Introduction to Java

Shakir Hussain

Contents

What is Java?

- High level programming language
- Originally developed by Sun Microsystems (now, Oracle), Which was initiated by James Gosling
- Designed with a concept of write once and run anywhere
- First version of java released in 1995
- Initial name was Greentalk, later renamed to Oak and finally Java
- Java is a Platform

Brief history of Java

James Gosling

https://dzone.com/articles/a-short-history-of-java

https://www.javatpoint.com/history-of-java

The Java Programming Language

Java Language - Buzzwords

- Platform Independent (architecture neutral)
  - *Write once run anywhere*
- Simple
  - *Small language, large libraries*
- Object Oriented
  - *Supports Abstraction, Encapsulation, Polymorphism, Inheritance etc.*

Java Language - Buzzwords..

- Auto Garbage Collection
  - *Memory management handled by Java Virtual Machine*

- Secure
  - *No memory pointers, program run inside virtual machine*
  - *Java Bytecode verification*
  - *Array Index limit checking*

Java Language – Buzzwords…

- Portable
  - *Primitive data type size and their arithmetic behavior are specified by the language.*
  - *Libraries define portable interfaces*
- Distributed
  - *Libraries for network programming*
  - *Remote method invocation*

Java Language – Buzzwords….

- Multithreaded
  - *Easy to create and use*

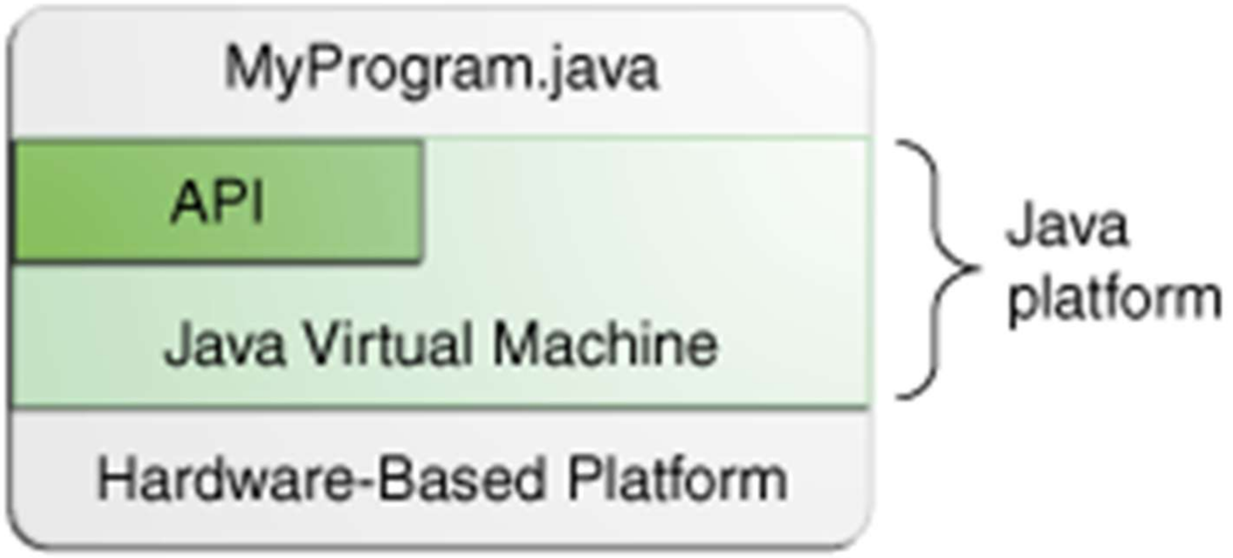
- Robust
  - *Strong memory mgmt.*

- Dynamic
  - *Finding runtime type information is easy.*
  - *The linking of data and methods to where they are located, is done at run-time.*
  - *New classes can be loaded while a program is running. Linking is done on the fly.*

The Java Platform

- Software-only platform that runs on top of other hardware-based platforms.

- Environment in which java program runs.

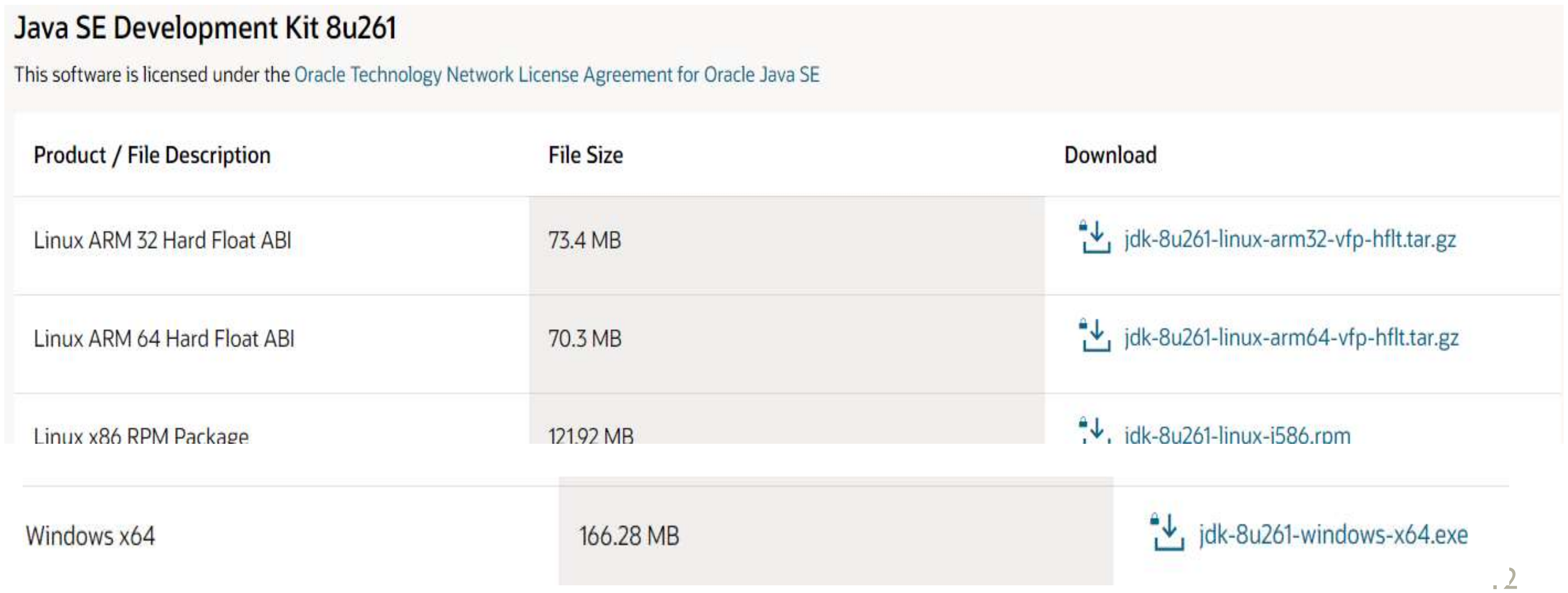


Setup Dev Environment

- What do you need to write and run java program?
  - Java Development Kit (JDK)
  - ASCII Text Editor

Download JDK

- https://www.oracle.com/in/java/technologies/javase/javase-jdk8-downloads.html
- Go to the below section, download platform specific installer



Java SE Development Kit 8u261

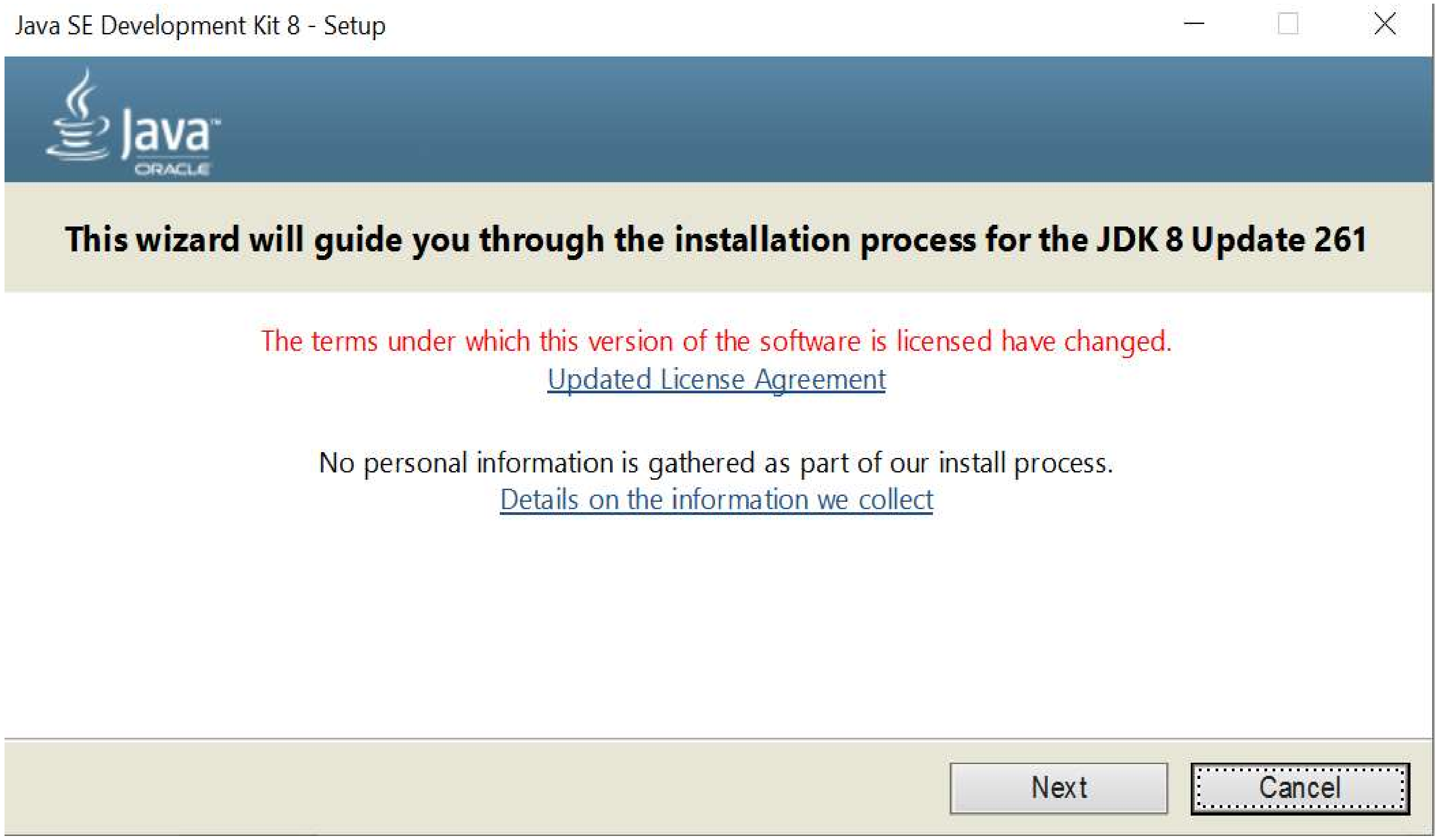
This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	73.4 MB	jdk-8u261-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	70.3 MB	jdk-8u261-linux-arm64-vfp-hflt.tar.gz
Linux x86 RPM Package	121.92 MB	jdk-8u261-linux-i586.rpm
Windows x64	166.28 MB	jdk-8u261-windows-x64.exe

Install Steps

- Once downloaded (e.g. Jdk-8u261-windows-x64.exe) , click to install.
- Steps for windows JDK will be shown in next slides

Install Steps – 1st Window



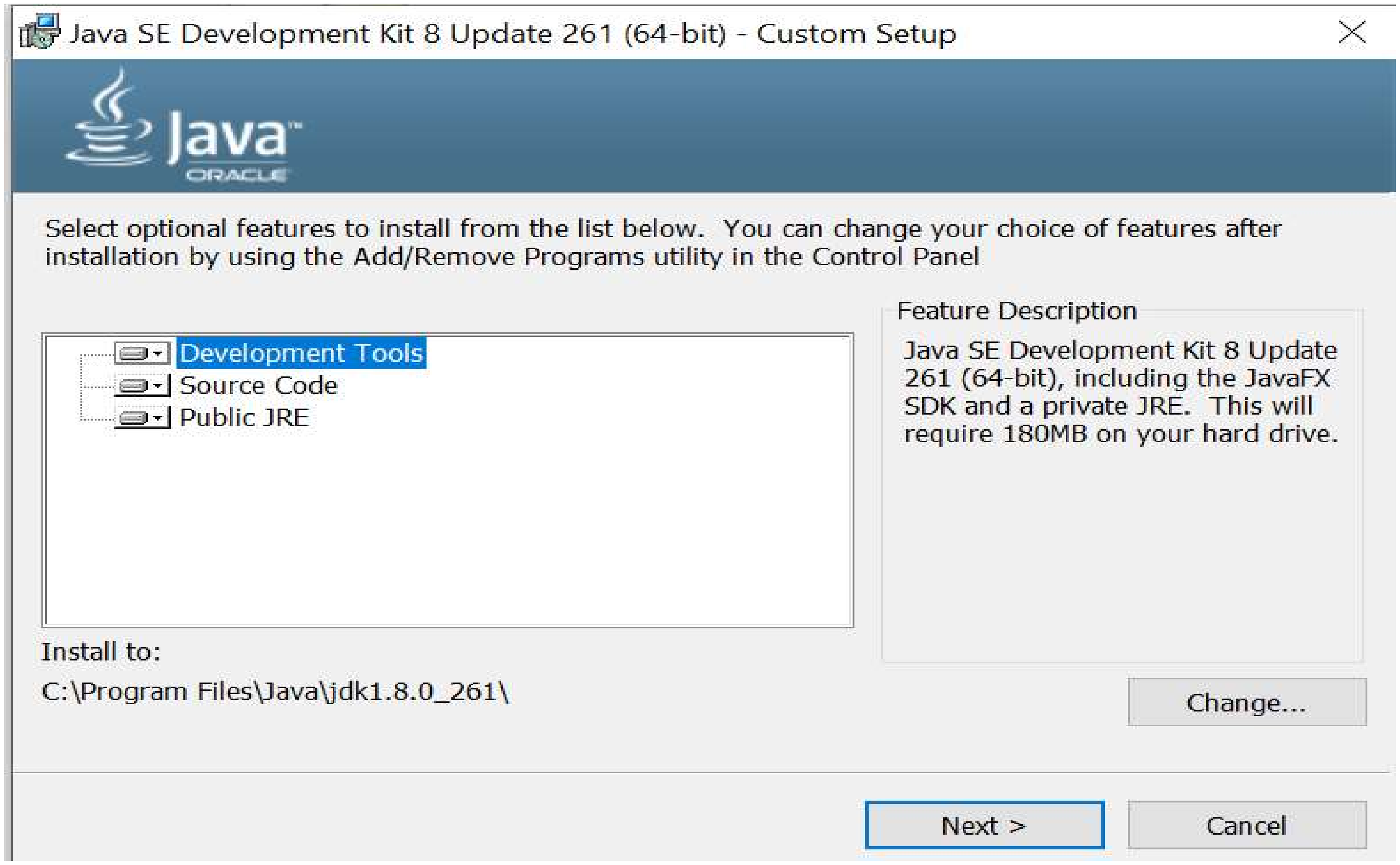
Java SE Development Kit 8 - Setup

This wizard will guide you through the installation process for the JDK 8 Update 261

The terms under which this version of the software is licensed have changed.
Updated License Agreement

No personal information is gathered as part of our install process.
Details on the information we collect

Install Steps – 2nd Window


Java SE Development Kit 8 Update 261 (64-bit) - Custom Setup

Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel

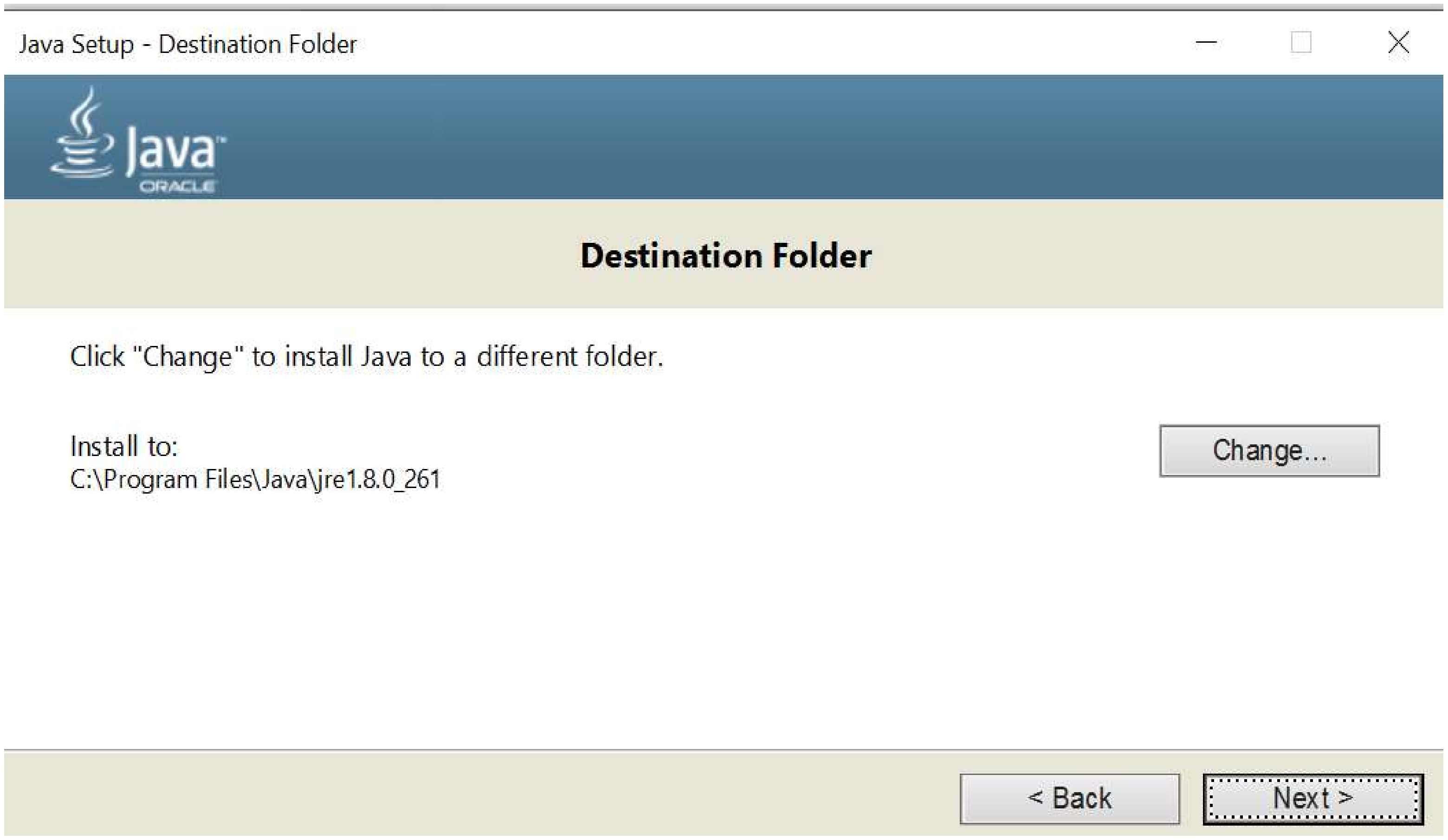
Feature Description

Java SE Development Kit 8 Update 261 (64-bit), including the JavaFX SDK and a private JRE. This will require 180MB on your hard drive.

Install to:

C:\Program Files\Java\jdk1.8.0_261\

Install Steps – 3rd Window

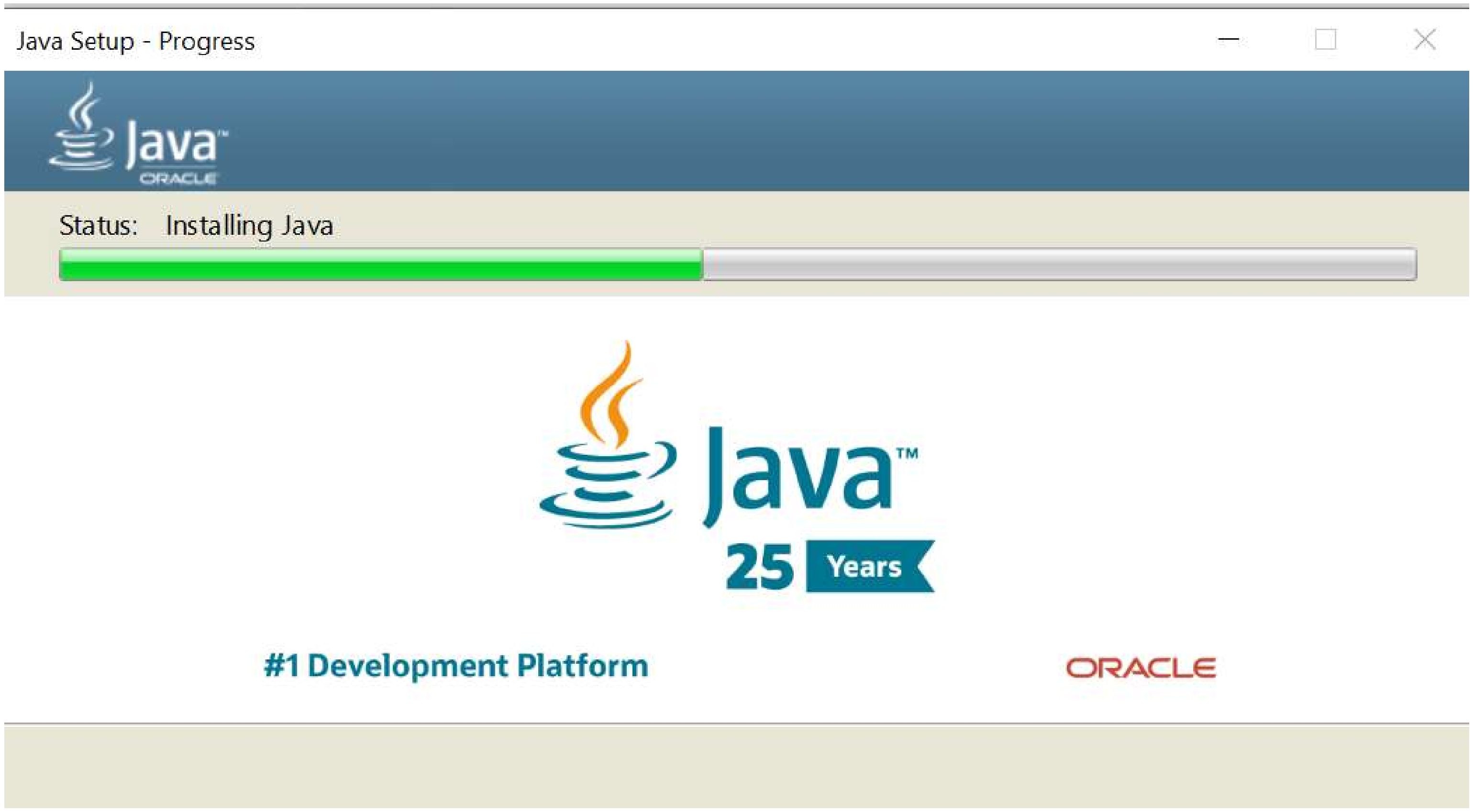

Java Setup - Destination Folder

Destination Folder

Click "Change" to install Java to a different folder.

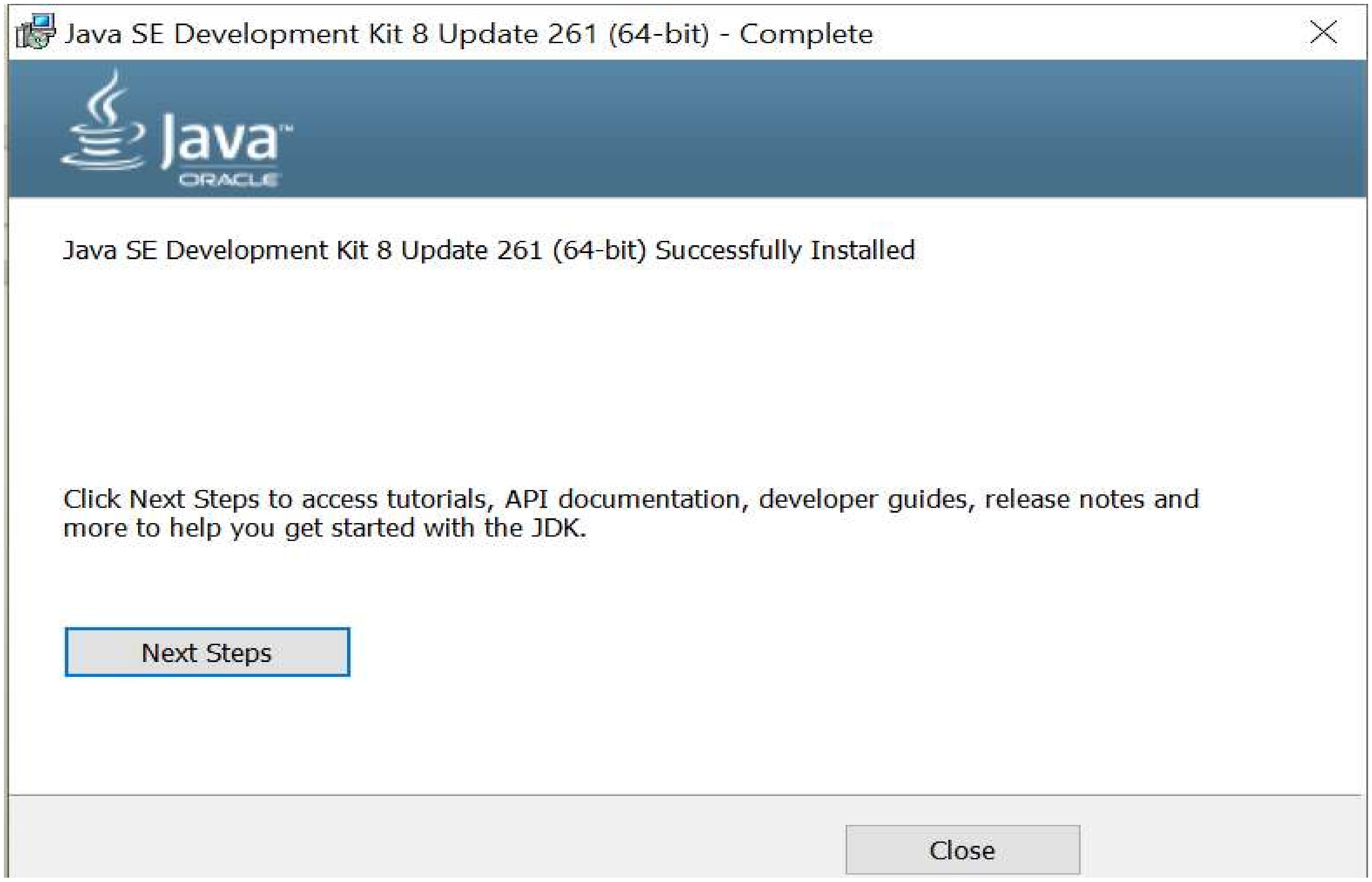
Install to:
C:\Program Files\Java\jre1.8.0_261

Install Steps - 4th Window



Status: Installing Java

Install Steps – 5th Window

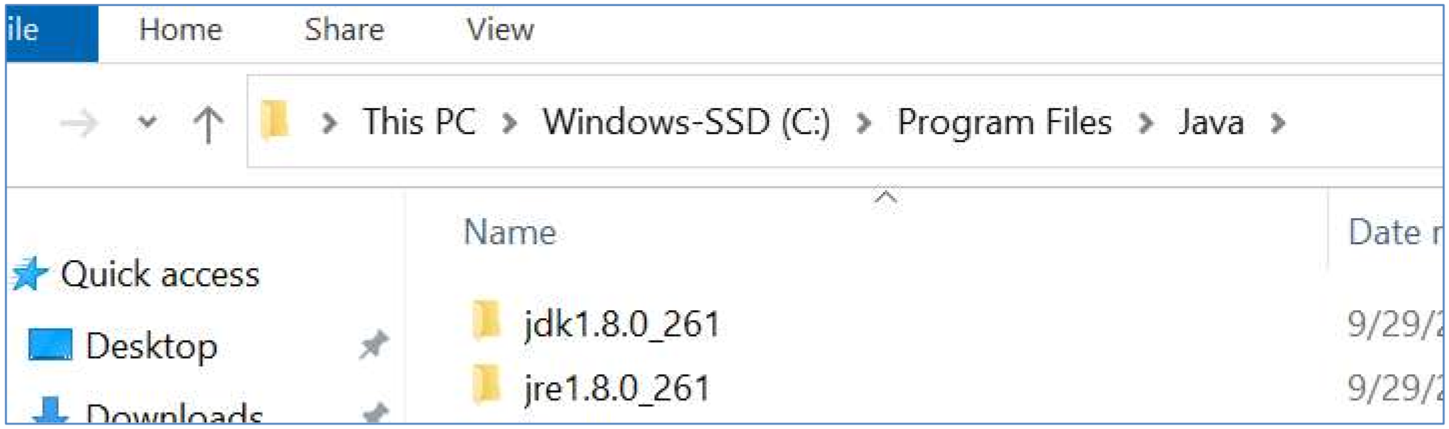

Java SE Development Kit 8 Update 261 (64-bit) - Complete

Java SE Development Kit 8 Update 261 (64-bit) Successfully Installed

Click Next Steps to access tutorials, API documentation, developer guides, release notes and more to help you get started with the JDK.

Setup

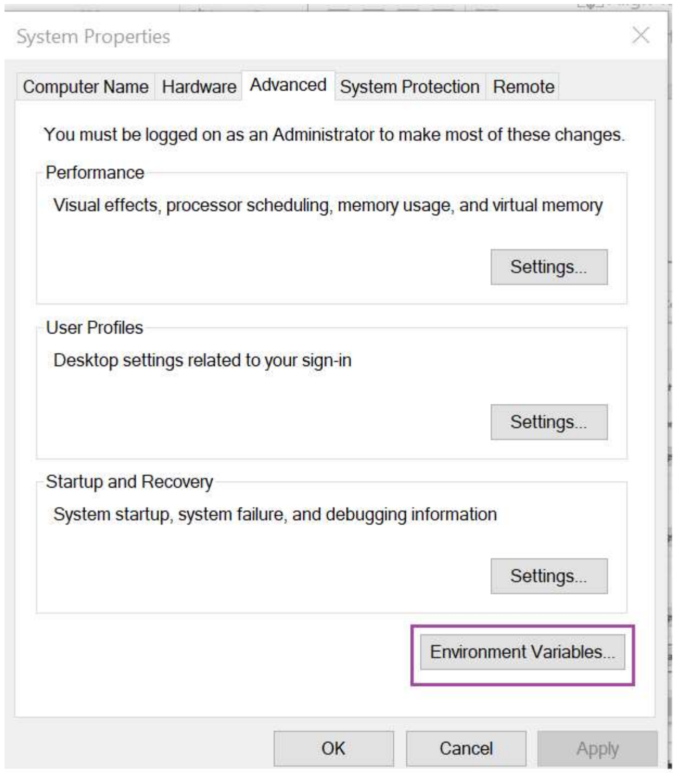
- Verify JDK and JRE on your hard disc.



jdk1.8.0_261 9/29/
jre1.8.0_261 9/29/

- Now, Set the path environment variable as explained in next slide.

Setup..



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

Settings...

User Profiles

Desktop settings related to your sign-in

Settings...

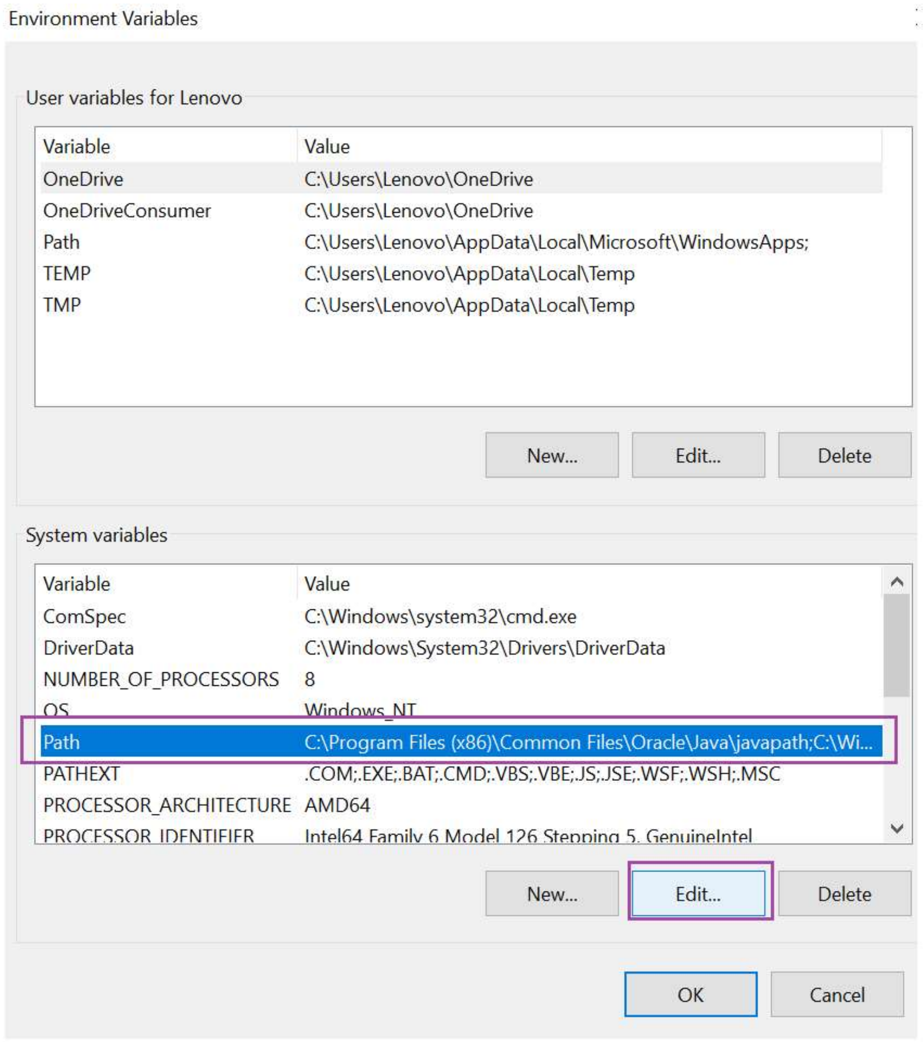
Startup and Recovery

System startup, system failure, and debugging information

Settings...

Environment Variables...

OK | Cancel | Apply



Environment Variables

User variables for Lenovo

Variable	Value
OneDrive	C:\Users\Lenovo\OneDrive
OneDriveConsumer	C:\Users\Lenovo\OneDrive
Path	C:\Users\Lenovo\AppData\Local\Microsoft\WindowsApps;
TEMP	C:\Users\Lenovo\AppData\Local\Temp
TMP	C:\Users\Lenovo\AppData\Local\Temp

New... | Edit... | Delete

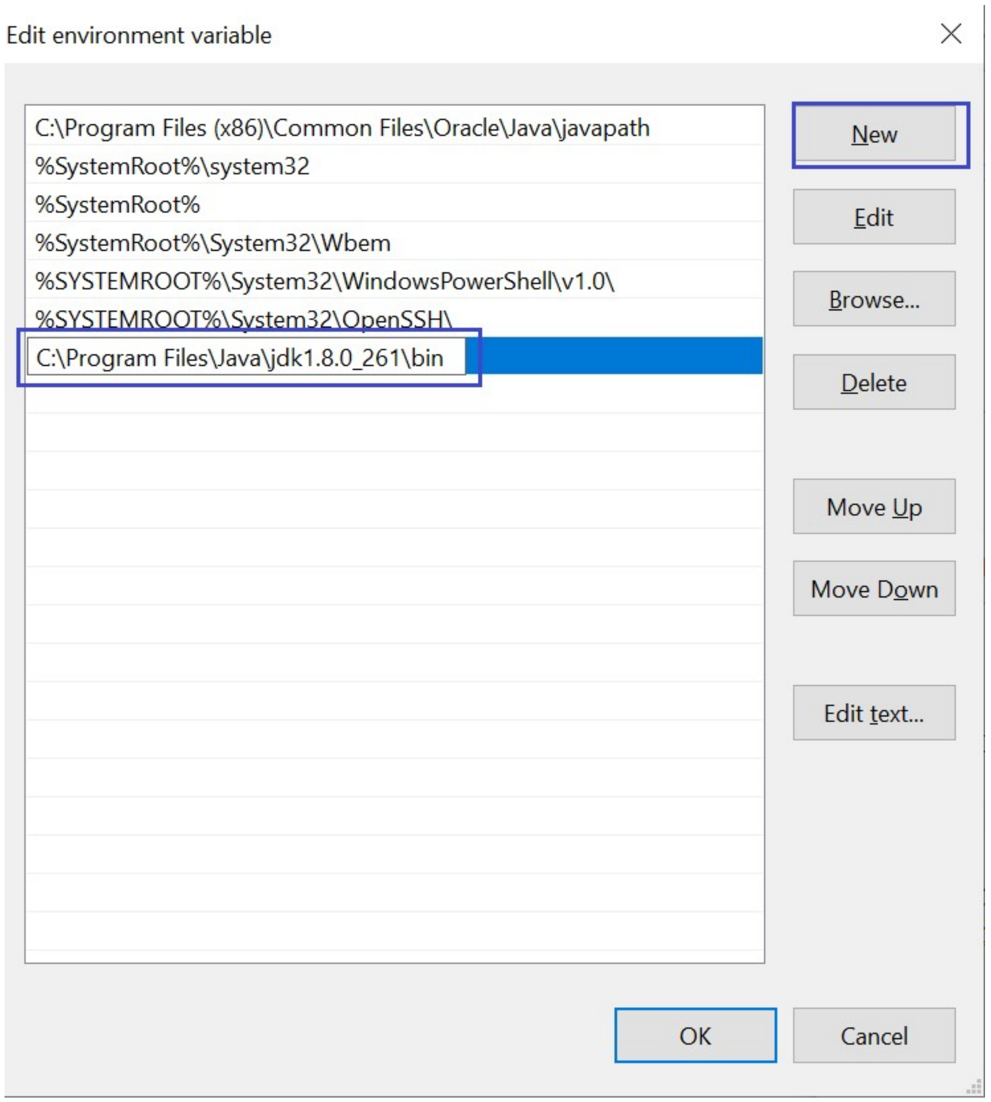
System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Wi...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 126 Stepping 5, GenuineIntel

New... | Edit... | Delete

OK | Cancel

Setup…

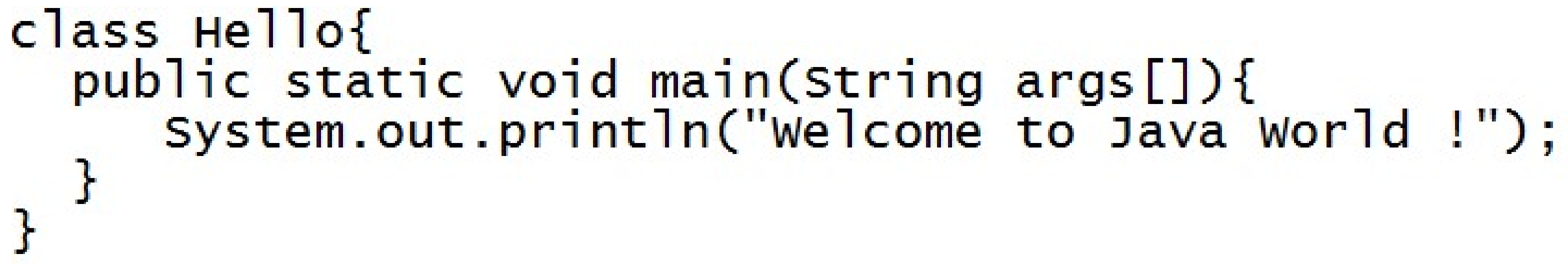

Edit environment variable

```
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Java\jdk1.8.0_261\bin
```

First Java Program

- What do you need?
  - Java Development Kit (JDK)
  - A Text Editor

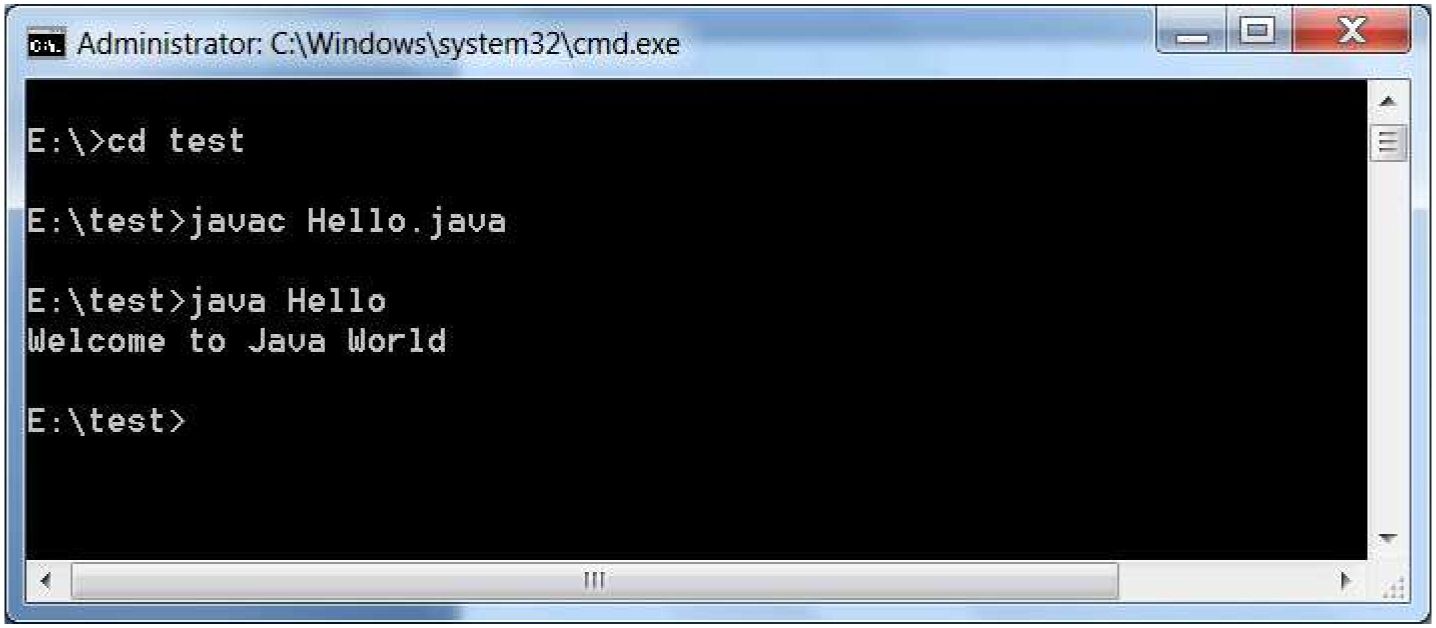
- Open the notepad, write the following code and save the file as Hello.java (Let's say file is saved as E:/test/Hello.java)



```
class Hello{
  public static void main(String args[]){
      System.out.println("Welcome to Java World !");
  }
}
```

First Java Program

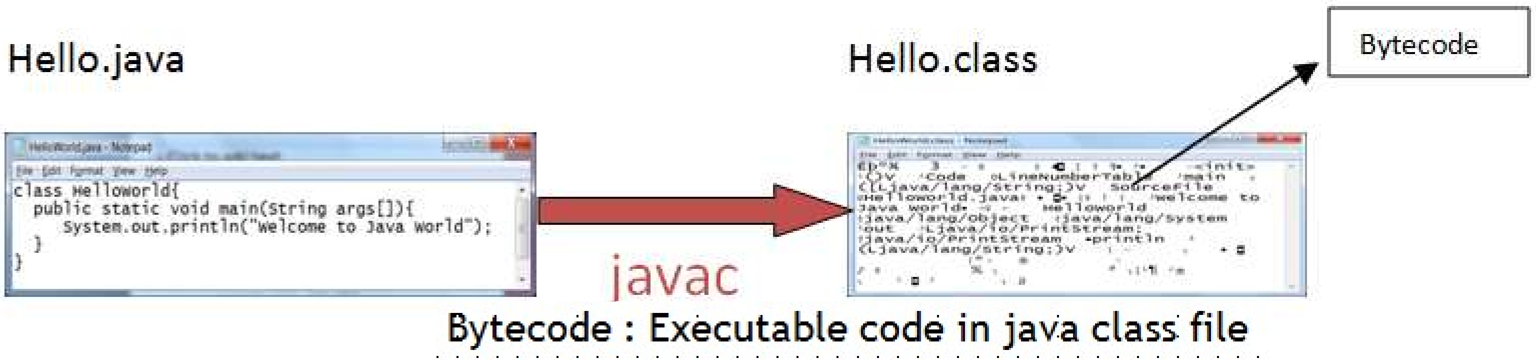
- Now, open command prompt, change directory to the folder where you have saved your file.
- Compile (javac Hello.java)
- Run (java Hello)



```
E:\>cd test

E:\test>javac Hello.java

E:\test>java Hello
Welcome to Java World

E:\test>
```

Java Environment

- Compilation
  - Java Compiler translates the java source file into the class file
  - Class file contains bytecodes, the "machine language" of the java Virtual machine (JVM).
  - This java class file can run on any hardware platform and Operating System, which hosts JVM.

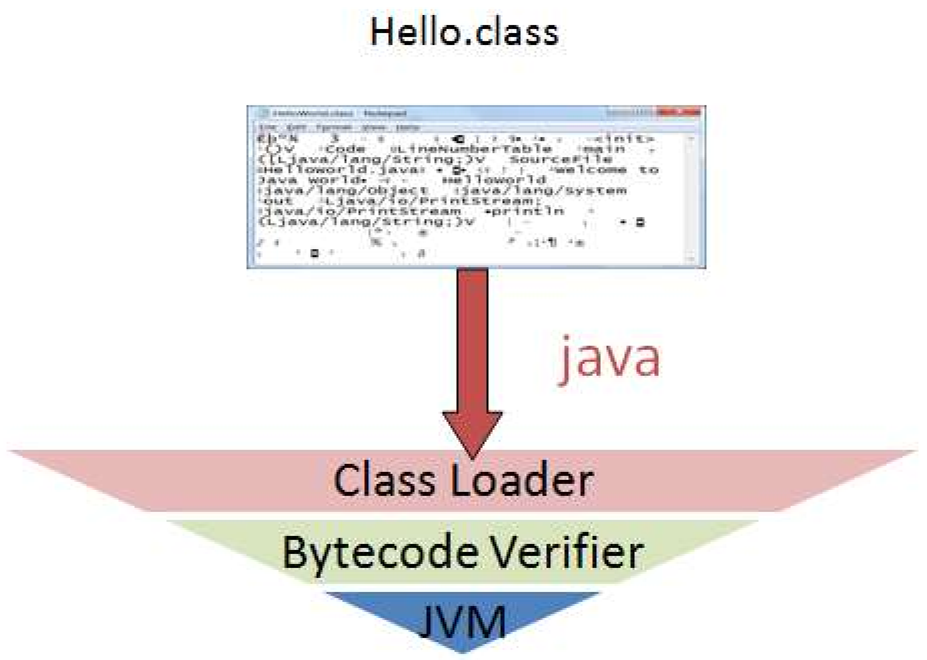


```
class HelloWorld{
   public static void main(String args[]){
      System.out.println("Welcome to Java World");
   }
}
```

Bytecode : Executable code in java class file

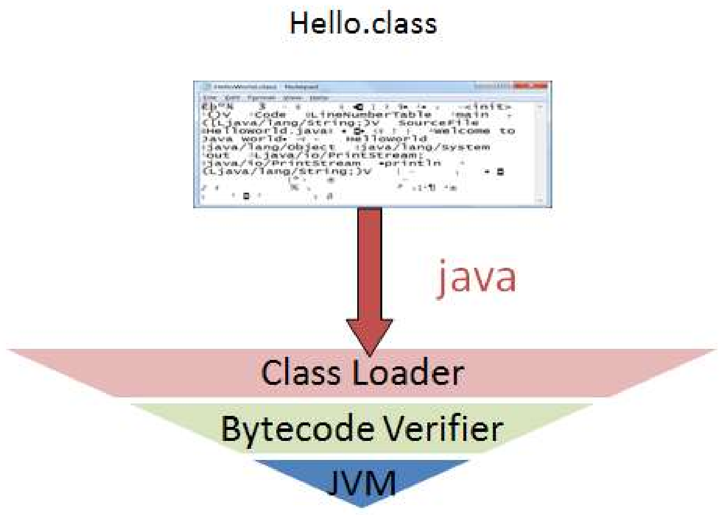
Java Environment

- Class Loader
  - Class Loader loads class files into memory.
  - Class file can be loaded from local disc, network or over the internet



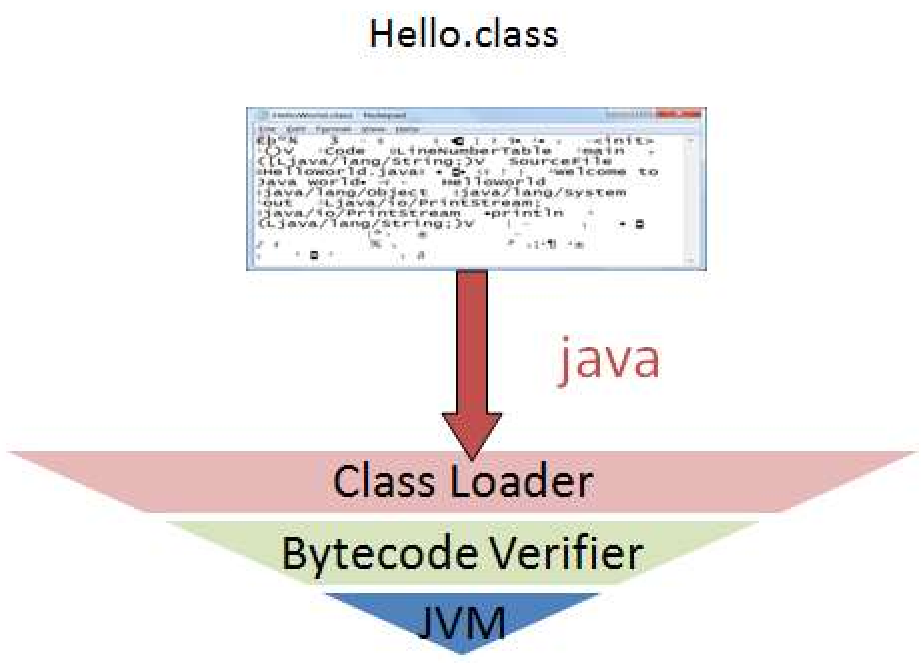
Java Environment

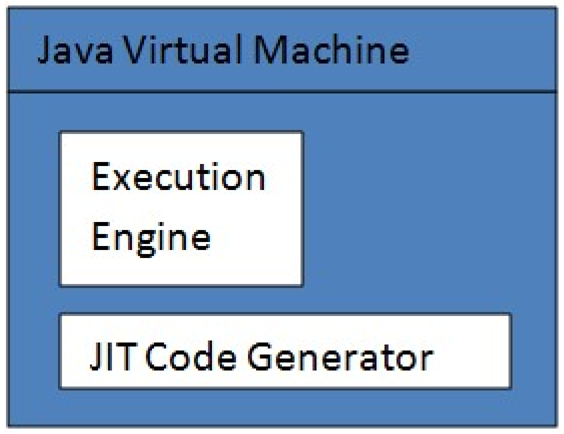
- Bytecode Verifier
  - It verifies that the bytecodes are valid and safe.
  - Does not violate java security restrictions
  - Checks the internal consistency of the class and validity of the code



Java Environment

- Java Virtual Machine
  - Translates the byte code into machine code depending upon the underling operating system and hardware combination. Which later executed by processor.

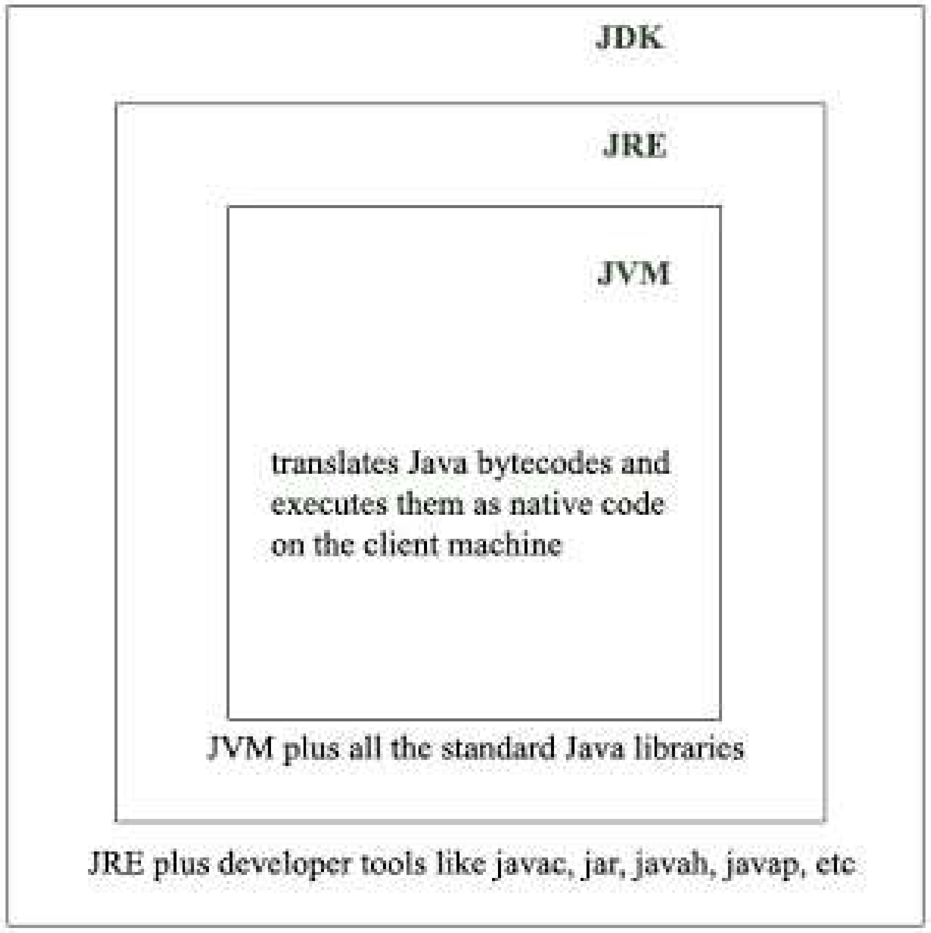




Java Terminology

- JDK : Java Development Kit
  - All you need to develop, compile, debug and run your java program.
- JRE : Java Runtime Environment
  - Subset of JDK
  - Includes Minimum Elements required to run java class file
  - Does not contain development tools like compiler, debugger etc.
- JVM : Java Virtual Machine
  - Actually runs the java program and uses the library and other supporting files provided by JRE

Java Terminology


JDK
JRE
JVM
translates Java bytecodes and executes them as native code on the client machine
JVM plus all the standard Java libraries
JRE plus developer tools like javac, jar, javah, javap, etc

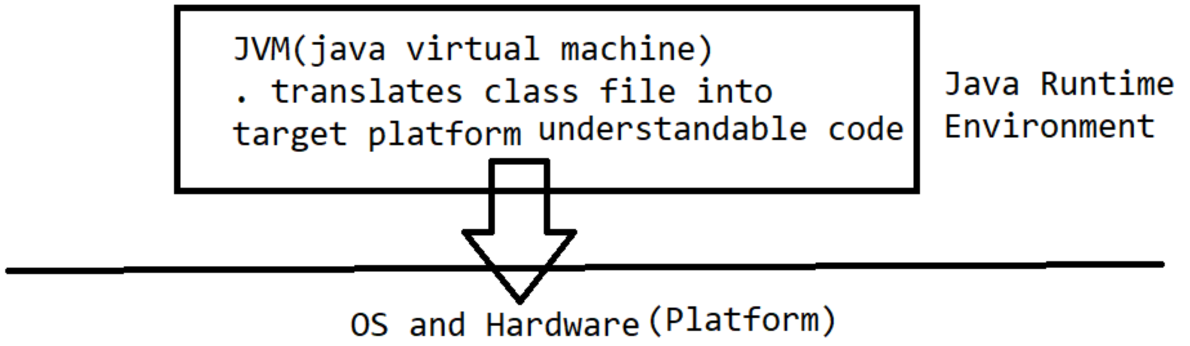
Points to remember

- Java Compiler, JVM ..All are platform dependent.

- Only java class file (Bytecode) is platform independent.

QA

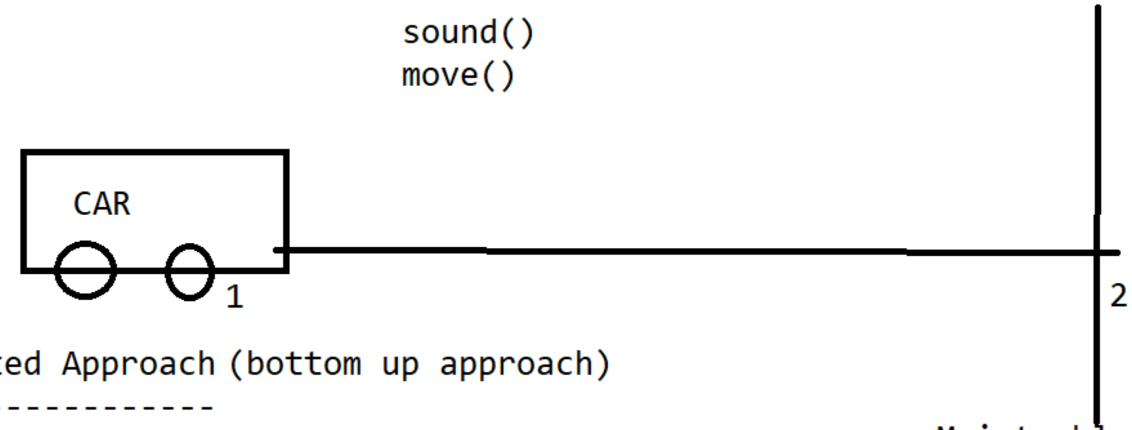
java Hello


JVM(java virtual machine)
. translates class file into
target platform understandable code
Java Runtime
Environment
OS and Hardware (Platform)

Problem : Write a program to move a car from point 1 to point 2

```
//Procedure (Function/Method) top-down
Car - how to draw car
void move(Object v, Point p1, Point 2){
   if(v is Car) move like car;
   if(v is Bus) move like Bus;
   ...Train
   ...Aeroplane
   ... bike
}
```

sound()
move()

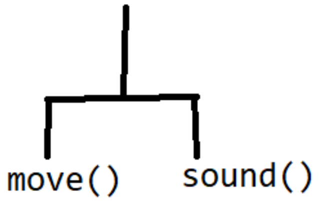


```
// single threaded(1 exec path)
void main(){
    move();
    sound();
}

void main(){
   run move() as well sound();
}
```

Maintanble
Exentable
Simpler

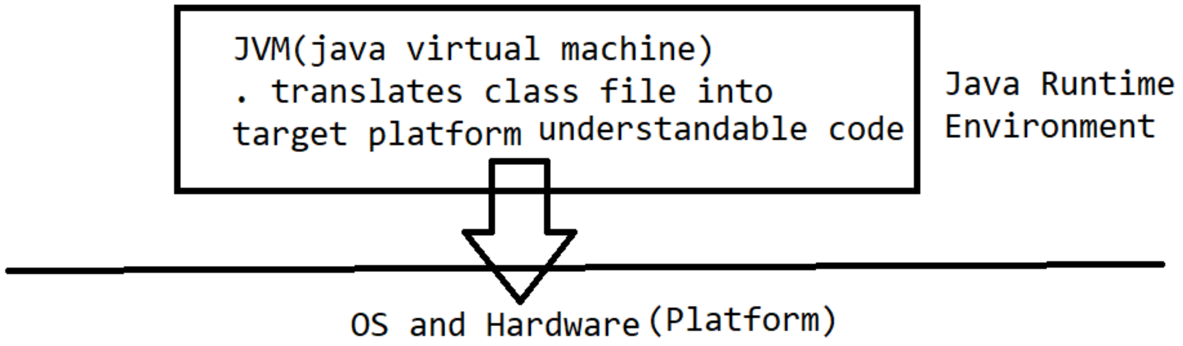
Object Oriented Approach (bottom up approach)

```
class Vehicle{....common functionaltiy...}
class Car extends Vehicle{

  move() {...}
}

class Bus extends Vehicle{
  move() {....}

}
```



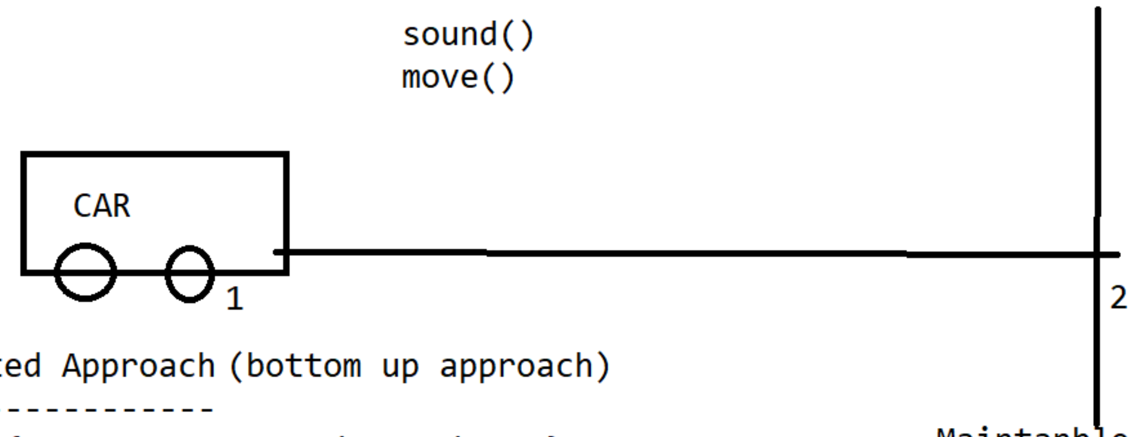
OO Concepts:
Abstraction, Encapsulation, Inheritance, Polymorphism

java Hello


JVM(java virtual machine)
. translates class file into
target platform understandable code
Java Runtime
Environment
OS and Hardware (Platform)

Problem : Write a program to move a car from point 1 to point 2

sound()
move()



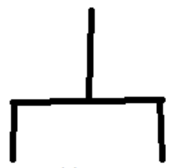
```
//Procedure (Function/Method) top-down
Car - how to draw car
void move(Object v, Point p1, Point 2){
   if(v is Car) move like car;
   if(v is Bus) move like Bus;
   ...Train
   ...Aeroplane
   ... bike
}
        // single threaded(1 exec path)
        void main(){
            move();
            sound();
        }

        void main(){
           run move() as well sound();
        }
```

Object Oriented Approach (bottom up approach)

```
class Vehicle{....common functionaltiy...}
class Car extends Vehicle{

  move() {...}
}

class Bus extends Vehicle{
  move() {....}

}
```

Maintanble
Exentable
Simpler

OO Concepts:
Abstraction, Encapsulation, Inheritance, Polymorphism


move() sound()

Basic Language Elements

Shakir Hussain

Identifiers

- A name in a program is called an Identifier.
- Used to denote classes, methods, variables and labels.

- Each character can be either `letter` or a `digits`
- `First` character in an Identifier must be a `letter` (or underscore or currency symbols like $, but never recommended)

Keywords

- Reserved words that are predefined in the language and can not be used to denote other entities.
- All keywords are in `lower case`.
- A reserved word can not be used as an Identifier.

JAVA KEYWORDS

abstract	continue	for	new	switch
assert	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

RESERVED KEYWORDS (not currently in use)

const	goto			

RESERVED LITERALS

null	true	false		

Literals

- Denotes a constant value, the value that a literal represents remains unchanged in the program.

- Example :

```
Boolean literals : true, false
Integer literals :  100, 200 etc.
```

Primitive Types and Variables

- boolean, char, byte, short, int, long, float, double etc.
- These basic (or primitive) types are the only types that are not objects (due to performance issues).
- This means that you don't use the new operator to create a primitive variable.
- Declaring primitive variables:

```
float initVal;
int retVal, index = 2;
double gamma = 1.2, brightness ;
boolean valueOk = false;
```

Initialisation

- If no value is assigned to local variable prior to use, then the compiler will give an error
- Java sets primitive variables to zero or false in the case of a boolean variable[non-local variable]
- All object references are initially set to null
- An array of anything is an object
  - Set to null on declaration

Declarations

```
int index = 1.2;               // compiler error
boolean retOk = 1;             // compiler error
double fiveFourths = 5 / 4;    // no error!
float ratio = 5.8f;            // correct
double fiveFourths = 5.0 / 4.0; // correct
```

- 1.2f is a float value accurate to 7 decimal places.
- 1.2 is a double value accurate to 15 decimal places.

Assignment

- All Java assignments are right associative

```
int a = 1, b = 2, c = 5;
a = b = c;
System.out.print(
"a= " + a + "b= " + b + "c= " + c)
```

- What is the value of a, b & c
- Done right to left: `a = (b = c);`

Basic Mathematical Operators

- `* / % + -` are the mathematical operators
- `* / %` have a higher precedence than `+` or `-`

```
double myVal = a + b % d - c * d / b;
```

- Is the same as:

```
double myVal = (a + (b % d)) -
                           ((c * d) / b);
```

Statements & Blocks

- A simple statement is a command terminated by a semi-colon:

```
name = "Fred";
```

- A block is a compound statement enclosed in curly brackets:

```
{
   name1 = "Fred"; name2 = "Bill";
}
```

- Blocks may contain other blocks

Flow of Control

- Java executes one statement after the other in the order they are written
- Many Java statements are flow control statements:

```
Alternation:   if, if else, switch
Looping:       for, while, do while
Escapes:       break, continue, return
```

If – The Conditional Statement

- The if statement evaluates an expression and if that evaluation is true then the specified action is taken

```
if ( x < 10 ) x = 10;
```

- If the value of x is less than 10, make x equal to 10
- It could have been written:

```
if ( x < 10 )
x = 10;
```

- Or, alternatively:

```
if ( x < 10 ) { x = 10; }
```

Relational Operators

==	Equal (careful)
!=	Not equal
>=	Greater than or equal
<=	Less than or equal
>	Greater than
<	Less than

If… else

- The if … else statement evaluates an expression and performs one action if that evaluation is true or a different action if it is false.

```
if (x != oldx) {
  System.out.print("x was changed");
}
else {
  System.out.print("x is unchanged");
}
```

Nested if … else

```
if ( myVal > 100 ) {
  if ( remainderOn == true) {
       myVal = mVal % 100;
  }
  else {
    myVal = myVal / 100.0;
  }
}
else
{
  System.out.print("myVal is in range");
}
```

else if

- Useful for choosing between alternatives:

```
if ( n == 1 ) {
  // execute code block #1
}
else if ( j == 2 ) {
  // execute code block #2
}
else {
  // if all previous tests have failed, execute
  code block #3
}
```

A Warning…

WRONG!

```
if( i == j )
    if ( j == k )

      System.out.print
    (
          "i equals
    k");
     else
      System.out.print(
       "i is not equal
       to j");
```

CORRECT!

```
if( i == j ) {
  if ( j == k )
  System.out.print(
       "i equals k");
}
else
  System.out.print("
  i is not equal to
  j");    //
  Correct!
```

The switch Statement

```
switch ( n ) {
  case 1:
    // execute code block #1
    break;
  case 2:
    // execute code block #2
    break;
    default:
    // if all previous tests fail then
    //execute code block #4
    break;
}
```

The for loop

- Loop n times

```
for ( i = 0; i < n; n++ ) {
  // this code body will execute n times
  // ifrom  0 to n-1
}
```

- Nested for:

```
for ( j = 0; j < 10; j++ ) {
  for ( i = 0; i < 20; i++ ){
    // this code body will execute 200 times
  }
}
```

while loops

```
while(response == 1) {
  System.out.print( "Hello World ");
  n++;
  response = readInt( "Enter response?");
}
```

What is the minimum number of times the loop is executed?

What is the maximum number of times?

do {… } while loops

```
do {
  System.out.print( “Hello World”);
  n++;
  response = readInt( “Enter response?” );
}while (response == 1);
```

What is the minimum number of times the loop is executed?

What is the maximum number of times?

Break

- A break statement causes an exit from the innermost containing while, do, for or switch statement.

```
for ( int i = 0; i < maxID, i++ ) {
  if ( userID[i] == targetID ) {
    index = i;
    break;
  }
}// program jumps here after break
```

Continue

- Can only be used with while, do or for.
- The continue statement causes the innermost loop to start the next iteration immediately

```
for ( int i = 0; i < maxID; i++ ) {
  if ( userID[i] != -1 ) continue;
  System.out.print( "UserID " + i + " :" +
    userID);
}
```

Break with label

```
first:
for( int i = 0; i < 10; i++) {
  second:
  for(int j = 0; j < 5; j ++ ) {
    break myLabel;
  }
}
```

You can replace `myLabel` with first or second.

Similarly , you can have continue with label in java.

QA