

Lecture1

1. History of Java

Java goes back to 1991, when a group of Sun engineers, led by Patrick Naughton and SunFellow (and all-around computer wizard) James Gosling, wanted to design a small computer language that could be used for consumer devices like cable TV switchboxes. Since these devices do not have a lot of power or memory, the language had to be small and generate very tight code. Also, because different manufacturers may choose different central processing units (CPUs), it was important not to be tied down to any single architecture.

The project got the code name “Green.”

The requirements for small, tight, and platform-neutral code led the team to restructure the model that some Pascal implementations tried in the early days of PCs. What Niklaus Wirth, the inventor of Pascal, had pioneered, and UCSD Pascal did commercially, was to design a portable language that generated intermediate code for a hypothetical machine. (These are often called *virtual machines*—hence, the Java Virtual Machine or JVM.) This intermediate code could then be used on any machine that had the correct interpreter.

The Green project engineers used a virtual machine as well, so this solved their main problem.

The Sun people, however, come from a UNIX background, so they based their language on C++ rather than Pascal. In particular, they made the language object oriented rather than procedure oriented. But, as Gosling says in the interview, “All along, the language was a tool, not the end.”

Gosling decided to call his language “Oak.” (Presumably because he liked the look of an oak tree that was right outside his window at Sun.) The people at Sun later realized that Oak was the name of an existing computer language, so they changed the name to Java.

In 1992, the Green project delivered its first product, called “*7.” It was an extremely intelligent remote control. (It had the power of a SPARCstation in a box that was 6 inches by 4 inches by 4 inches.) Unfortunately, no one was interested in producing this at Sun, and the Green people had to find other ways to market their technology. However, none of the standard consumer electronics companies were interested. The group then bid on a project to design a cable TV box that could deal with new cable services such as video on demand.

They did not get the contract. (Amusingly, the company that did was led by the same Jim Clark who started Netscape—a company that did much to make Java successful.)

The Green project (with a new name of “First Person, Inc.”) spent all of 1993 and half of 1994 looking for people to buy its technology—no one was found. (Patrick Naughton, one of the founders of the group and the person who ended up doing most of the

marketing, claims to have accumulated 300,000 air miles in trying to sell the technology.) First Person was dissolved in 1994.

While all of this was going on at Sun, the World Wide Web part of the Internet was growing bigger and bigger. The key to the Web is the browser that translates the hypertext page to the screen. In 1994, most people were using Mosaic, a noncommercial Web browser that came out of the supercomputing center at the University of Illinois in 1993. (Mosaic was partially written by Marc Andreessen for \$6.85 an hour as an undergraduate student on a work-study project. He moved on to fame and fortune as one of the cofounders and the chief of technology at Netscape.)

In the *SunWorld* interview, Gosling says that in mid-1994, the language developers realized that “We could build a real cool browser. It was one of the few things in the client/server mainstream that needed some of the weird things we'd done: architecture neutral, real-time, reliable, secure—issues that weren't terribly important in the workstation world. So we built a browser.”

The actual browser was built by Patrick Naughton and Jonathan Payne and evolved into the HotJava browser that we have today. The HotJava browser was written in Java to show off the power of Java. But the builders also had in mind the power of what are now called applets, so they made the browser capable of executing code inside web pages. This “proof of technology” was shown at SunWorld '95 on May 23, 1995, and inspired the Java craze that continues unabated today.

The big breakthrough for widespread Java use came in the fall of 1995, when Netscape decided to make the Navigator browser Java enabled in January 1996.

Other licensees include IBM, Symantec, Inprise, and many others. Even Microsoft has licensed Java. Internet Explorer is Java enabled, and Windows ships with a Java virtual machine. (Note that Microsoft does not support the most current version of Java, however, and that its implementation differs from the Java standard.)

Sun released the first version of Java in early 1996. It was followed by Java 1.02 a couple of months later. People quickly realized that Java 1.02 was not going to cut it for serious application development. Sure, you could use Java 1.02 to make a nervous text applet that moves text randomly around in a canvas. But you couldn't even *print* in Java 1.02. To be blunt, Java 1.02 was not ready for prime time.

The big announcements about Java's future features trickled out over the first few months of 1996. Only at the JavaOne conference held in San Francisco in May of 1996 did the bigger picture of where Java was going become clearer. At JavaOne the people at Sun Microsystems outlined their vision of the future of Java with a seemingly endless stream of improvements and new libraries.

The big news of the 1998 JavaOne conference was the upcoming release of Java 1.2, which replaces the early toy-like GUI and graphics toolkits with sophisticated and scalable versions that come a lot closer to the promise of “Write Once, Run Anywhere”™ than their predecessors. Three days after (!) its release in December 1998, the name was changed to Java 2.

Since then, the core Java platform has stabilized. The current release, with the catchy name *Java 2 Software Development Kit, Standard Edition version 1.3*, is an incremental improvement over the initial Java 2 release, with a small number of new features, increased performance and, of course, quite a few bug fixes. Now that a stable foundation exists, innovation has shifted to advanced Java libraries such as the Java 2 Enterprise Edition and the Java 2 Micro Edition.

History of Java

1. [Brief history of Java](#)
2. [Java Version History](#)

Java history is interesting to know. The history of java starts from Green Team. Java team members (also known as **Green Team**), initiated a revolutionary task to develop a language for digital devices such as set-top boxes, televisions etc.

For the green team members, it was an advance concept at that time. But, it was suited for internet programming. Later, Java technology as incorporated by Netscape.



James Gosling

Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describes the history of java.

- 1) **James Gosling, Mike Sheridan, and Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.
- 2) Originally designed for small, embedded systems in electronic appliances like set-top boxes.

3) Firstly, it was called "**Greentalk**" by James Gosling and file extension was .gt.

4) After that, it was called **Oak** and was developed as a part of the Green project.

Why Oak name for java language?

5) **Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like U.S.A., France, Germany, Romania etc.

6) In 1995, Oak was renamed as "**Java**" because it was already a trademark by Oak Technologies.

Why Java name for java language?

7) **Why they choosed java name for java language?** The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA" etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell and fun to say.

According to James Gosling "Java was one of the top choices along with **Silk**". Since java was so unique, most of the team members preferred java.

8) Java is an island of Indonesia where first coffee was produced (called java coffee).

9) Notice that Java is just a name not an acronym.

10) Originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.

11) In 1995, Time magazine called **Java one of the Ten Best Products of 1995**.

12) JDK 1.0 released in(January 23, 1996).

Java Version History



There are many java versions that has been released. Current stable release of Java is Java SE 8.






1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan, 1996)
3. JDK 1.1 (19th Feb, 1997)
4. J2SE 1.2 (8th Dec, 1998)
5. J2SE 1.3 (8th May, 2000)
6. J2SE 1.4 (6th Feb, 2002)
7. J2SE 5.0 (30th Sep, 2004)
8. Java SE 6 (11th Dec, 2006)
9. Java SE 7 (28th July, 2011)
10. Java SE 8 (18th March, 2014)

Java Code Names

1. I was crawling web and found some intersting things about java. Firstly when java was release its name is Oak. After some time the creator of java found the name **Java**.
2. I found some of the intersting things like the code names of the reeases of the java which I am going to describe below.

VERSION CODE NAME RELEASE DATE

Version	Description of Code Name	Code Name	Date of Release
JDK 1.1.4		<i>Sparkler</i>	Sept 12, 1997
JDK 1.1.5		<i>Pumpkin</i>	Dec 3, 1997
JDK 1.1.6	A female character in Bible	<i>Abigail</i>	April 24, 1998
JDK 1.1.7	Roman cognomen used by	<i>Brutus</i>	Sept 28,

JDK 1.1.8	several politicians Name of a person/Football club	<i>Chelsea</i>	1998 April 8, 1999
J2SE 1.2	Playground	<i>Playground</i>	Dec 4, 1998
J2SE 1.2.1		(none)	March 30, 1999
J2SE 1.2.2		<i>Cricket</i>	July 8, 1999
J2SE 1.3		<i>Kestrel</i>	May 8, 2000
J2SE 1.3.1		<i>Ladybird</i>	May 17, 2001
J2SE 1.4.0		<i>Merlin</i>	Feb 13, 2002
J2SE 1.4.1		<i>Hopper</i>	Sept 16, 2002
J2SE 1.4.2		<i>Mantis</i>	June 26, 2003
J2SE 5.0 (1.5.0)		<i>Tiger</i>	Sept 29, 2004
Java SE 6		<i>Mustang</i>	

Java SE 7



Dolphin

Features of Java

There is given many features of java. They are also known as java buzzwords. The Java Features given below are simple and easy to understand.

1. Simple
2. Object-Oriented
3. Platform independent
4. Secured
5. Robust
6. Architecture neutral
7. Portable
8. Dynamic
9. Interpreted
10. High Performance
11. Multithreaded
12. Distributed

Simple

According to Sun, Java language is simple because:

 syntax is based on C++ (so easier for programmers to learn it after C++).

 removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc.

 No need to remove un referenced objects because there is Automatic Garbage Collection in java.

Object-oriented

Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behaviour.

Object-oriented programming(OOPs) is a methodology that simplify software development and maintenance by providing some rules.

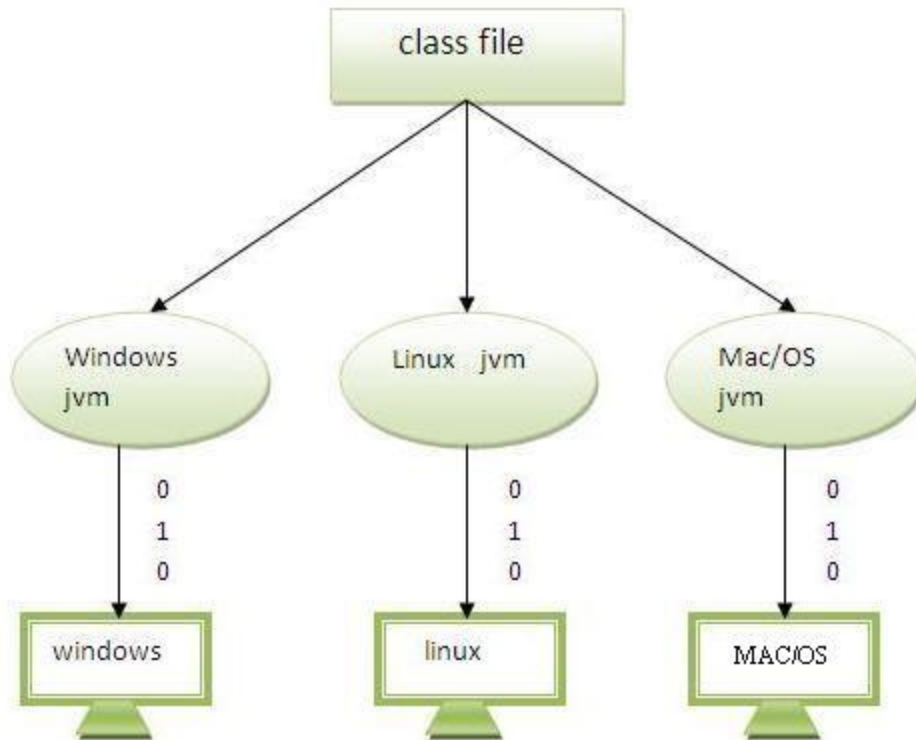
Basic concepts of OOPs are:

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

Platform Independent

A platform is the hardware or software environment in which a program runs. There are two types of platforms software-based and hardware-based. Java provides software-based platform. The Java platform differs from most other platforms in the sense that it's a software-based platform that runs on top of other hardware-based platforms. It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

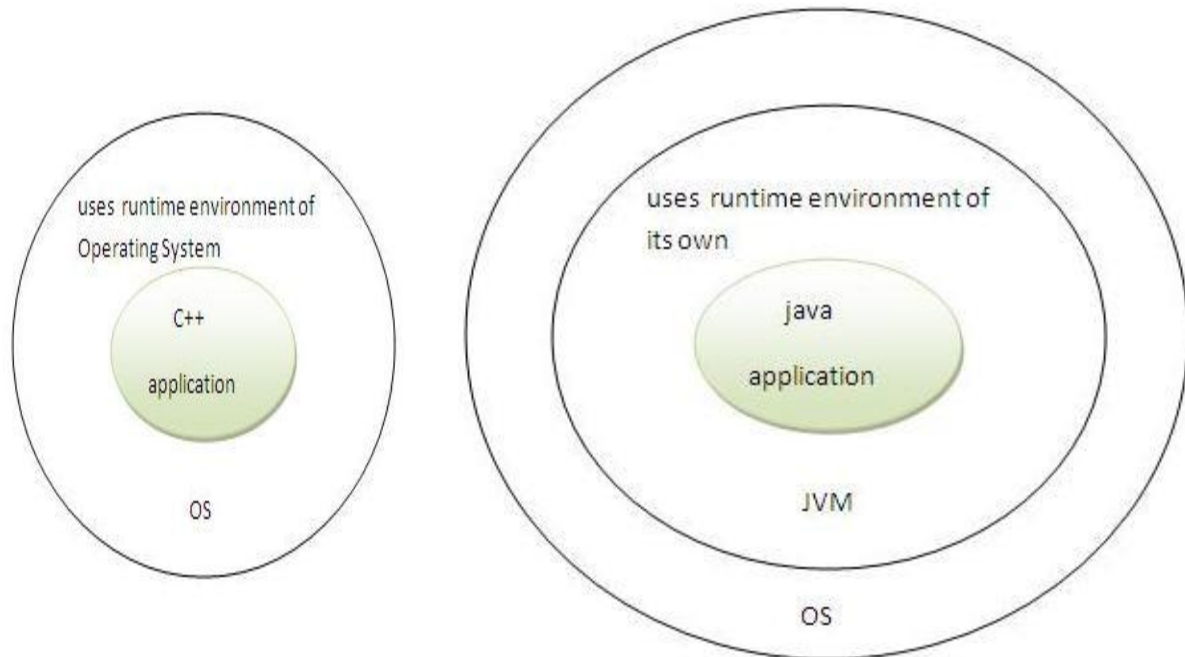


Java code can be run on multiple platforms e.g.Windows, Linux, Sun Solaris, Mac/OS etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere(WORA).

Secured

Java is secured because:

- No explicit pointer
- Programs run inside virtual machine sandbox.



- **ClassLoader-** adds security by separating the package for the classes of the local file system from those that are imported from network sources.
- **Bytecode Verifier-** checks the code fragments for illegal code that can violate access right to objects.
- **Security Manager-** determines what resources a class can access such as reading and writing to the local disk.

These security are provided by java language. Some security can also be provided by application developer through SSL,JAAS,cryptography etc.

Robust

Robust simply means strong. Java uses strong memory management. There are lack of pointers that avoids security problem. There is automatic garbage collection in java. There is exception handling and type checking mechanism in java. All these points makes java robust.

Architecture-neutral

There is no implementation dependent features e.g. size of primitive types is set.

Portable

We may carry the java bytecode to any platform.

High-performance

Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++)

Distributed

We can create distributed applications in java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet.

Multi-threaded

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it shares the same memory. Threads are important for multi-media, Web applications etc.

Simple Program of Java

1. [Software Requirements](#)
2. [Creating Hello Java Example](#)
3. [Resolving javac is not recognized problem](#)

In this page, we will learn how to write the simple program of java. We can write a simple hello java program easily after installing the JDK.

To create a simple java program, you need to create a class that contains main method. Let's understand the requirement first.

Requirement for Hello Java Example

For executing any java program, you need to

- install the JDK if you don't have installed it, [download the JDK](#) and install it.
- set path of the jdk/bin directory. <http://www.javatpoint.com/how-to-set-path-in-java>
- create the java program
- compile and run the java program

Creating hello java example

Let's create the hello java program:

```
1. class Simple{
2.     public static void main(String args[]){
3.         System.out.println("Hello Java");
4.     }
5. }
```

Test it Now

save this file as Simple.java

To compile: javac Simple.java

To execute: java Simple

Output: Hello Java

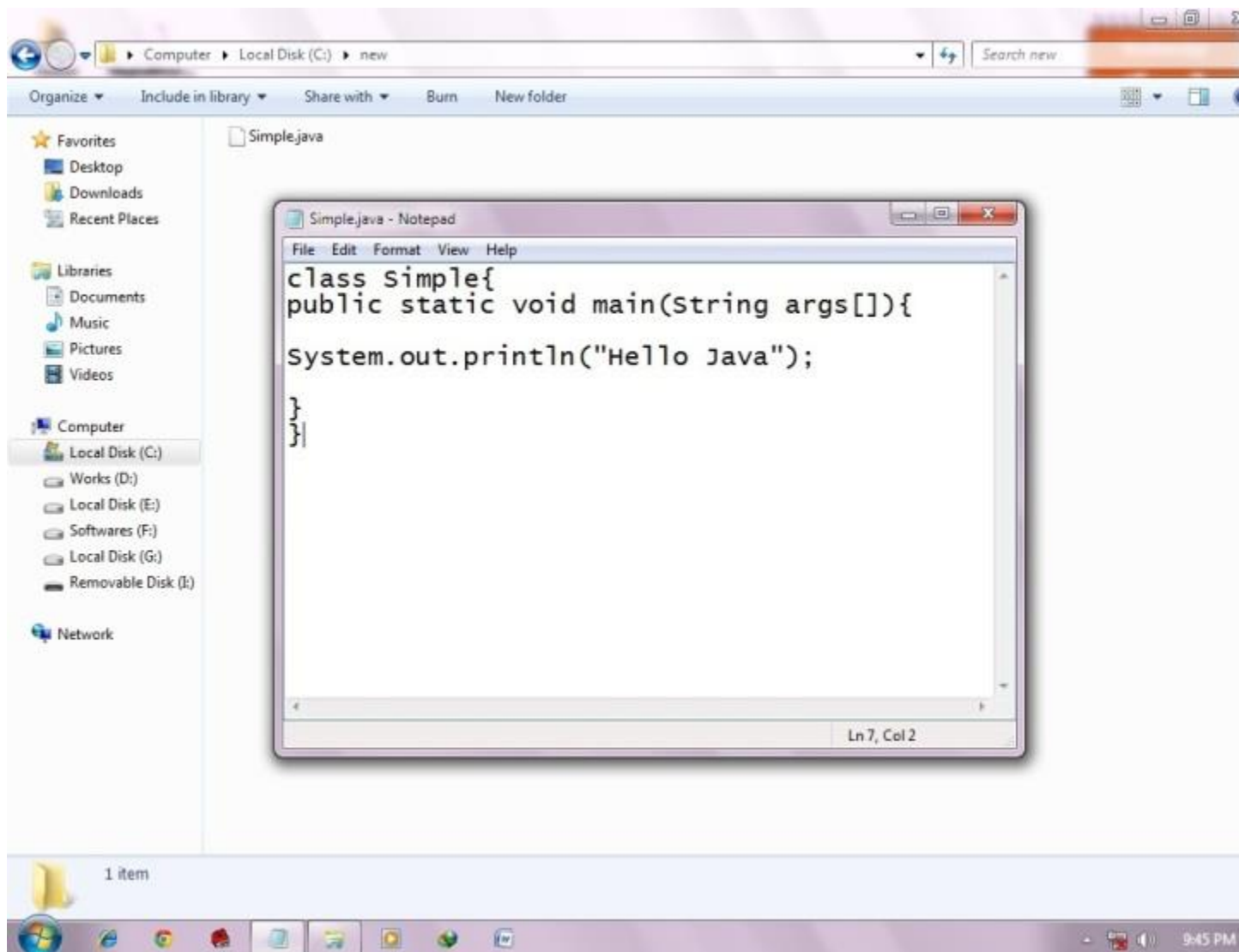
Understanding first java program

Let's see what is the meaning of class, public, static, void, main, String[], System.out.println().

- **class** keyword is used to declare a class in java.
- **public** keyword is an access modifier which represents visibility, it means it is visible to all.
- **static** is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create 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 startup of the program.
- **String[] args** is used for command line argument. We will learn it later.
- **System.out.println()** is used print statement. We will learn about the internal working of System.out.println statement later.

To write the simple program, open notepad by **start menu -> All Programs -> Accessories -> notepad** and write simple program as displayed below:



As displayed in the above diagram, write the simple program of java in notepad and

saved it as Simple.java. To compile and run this program, you need to open command prompt by **start menu -> All Programs -> Accessories -> command prompt**.

How to set path in Java

1. [How to set path of JDK in Windows OS](#)
1. [Setting Temporary Path of JDK](#)
2. [Setting Permanent Path of JDK](#)
2. [How to set path of JDK in Linux OS](#)

The path is required to be set for using tools such as javac, java etc.

If you are saving the java source file inside the jdk/bin directory, path is not required to be set because all the tools will be available in the current directory.

But If you are having your java file outside the jdk/bin folder, it is necessary to set path of JDK.

There are 2 ways to set java path:

1. temporary
2. permanent

1) How to set Temporary Path of JDK in Windows

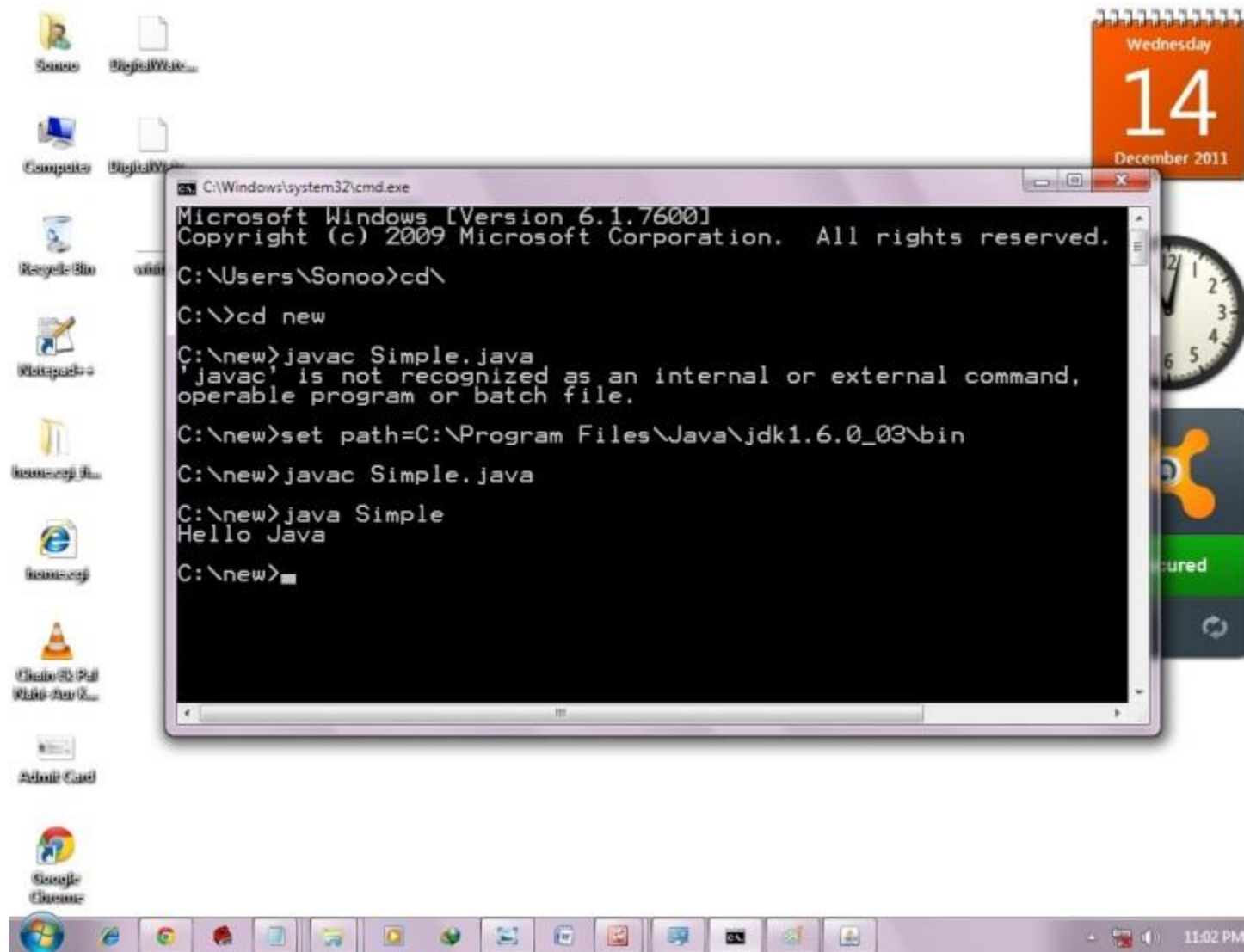
To set the temporary path of JDK, you need to follow following steps:

- Open command prompt
- copy the path of jdk/bin directory
- write in command prompt: set path=copied_path

For Example:

```
set path=C:\Program Files\Java\jdk1.6.0_23\bin
```

Let's see it in the figure given below:



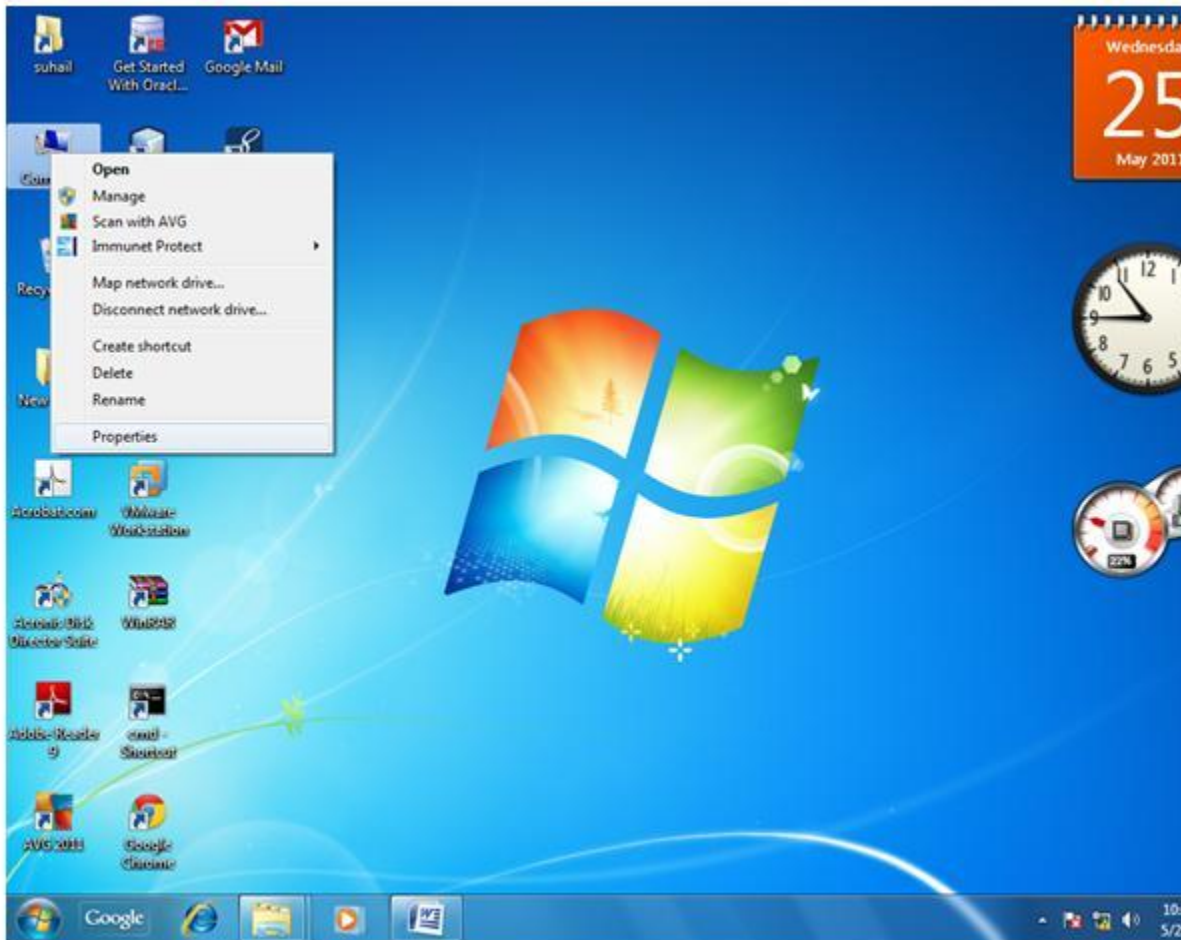
2) How to set Permanent Path of JDK in Windows

For setting the permanent path of JDK, you need to follow these steps:

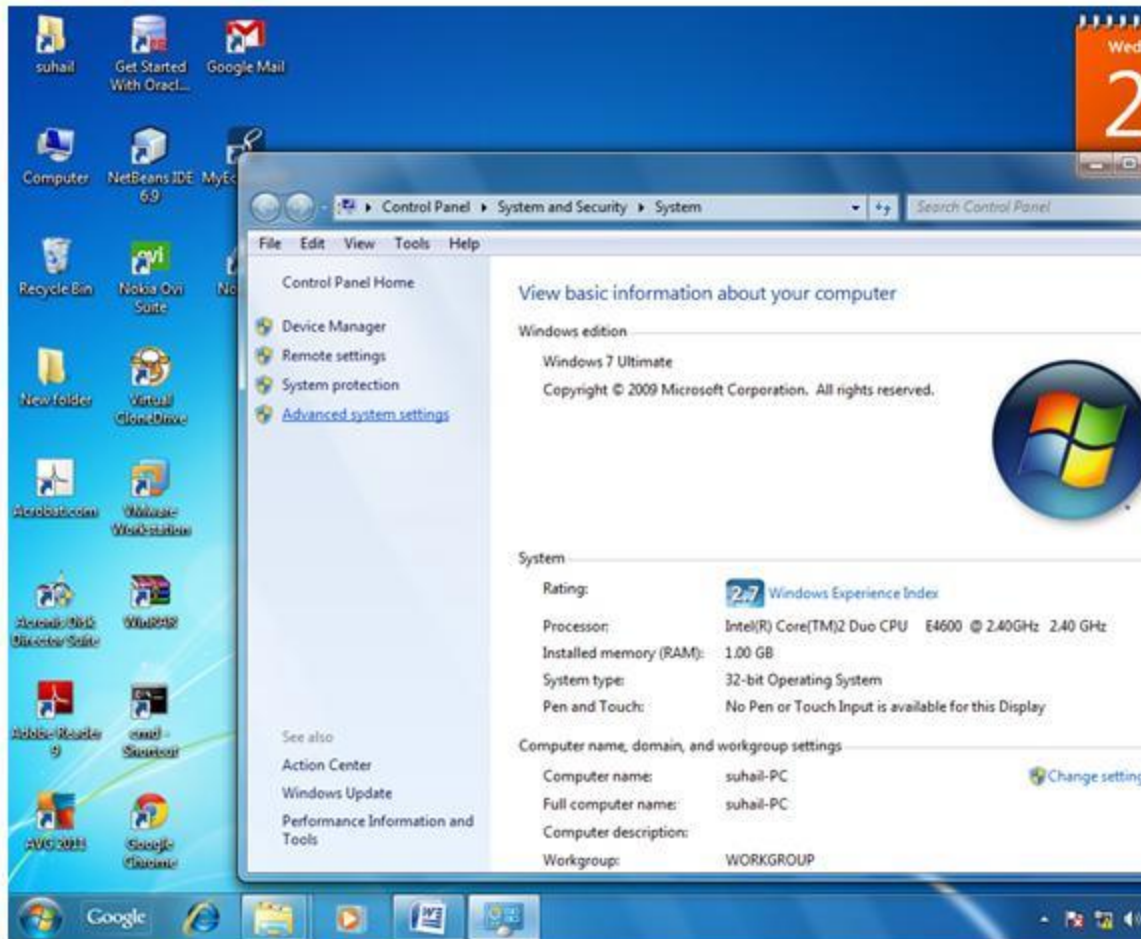
- Go to MyComputer properties -> advanced tab -> environment variables -> new tab of user variable -> write path in variable name -> write path of bin folder in variable value -> ok -> ok -> ok

For Example:

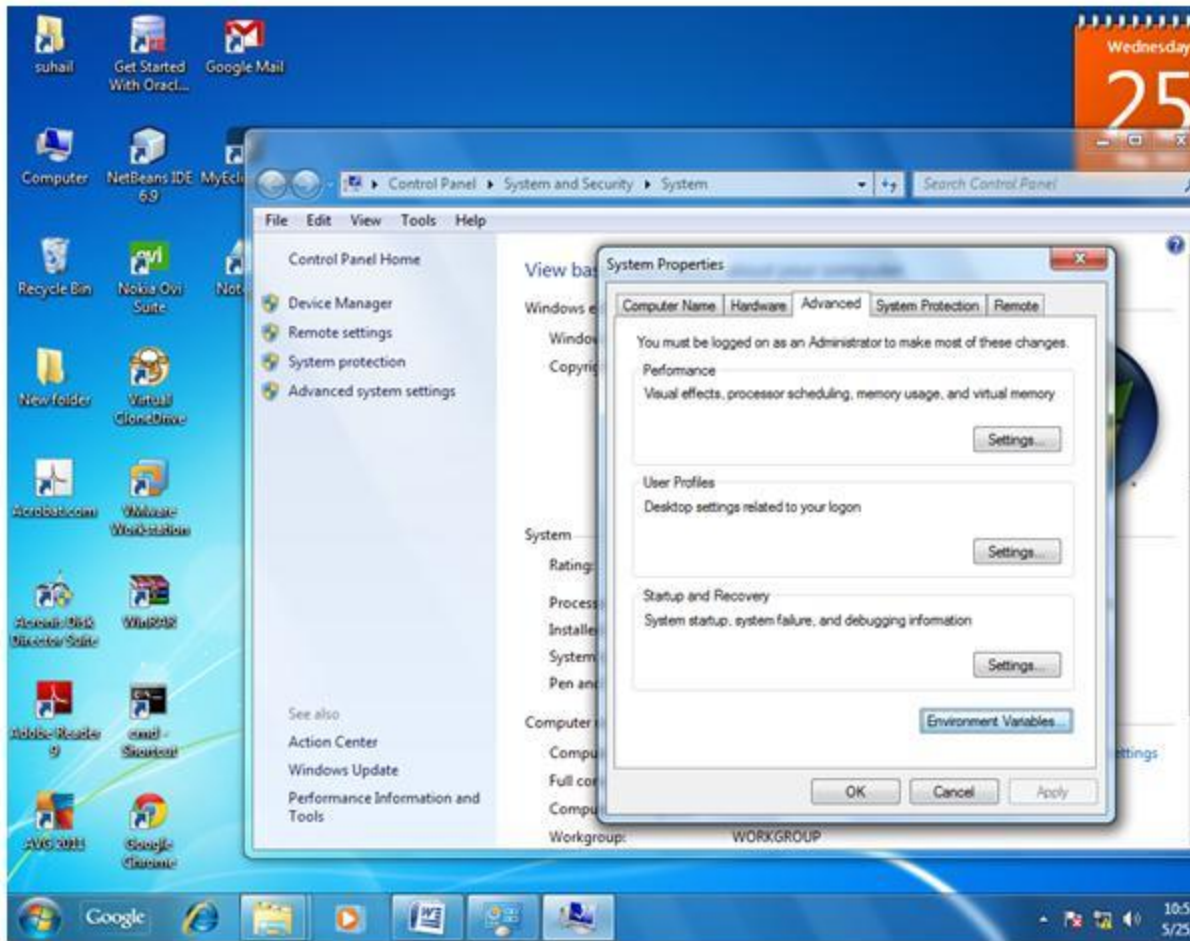
1)Go to MyComputer properties



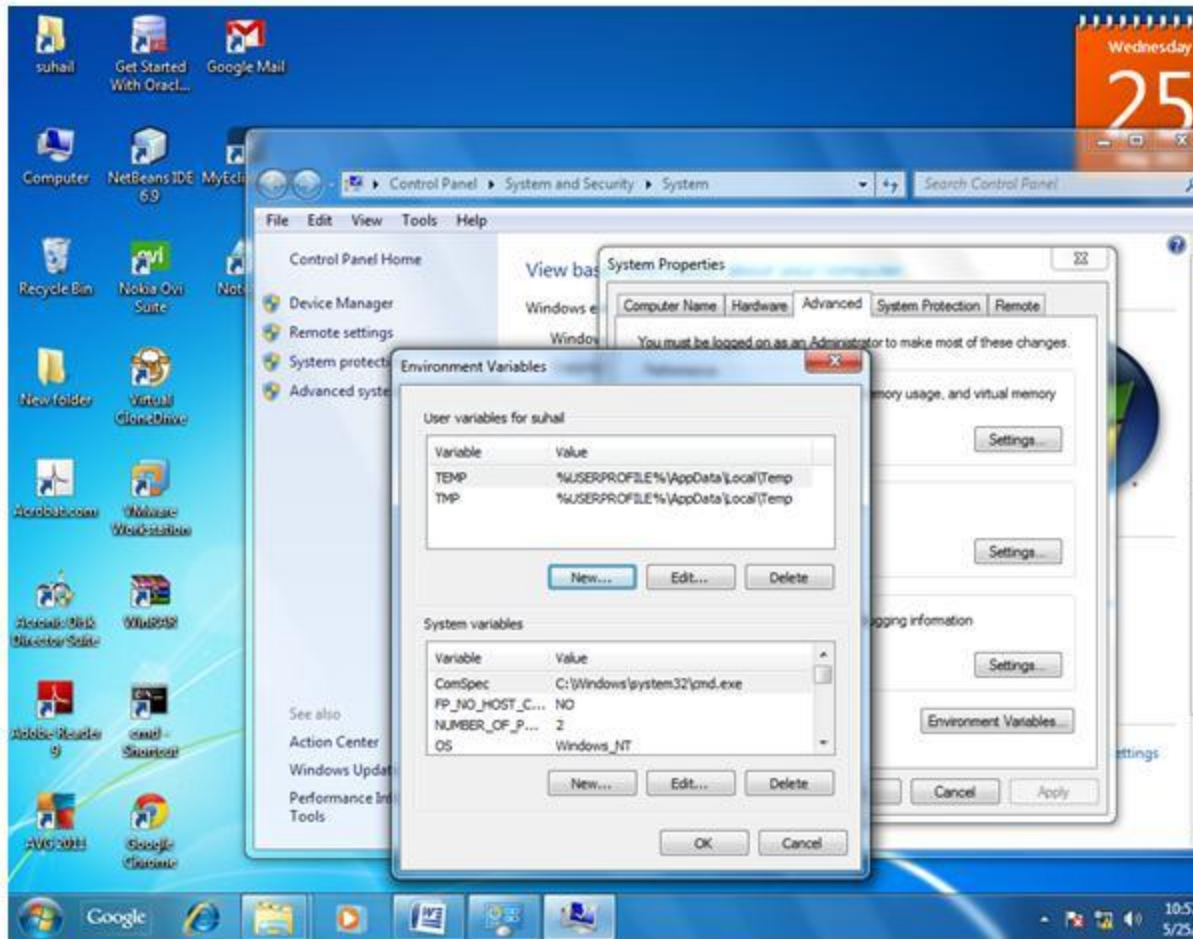
2)click on advanced tab



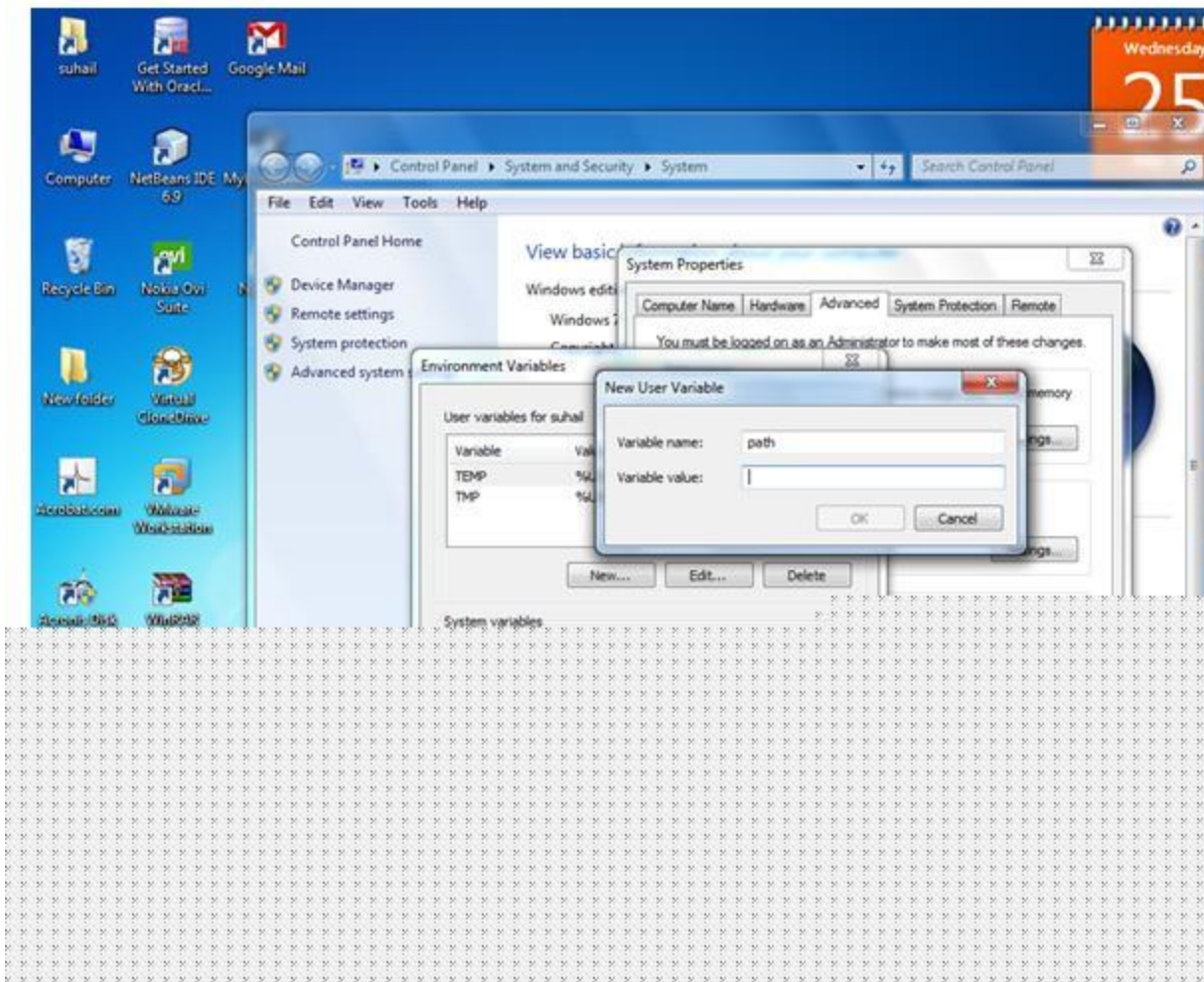
3)click on environment variables



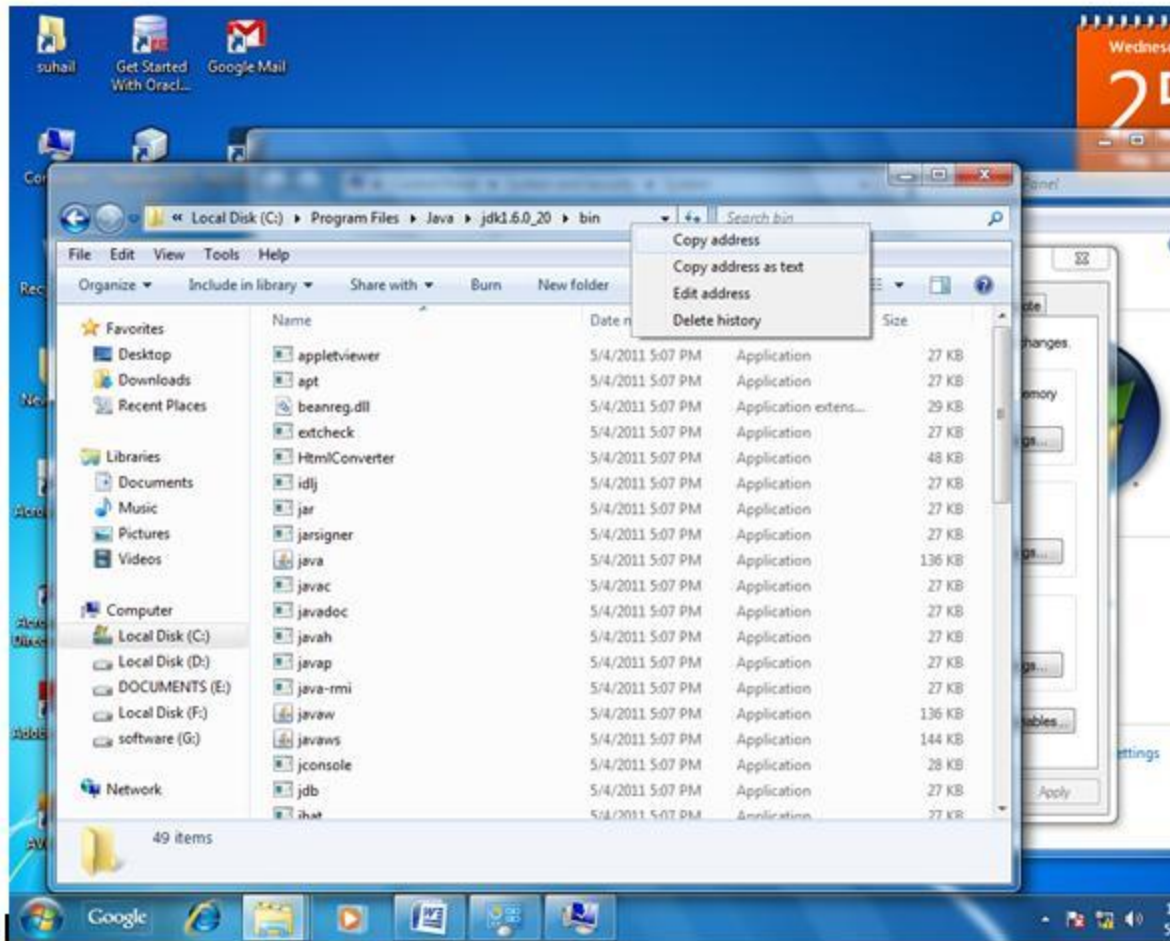
4)click on new tab of user variables



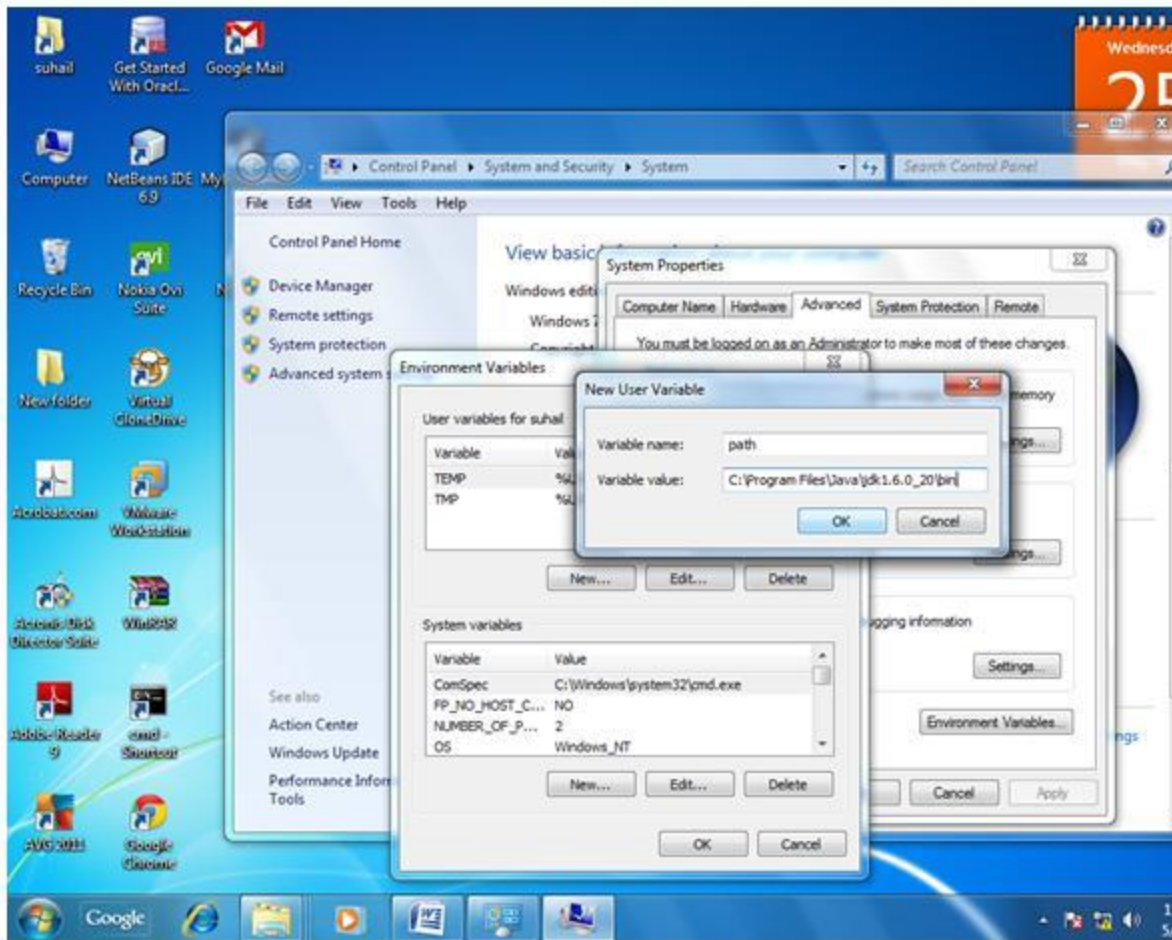
5) write path in variable name



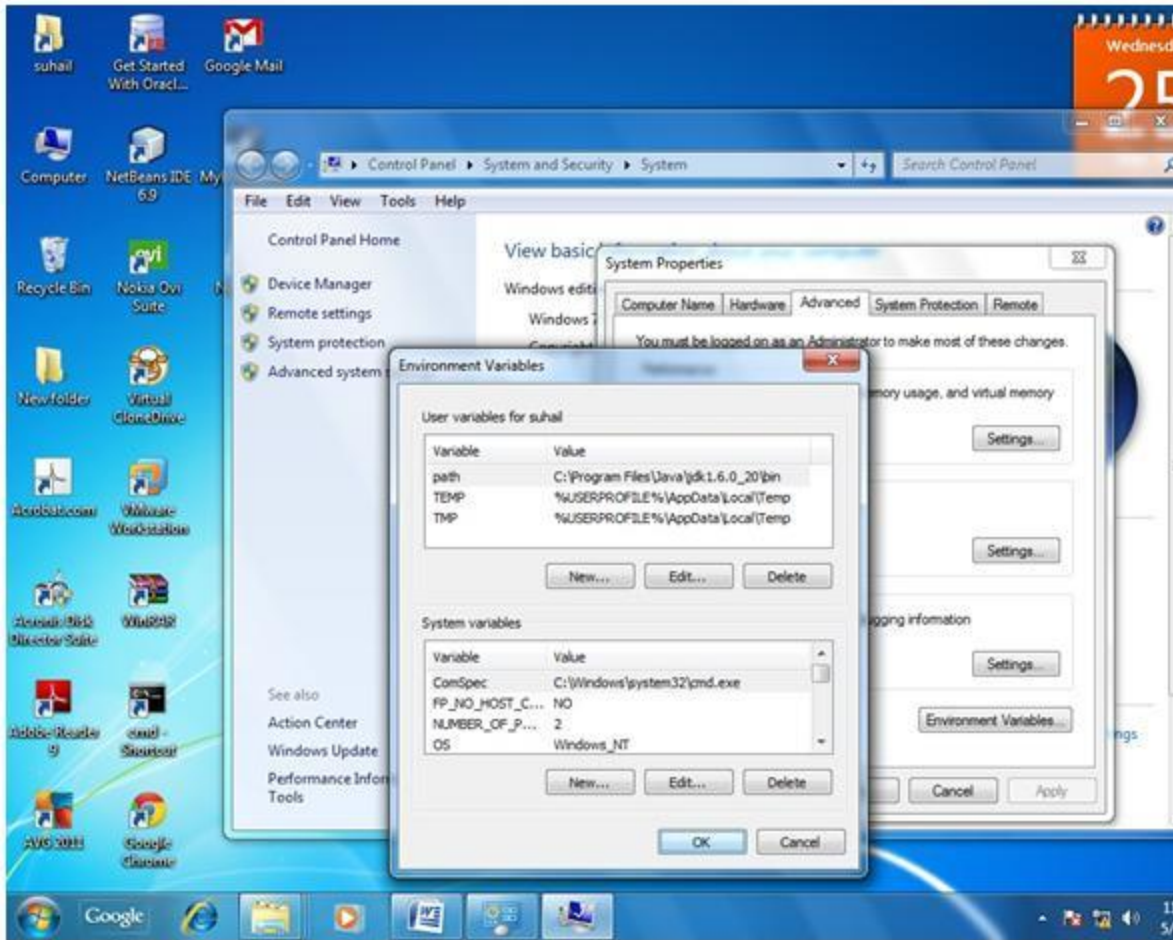
6) Copy the path of bin folder



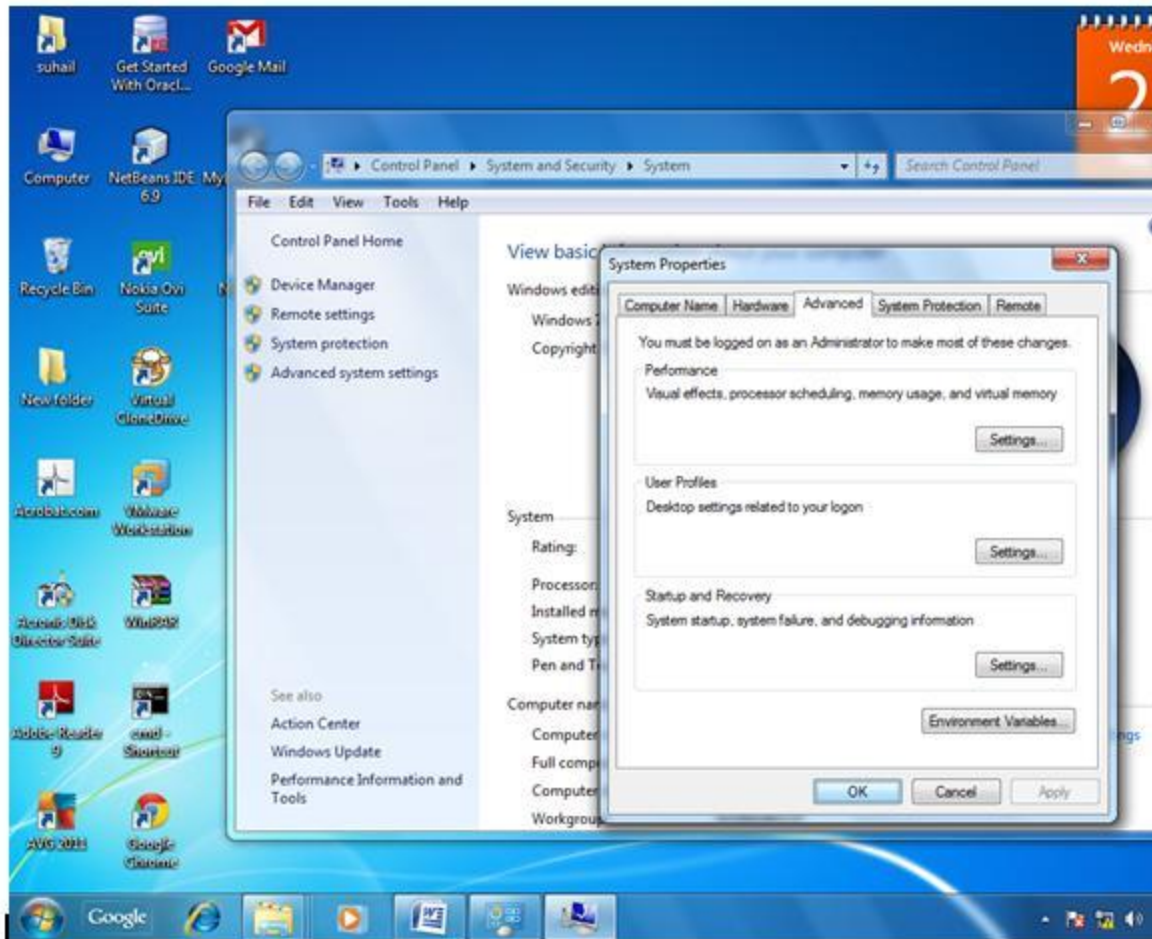
7) paste path of bin folder in variable value



8)click on ok button



9)click on ok button



Now your permanent path is set. You can now execute any program of java from any drive.

Setting Java Path in Linux OS

Setting the path in Linux OS is same as setting the path in the Windows OS. But here we use export tool rather than set. Let's see how to set path in Linux OS:

```
export PATH=$PATH:/home/jdk1.6.01/bin/
```

Here, we have installed the JDK in the home directory under Root (/home).

Difference between JDK, JRE and JVM

1. Brief summary of JVM
2. Java Runtime Environment (JRE)
3. Java Development Kit (JDK)

Understanding the difference between JDK, JRE and JVM is important in Java. We are having brief overview of JVM here.

If you want to get the detailed knowledge of Java Virtual Machine, move to the next page. Firstly, let's see the basic differences between the JDK, JRE and JVM.

JVM

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides 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 configuration of each OS differs. But, Java is platform independent.

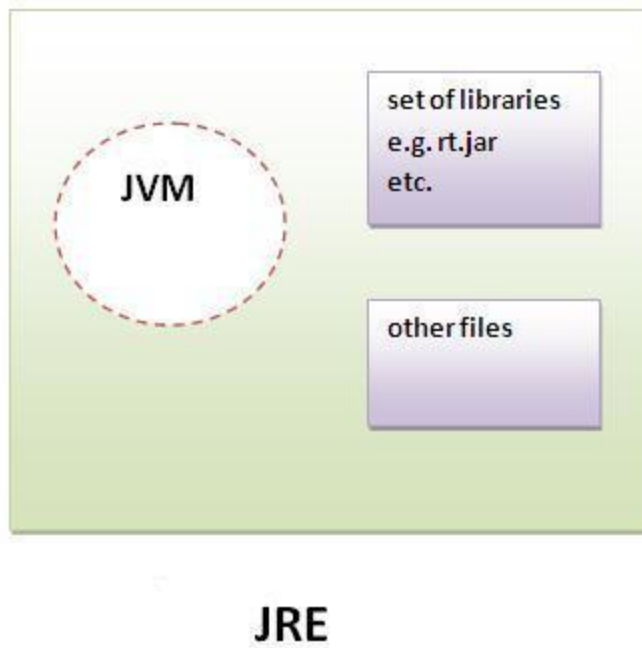
The JVM performs following main tasks:

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

JRE

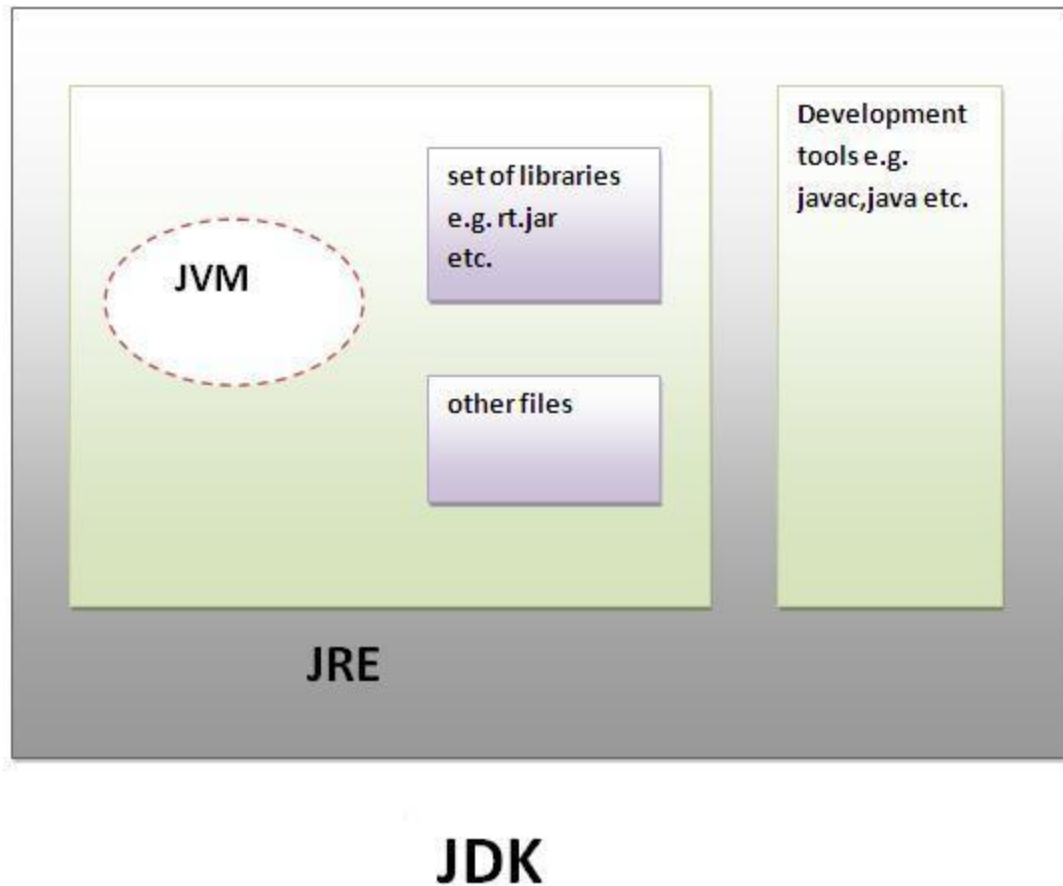
JRE is an acronym for Java Runtime Environment. It is used to provide runtime environment. It is the implementation of JVM. It physically exists. It contains set of libraries + other files that JVM uses at runtime.

Implementation of JVMs are also actively released by other companies besides Sun Micro Systems.



JDK

JDK is an acronym for Java Development Kit. It physically exists. It contains JRE + development tools.



JVM (Java Virtual Machine)

1. Java Virtual Machine
2. Internal Architecture of JVM

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

What is JVM?

It is:

1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.

2. **An implementation** Its implementation is known as JRE (Java Runtime Environment).
3. **Runtime Instance** Whenever you write java command on the command prompt to run the java class, and instance of JVM is created.

What it does?

The JVM performs following operation:

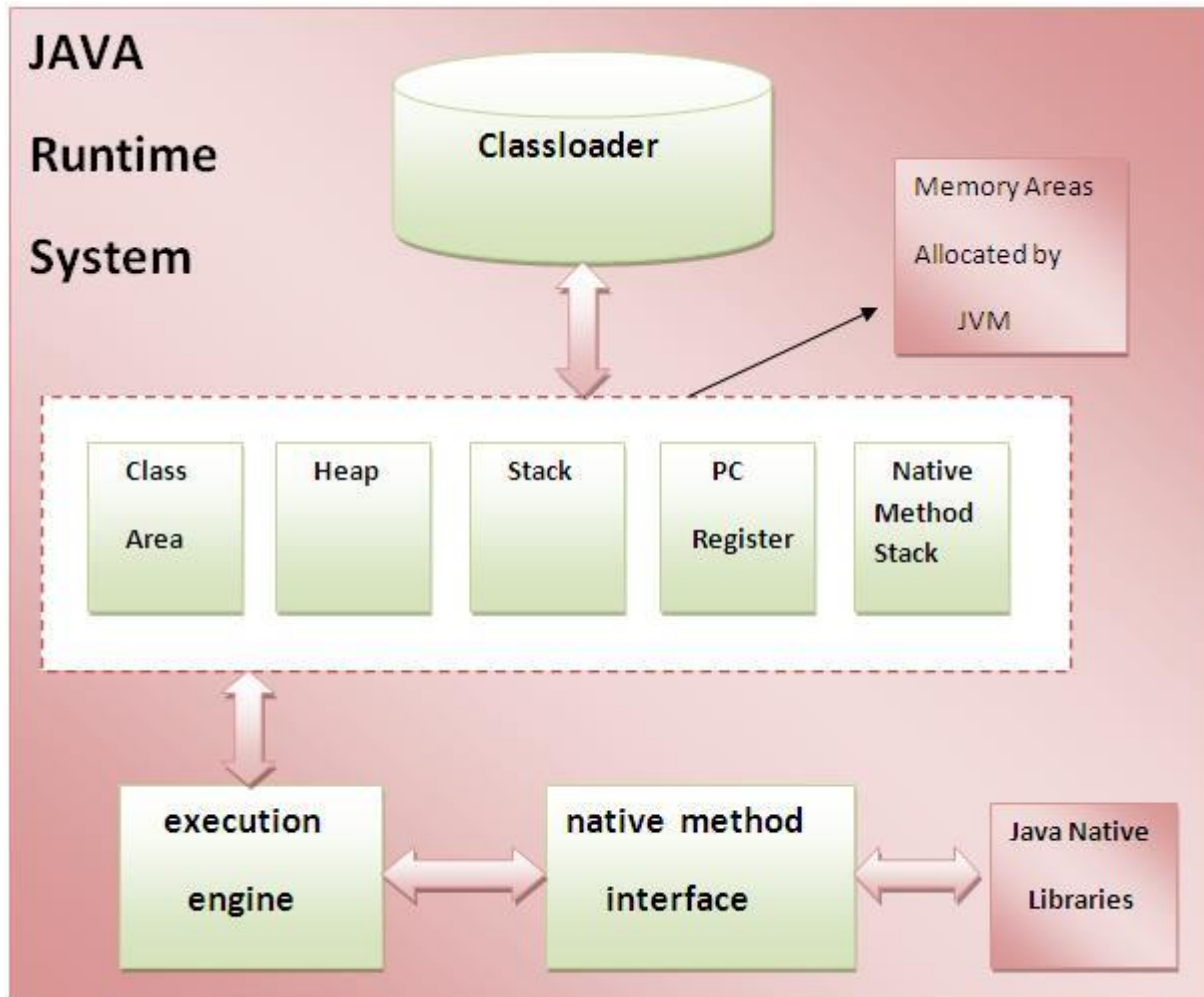
- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap
- Fatal error reporting etc.

Internal Architecture of JVM

Let's understand the internal architecture of JVM. It contains classloader, memory area, execution engine etc.



1) Classloader:

Classloader is a subsystem of JVM that is used to load class files.

2) Class(Method) Area:

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap:

It is the runtime data area in which objects are allocated.

4) Stack:

Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register:

PC (program counter) register. It contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack:

It contains all the native methods used in the application.

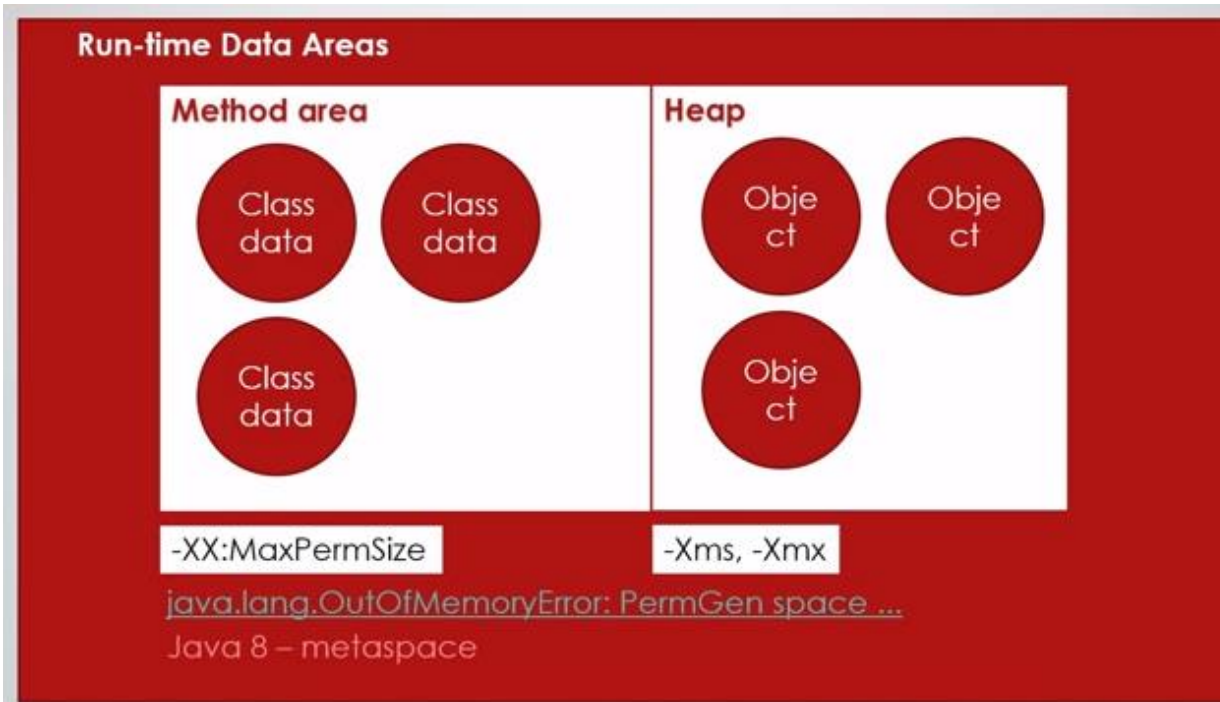
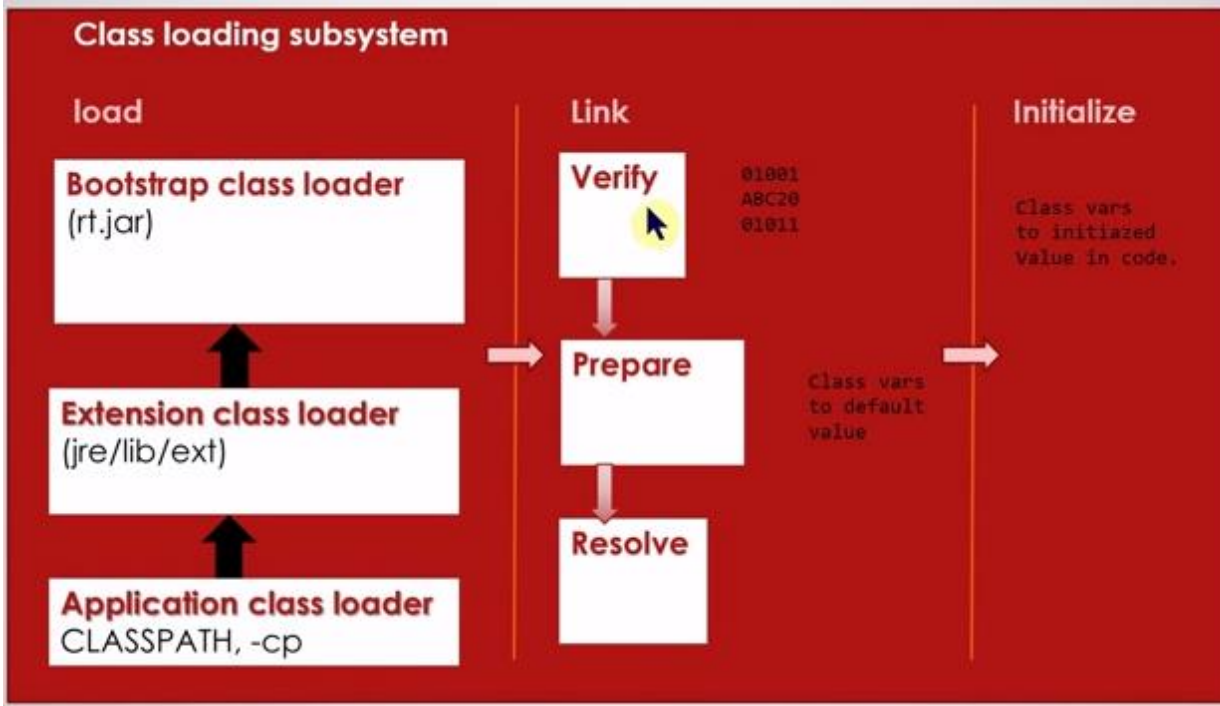
7) Execution Engine:

It contains:

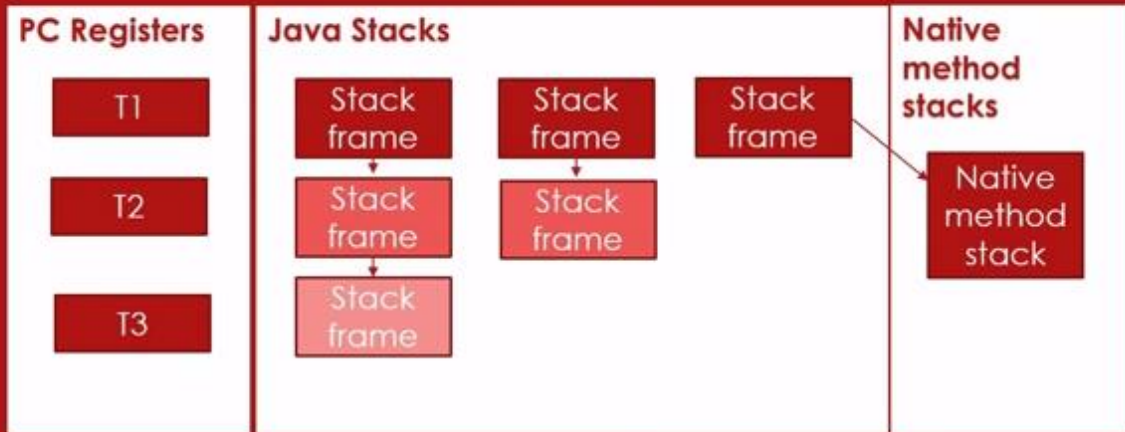
1) A virtual processor

2) Interpreter: Read bytecode stream then execute the instructions.

3) Just-In-Time (JIT) compiler: It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here the term ?compiler? refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.



Run-time Data Areas (Contd.)



-Xss

`java.lang.StackOverflowError`

