) compiler is a component of the Java Runtime Environment.
JIT Compiler compiles or translates or converts the neccessary part of the bytecode into machine code instead of converting line by line. Because of this, performance of Java program has improved.

Q10) What is JRE?

Ans:

JRE stands for Java Runtime Environment. It is an implementation of JVM. It contains class libraries, Interpreter, JIT Compiler etc. Only JRE is enough to run the Java program.

Q11) What is JVM?

Ans:

JVM stands for Java Virtual Machine. It is a specification provided by SUN Microsystem whose implementation provides an environment to run our Java applications. JVM becomes an instance of JRE at run time.
Sun's implementations of the Java Virtual Machine (JVM) is itself called as JRE.
Sun's JRE is availabe as a part of JDK and also as a separate application.
Many vendors has implemented JVM. Some of them are SUN JRE, IBM JRE, Oracle JRE etc.

Q12) What is JDK (Java Development Kit) / SDK (Software Development Kit)?

Ans:

It is a set of various utility programs which are required for developing and executing the java programs.
It is Platform dependent. Various JDKs are provided for various Operating Systems.

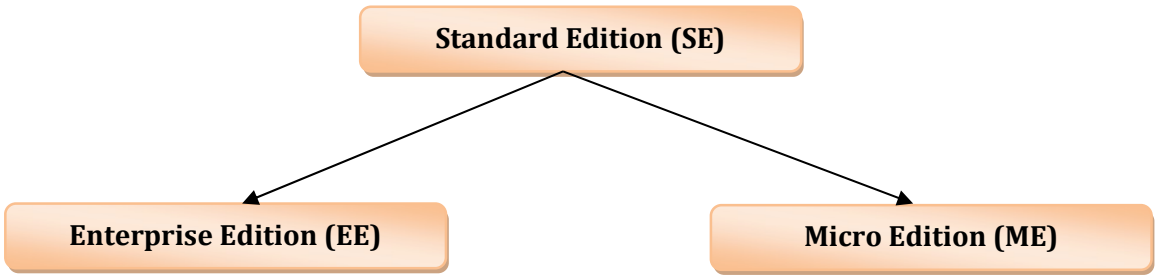
Following are various utility programs provided under JDK:

1) Java Development Tools
    i. javac
    ii. java
    iii. javap
    iv. jar
    etc
2) Source Files
3) JRE
    etc

1.4 History

- Java was designed by Sun Microsystems in the early 1990s to solve the problem of connecting many household machines together.
- Initial version of Java is called as OAK.
- Later Java became popular for developing internet applications.
- Java is developed by James Gosling and team.
- James Gosling is considered as FATHER OF JAVA.
- It was publicly released on 1995.
- Now Sun Microsystems is the part of Oracle Corp.

1.5 Editions



- **Java Standard Edition (JSE):**
  - Used to develop standalone applications using applet and swing.
- **Java Enterprise Edition (JEE):**
  - Used to develop enterprise applications using Servlets, JSP, JDBC etc.
- **Java Micro Edition (JME):**
  - Used to develop applications for micro devices like Mobiles,Setup Box etc.

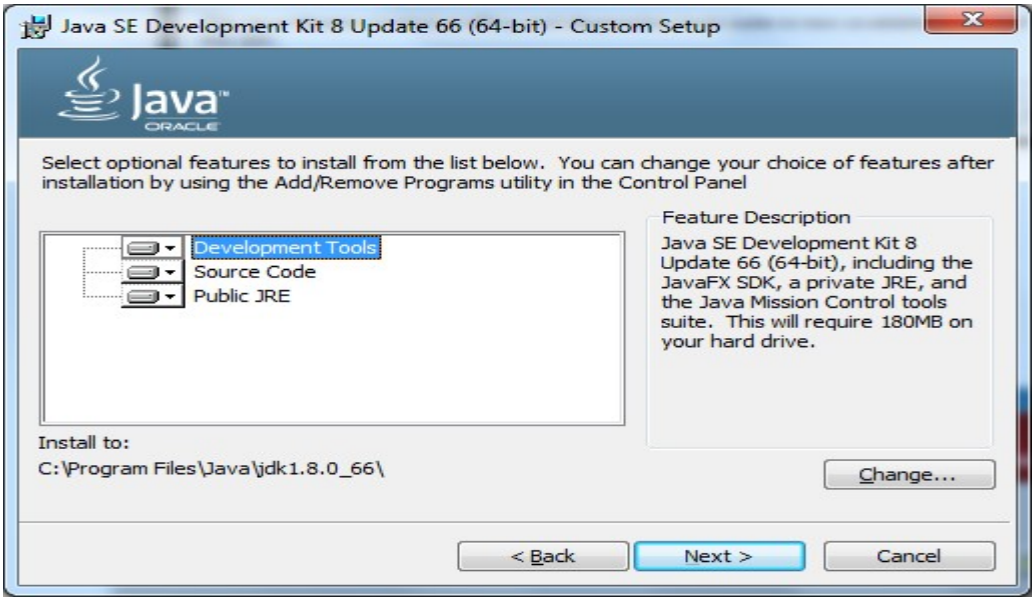
1.6 Java Versions

Java Version	Class Format Version	Code Name	Released Year
JDK Alpha and Beta			1995
JDK 1.0		Oak	1996
JDK 1.1			1997
J2SE 1.2		Playground	1998
J2SE 1.3		Kestrel	2000
J2SE 1.4	48	Merlin	2002
J2SE 5	49	Tiger	2004
Java SE 6	50	Mustang	December 2006
Java SE 7	51	Dolphin	July 2011
Java SE 8 (LTS)	**52**		**March 2014**
Java SE 9	53		September 2017
Java SE 10	54		March 2018
Java SE 11(LTS)	**55**		**September 2018**
Java SE 12	56		March 2019
Java SE 13	57		September 2019
Java SE 14	58		March 2020
Java SE 15	59		September 2020
Java SE 16	60		March 2021
Java SE 17(LTS)	**61**		**September 2021**
Java SE 18	62		March 2022
Java SE 19	63		September 2022
Java SE 20	64		March 2023
Java SE 21(LTS)	**65**		**September 2023**
Java SE 22	66		March 2024

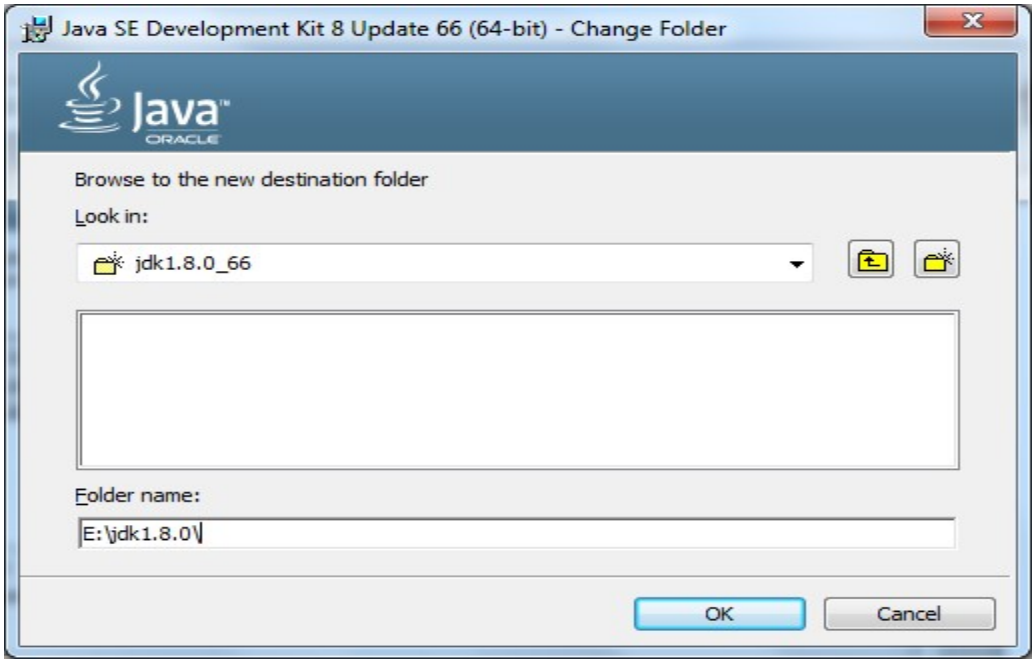
1.7 Installation Steps

- Run the installer file jdk-8u45-windows-x64.exe or jdk-8u66-windows-i586.exe from Student DVD/JDK 8
- Following screen will appear

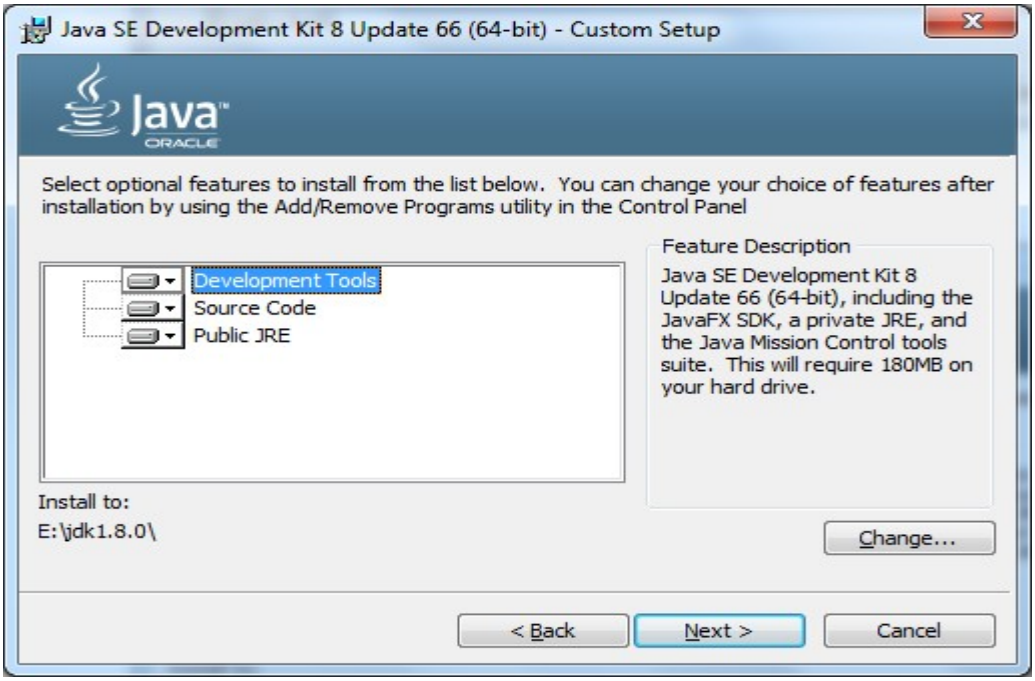
- Click on NEXT button.
- Following screen will appear



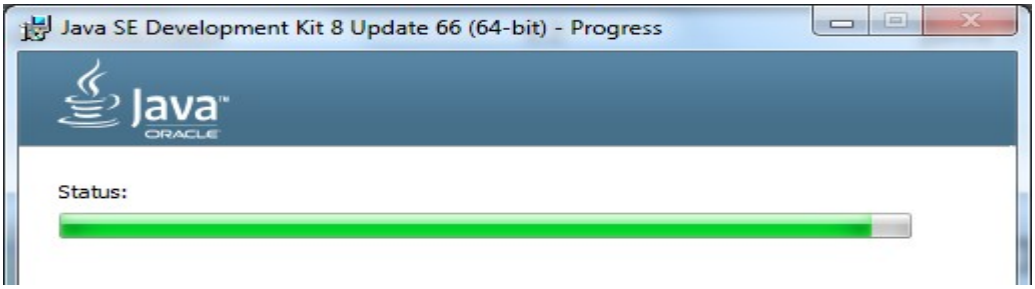
- Click on CHANGE button.
- Change the destination location to E:\jdk1.8.0\
- Following screen will appear



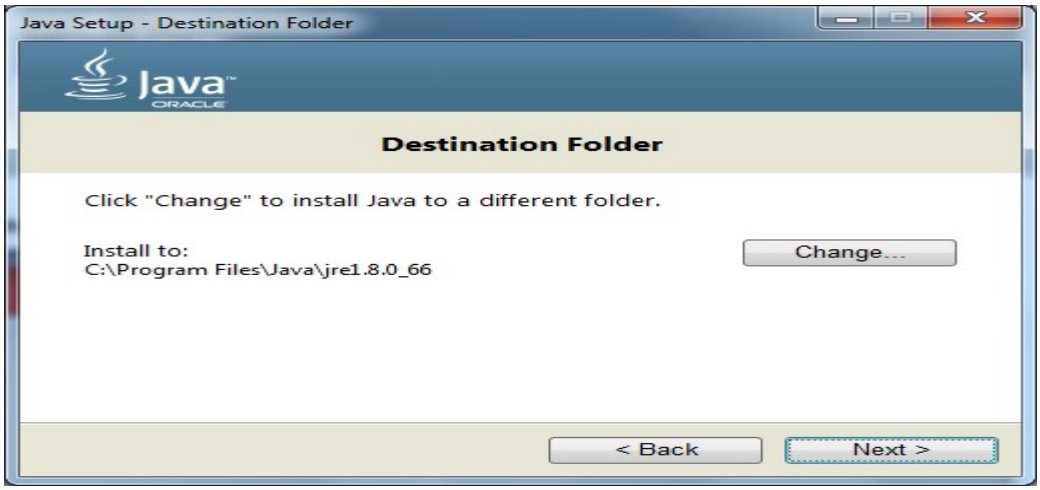
- Click on OK button.
- Following screen will appear.



- Click on NEXT button.
- Following screen will appear. It will install JDK in your machine.

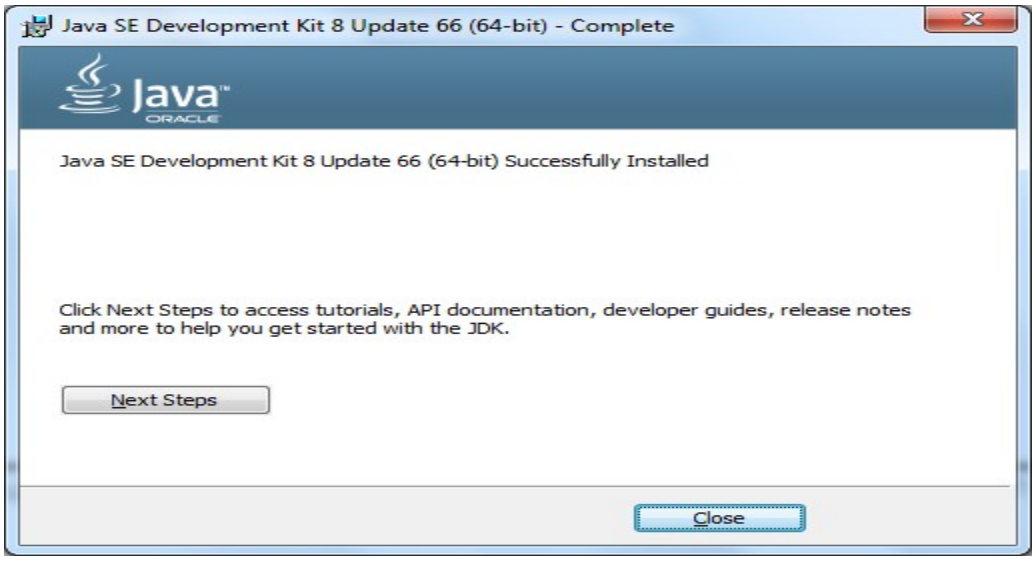


- After some time, Installer will prompt to select destination location for JRE.



- Use the defaults as it is and click on NEXT button then following screen will appear.

- After installation completed successfully,
- Following screen will appear.



- Click on CLOSE button.

1.8. Directory Structure

- After installing JDK you will get the following directory structure

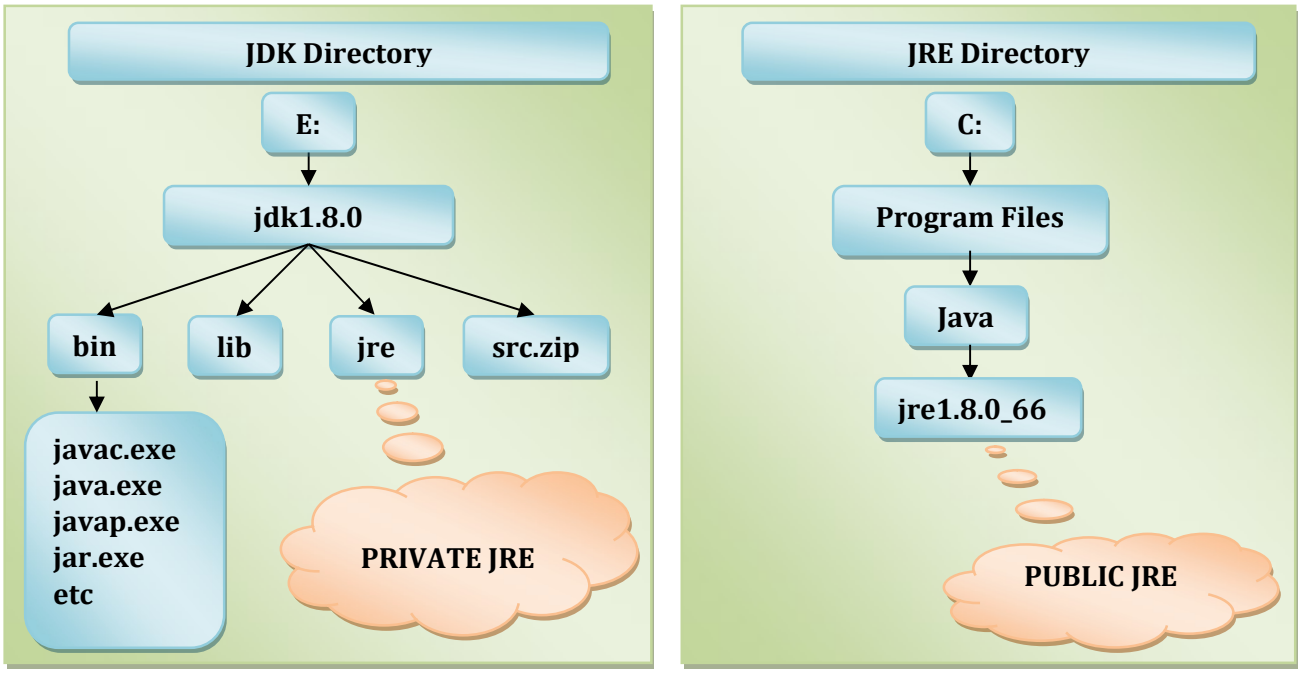


Diagram – 4

1.9. Steps to Write, Compile and Run Java Program

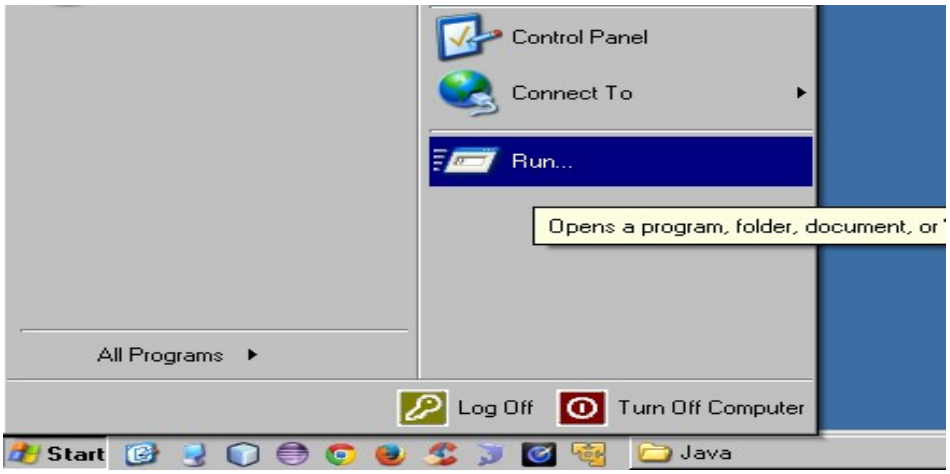
1.9.1. Steps to Write Java Program

A *workspace* is a directory on your hard drive (file system) where the files related to your project will be stored.

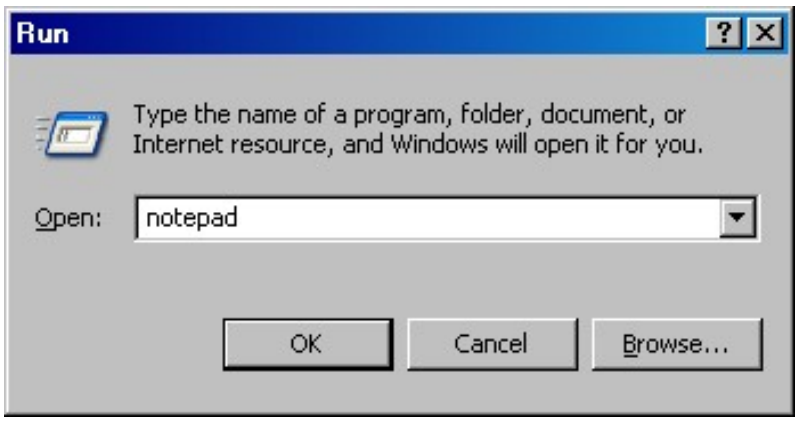
- Select the required WORKSPACE
- Assume we have selected workspace as
  D:\JLC\Core
- Use NOTEPAD or some other text editor to create the source file.

Steps

1. Click on Start -> Run

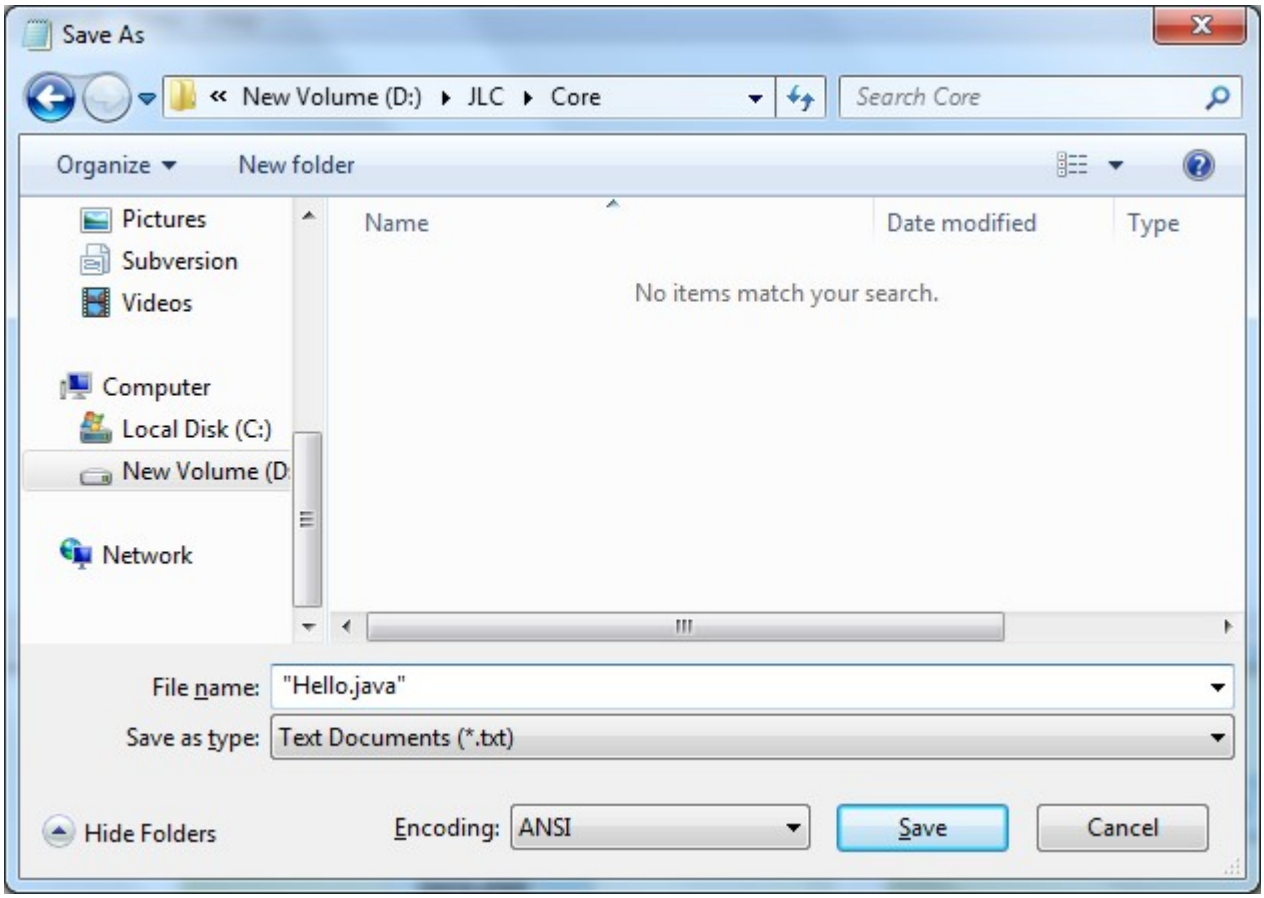


2. Type notepad and click OK

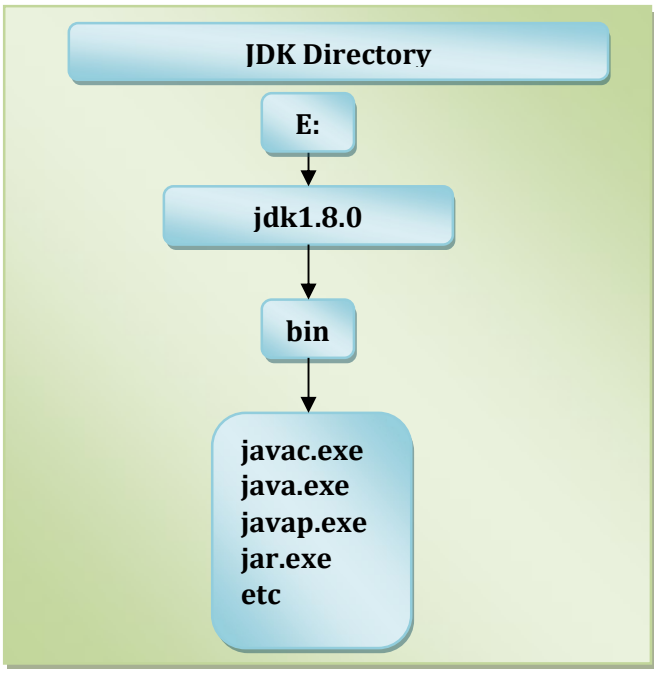


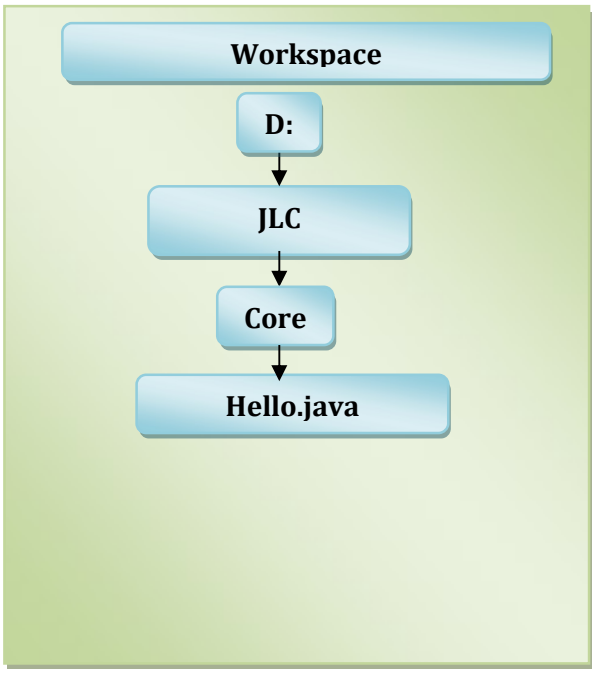
3. Write the following code in the text editor.

4. Save the file with the name Hello.java in the Directory D:\JLC\Core.

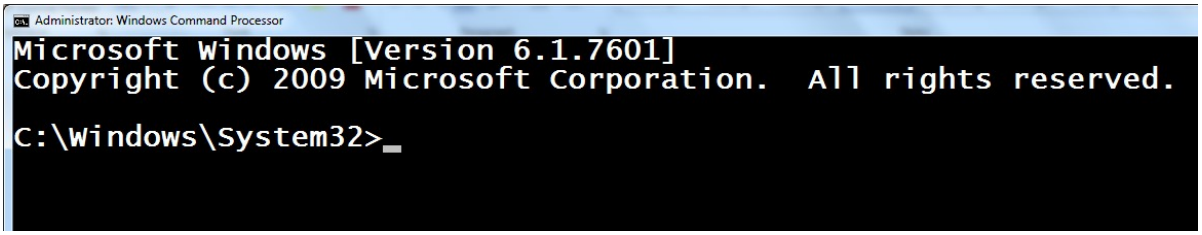


1.9.2. Steps to Compile Java Program



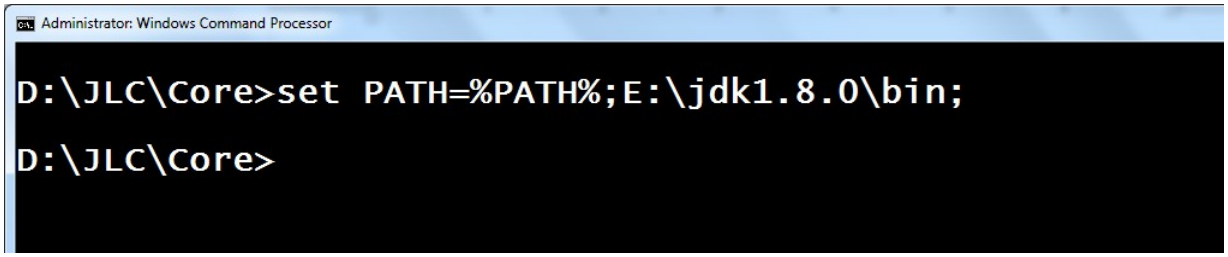


1. Open Command Prompt as follows
    a. Click on Start ->Run
    b. Type command or cmd
    c. Click on OK

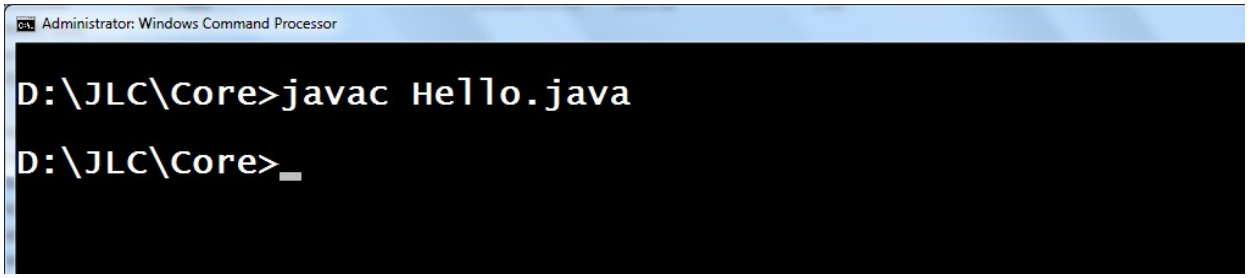


2. Change Working Directory to Workspace Directory (D:\JLC\Core)

3. Set the PATH for bin directory of JDK.



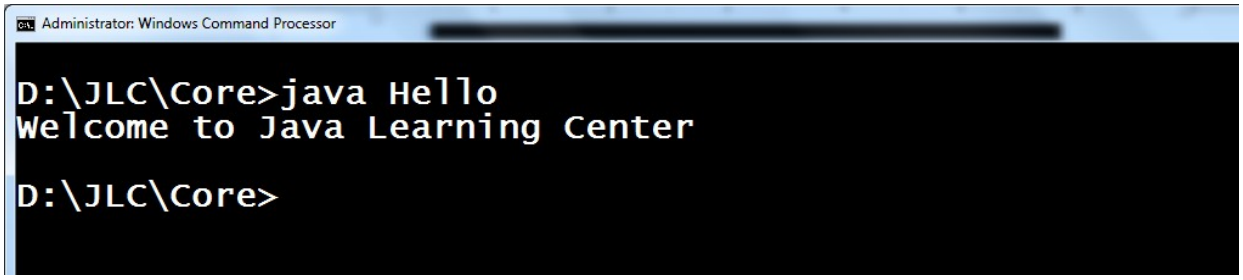
4. Use javac compiler to compile the source code.
   javac <sourceFileNameWith .java>



Compiler will generate Hello.class file in the location D:\JLC\Core

1.9.3. Steps to run Java Program

1. Use java interpreter to execute/run the java class.
   java <classFileNameWithout .class>



- If you set the path at command line then that path will be accessed as long as that command prompt is running. Even that path will not be accessed with other command prompt running at a time. So it is temporary.

You can set the PATH permanently in System Environment Variables as follows.

- Right Click on This PC Area and Select Properties
- From Left Side Panel, Click on Advanced system settings
- Click on Environment Variables Button
- Click on Select PATH and Click on Edit Button
- Add the following

  E:\JDK1.8.0\bin;
- Click on OK button in all the windows.

1.10. Comments

1) Single Line Comments (//)

```
// the whole line will be ignored by the compiler.
```

2) Multi Line Comments (/* */)

```
/*      you can define
        as many line as
        you want
        to ignore
        from compilation.
*/
```

3) Documentation Comments

```
/**
* Will be used
* to define documentation
* of any variable, methods etc
* in Java Source File.
*/
```

1.11. Assignment

Q1) State whether the following statements are true or false?

SNO	Question	Answer
A	C program is platform independent.	**False**
B	Java program is platform independent.	**True**
C	Java Compiler is platform independent.	**False**
D	Java Interpreter is platform dependent	**True**
E	JVM is platform dependent.	**True**
F	JRE is platform independent.	**False**
G	We can use Byte code in C Language.	**False**
H	Native code will be understood by Machine directly.	**True**
I	Byte code will be understood by Machine directly.	**False**
J	You can use "WORA" for C and C++ .	**False**

Q2) I have written a Java program and compiled with Java 9.
Can I execute this with JRE 8?

Ans:

No, You can Not Run Java9 compiled program with Java8
You will get Error called UnsupportedClassVersionError.

Q3) I have written a Java program and compiled with Java 8.
Can I execute this with JRE 9?

Ans:

Yes, You can Run Java8 compiled program with Java9

Q4) How can I check the version of Java compiler I am using?

Ans:

javac -version

Q5) How can I check the version of JRE I am using?

Ans:

java -version

Q6) What is the reason for the following error?

```
'javac' is not recognized as an internal or external command,
operable program or batch file.
```

Ans:

javac.exe file is not available in the current folder and also not available in the path.

Q7) What is the reason for the following error?

```
javac: file not found: Hello.java
```

Ans:

Hello.java is not available in the current folder

Q8) What is the reason for the following error?

```
Error: Could not find or load main class Hai
```

Ans:

Hai.class is not available in the current folder and also not available in te Classpath.

Q9) What is the reason for the following error?

```
java.lang.NoClassDefFoundError: hello (wrong name: Hello)
```

Ans:

Hello.class is found but you are using the wrong name.

Q10) What is the reason for the following error?

```
java.lang.ClassFormatError: Incompatible magic value 4
```

Ans:

Byte Code is also called as Madic Value.

When Byte Code is manipulated the you will get Incompatiable magic value Error.

Q11) What is the reason for the following error?

```
java.lang.ClassFormatError: Truncated class file
```

Ans:

When Byte Code is fully removed from .class file then you will get

ClassFormatError: Truncated class file

Q12) What is the reason for the following error?

```
java.lang.UnsupportedClassVersionError: Hello
```

Ans:

When you are trying run the Higher Java version compiled butecode with Lower JRE then you will get UnsupportedClassVersionError

Q13) What is the use of setting the PATH?

Ans:

To access the Binary files from one location to any other locations , You need to set the path for that Binary files.

Q14) What are the ways available to set the PATH?

Ans:

There are two ways to set the path.

a) Temporarly at Command Line
b) Permanently at System Environment Variables.

Q15) What is the use of setting the ClassPATH?

Ans:

To access the .class files from one location to any other locations, You need to set the classpath for that .class files.

Q16) What are the ways available to set the ClassPATH?

Ans:

There are two ways to set the ClassPATH.

a) Temporarly at Command Line
b) Permanently at System Environment Variables.

Q17) What is the difference between PATH and ClassPATH?

Ans:

Path is used to access the .exe files where as ClassPATH is used to access the .class files

Q18) What is the difference between PRIVATE JRE and PUBLIC JRE?

Ans:

JRE which installed under JDK is Private JRE where as JRE which installed under Program Files is Public JRE

Q19) Which JRE will be used while compiling the Java Source File?

Ans:

First Private JRE will be used.

If Private JRE is not found the Public JRE will be Used.

Q20) Which JRE will be used while executing Java application?

Ans:

Public JRE will be Used also

Q21) Can we execute java program without setting PATH?

Ans:

Yes, But Not Recomanded

Q22) What is the difference between JVM and JRE?

Ans:

JRE is an Implemetaton of JVM Specification.

Oracle JRE, IBM JRE,Open JDK etc

Q23) What is the difference between JIT Compiler and Java Interpreter?

Ans:

- ✓ Java Interpreter coverts the Byte code to Native Code line by line and then executes the Native Code Line by line
- ✓ JIT Compiler coverts the Bunch of Byte code to Native Code once and then executes the Bunch of Native Code once.
- ✓ So JIT Compiler is faster than Java Interpreter
- ✓ Java Interpreter is used before Java2.
- ✓ After Java2, Java Interpreter is replced with JIT Compiler.

Q24) What is the difference between Byte code and Native code?

Ans:

Byte code is JVM Understandable Code where as Native Code is Machine Understandable Code

Q25) Is it possible to execute Java program without having JDK in machine?

Ans:

Yes

Q26) Can I install multiple JDK versions in my machine?

Ans:

Yes

Q27) Can I install multiple JRE versions in my machine?

Ans:

Yes

Q28) Can I install only JRE in my machine?

Ans:

Yes

Q29) Can I use different names for File and Class?

Ans:

Yes

Q30) Can I write multiple classes in One Source File?

Ans:

Yes

Q31) Can I write main method as follows?

public static void main(String args)

Ans:

NO

Q32) What is the signature of standard main() method?

Ans:

public static void main(String[] args)

Q33) What is the difference between C++ and Java?

Ans:

Java	C++
1. Java doesn't support Pointers.	1. C++ supports Pointer.
2. Java does not support multiple inheritance with classes.	2. C++ supports multiple inheritance with classes.
3. Java doesn't support Global Variables (Variables declared outside the class).	3. C++ supports Global Variables.
4. Java doesn't support header files.	4. C++ supports header files.
5. const and goto keywords can't be used in Java.	5. const and goto keywords can be used in C++.
6. Java doesn't support Destructors.	6. C++ supports Destructors.
7. Java doesn't support Virtual Functions.	7. C++ supports Virtual Functions.
8. Java doesn't support Operator Overloading.	8. C++ supports Operator Overloading.
9. Java doesn't support Scope Resolution Operator.	9. C++ supports Scope Resolution Operator (::).
10. Java has Built-in API for Multithreading and Network Programming.	10. C++ doesn't have Built-in API for Multithreading and Network Programming.
11. Java has a Garbage Collector to clean the memory automatically.	11. No automatic memory cleaning process in C++.