Unit-1

1. Introduction to Object Oriented Programming

The Object Oriented Programming (OOP) is one of the most interesting innovations in the Software Development. It addresses problems commonly known as "**Software Crisis**". Many software failed in the past. The term 'Software Crisis' is described in terms of failure in the software

- Exceeding Software Budget
- Software is not meeting the client's requirements.
- Bugs in the software.

OOP is a programming paradigm with deals with the concepts of the objects to build programs and software applications. IT is molded around the real world objects. Every object has well-defined ***identity, attributes, and behavior***. The features of the object oriented programming are similar to the real world features like ***Inheritance, abstraction, encapsulation, and polymorphism***.

2. Need of OOP

Whenever a new programming language is designed some trade-offs are made, such as

- ease-of-use verses power
- safety verses efficiency
- Rigidity versus extensibility

Prior to the c language, there were many languages:

- FORTRAN, which is efficient for scientific applications, but is not good for ***system code.***
- BASIC, which is easy to learn, but is not powerful, it is lack of ***structure.***
- Assembly Language, which is highly efficient, but is not ***easy to learn***.

These languages are designed to work with GOTO statement. As a result programs written using these languages tend to produce "**spaghetti code**", which is impossible to understand.

Dennis Ritchie, during his implementation of UNIX operating system, he developed C language, which similar to an older language called **BCPL**, which is developed by Martin Richards. The BCPL was influenced by a language called **B**, invented by Ken Thomson. C was formally standardized in December 1989, when the American National Standards Institute (ANSI) standard for C was adopted.

During the late 1970s and early 1980s, C became the dominant computer programming language, and it is still widely used today. The C language is ***structured***, ***efficient*** and ***high level language***. Since C is a successful and useful language, you might ask why ***a need for something else existed***. The answer is ***complexity***. As the program complexity is increasing it demanded the better way to manage the complexity.

When computers were first programming was done by manually **toggling** in the binary machine instructions by use of the front panel. As long as programs were just a few hundred instructions long, this approach worked. As programs grew, **assembly language** was invented so that a programmer could deal with larger, increasingly complex programs by using symbolic representations of the machine instructions. As programs continued to grow, **high-level languages** were introduced that gave the programmer more tools with which to handle complexity.

The 1960s gave birth to ***structured programming***. This is the method of programming championed by languages such as C. The use of structured languages enabled programmers to write, for the first time, moderately complex programs fairly easily. However, even with structured programming methods, once a project reaches a certain size, its complexity exceeds what a programmer can manage. To solve this problem, a new way to program was invented, called ***object-oriented programming (OOP).*** OOP is a programming methodology that helps organize complex programs through the use of **inheritance**, **encapsulation**, and **polymorphism**.

3. The OOP Principles

OOP languages follow certain principles such as, class, object, abstraction, encapsulation, inheritance and polymorphism.

3.1 Classes

A class is defined as the blueprint for an object. It serves as a plan or template. An object is not created just by defining the class, but it has to be created explicitly. It is also defined as new data type **contains data and methods.**

3.2 Objects

Objects are defined as the instances of a class. For example chairs and tables are all instances of Furniture. The objects have unique Identity, State and Behavior.

3.3 Abstraction

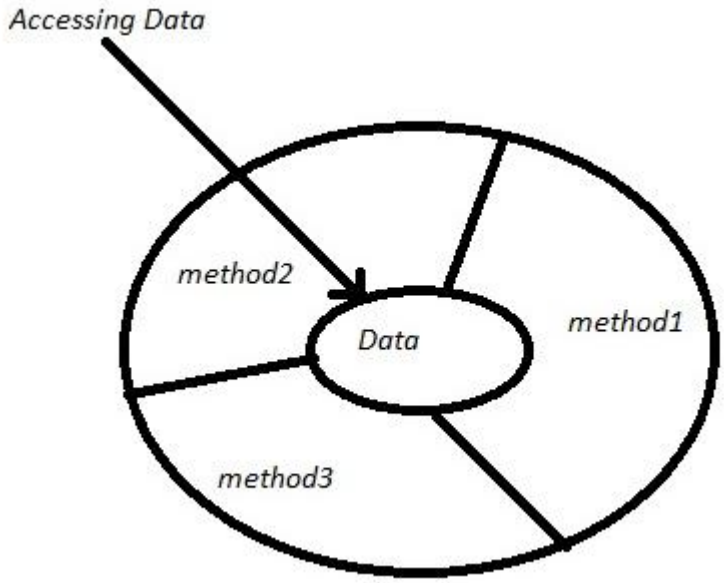
We can group the following items as Animals, Furniture and Electronic Devices.

Elephant	Television
CD Player	Chair
Table	Tiger

Here just by focusing on the generic characteristics of the items we can group the items into their classes rather than specific characteristics. This is called *"**abstraction***".

3.4 Encapsulation

- ***Encapsulation*** is the mechanism that binds ***methods*** and ***data*** together into a single unit.
- This hides the data from the outside world and keeps both safe from outside interference and misuse.
- It puts some restriction on outside code from directly accessing data.
- Encapsulation is also known as "***Data Hiding***".
- The Data can be accessed by the methods of the same class. Access to the code and data inside the wrapper is tightly controlled through a well-defined interface.
- In Java, the basis of encapsulation is the **class**.
- A *class* defines the structure and behavior (Data and Method) that will be shared by a set of **objects**. Each object of a given class contains the structure and behavior defined by the class.
- The objects are sometimes referred to as *instances of a class.* Thus, a class is a **logical** construct; an object has **physical** reality.
- When you create a class, you will specify the code and data that constitute that class. Collectively, these elements are called ***members*** of the class.
- Specifically, the data defined by the class are referred to as ***member variables*** **or** ***instance variables***.
- The code that operates on that ata is referred to as ***member methods*** **or just** ***methods***. The members can be **public or private**. When a member is made public any code outside the class can access them. If the members are declared as private, then only the members of that class can access its members.



3.5 Inheritance

- *Inheritance* is the process by which one class acquires the properties or characteristics from another class.
- Here we have two types of classes: ***base*** class and ***derived*** class.
- The class from which the properties or characteristics are acquired is called "***Base Class***". The class that acquires the properties is called "***Derived Class***".
- The base class is also called super class or parent class. The derived class is also called sub class or child class.
- The main use of Inheritance is code reusability.
- The keyword "**extends**" is used for inheriting the properties.

3.6 Polymorphism

- *Polymorphism simply means many forms* (from Greek, meaning "many forms").
- It can be defined as same thing being used in different forms.
- It has two forms: ***compile-time*** polymorphism and ***run-time*** polymorphism.
- We know that binding is the process of linking function call to the function definition. If the linking is done at compile-time, then it is called compile-time binding. If it is done at the run time it is called run-time binding.
- The compile-time binding is also called as "static binding". The run-time binding is also called as "dynamic binding".
- The compile-time binding is implemented using method ***overloading***. The run-time binding is implemented using method ***overriding***.

4. Procedural language Vs OOP

All computer programs consist of two elements: methods and data. Furthermore, a program can be conceptually organized around its methods or around its data. That is, some programs are written around "***what is happening***" and others are written around "***who is being affected.***" These are the two paradigms that govern how a program is constructed. The first way is called the ***process-oriented model or procedural language***. This approach characterizes a program as a series of linear steps (that is, code). The process-oriented model can be thought of as ***code acting on data.*** Procedural languages such as C employ this model to considerable success.

To manage increasing complexity, the second approach, called ***object-oriented programming,*** was designed. Object-oriented programming organizes a program around its data (that is, objects) and a set of well-defined interfaces to that data. An object-oriented program can be characterized as ***data controlling access to code***.

Procedural language	Object Oriented Programming language
Separates data from functions that operate on them	Encapsulates the data and methods in a class
Not suitable for defining abstract types	Not suitable for defining abstract types
Debugging is difficult	Debugging is easier
Difficult to implement the change	Easier manage and implement the change

Not suitable for large programs and applications	suitable for large programs and applications
Analysis and design is not so easy	Analysis and design is easy
Faster	Slower
Less flexible	Highly flexible
Less reusable	More reusable
Only procedures and data are there	Inheritance, Encapsulation and polymorphism are the key features.
Uses top-Down approach	Uses Bottom-Up approach
Examples: C, Basic, FORTRAN	Example: C++,Java

5. Applications of OOP

There are mainly 4 types of applications that can be created using java programming:

1) Standalone Application

It is also known as desktop application or window-based application. An application that we need to install on every machine such as media player, antivirus etc. AWT and Swing are used in java for creating standalone applications.

2) Web Application

An application that runs on the server side and creates dynamic page, is called web application. Currently, servlet, jsp, struts, etc. technologies are used for creating web applications in java.

3) Enterprise Application

An application that is distributed in nature, such as banking applications etc. It has the advantage of high level security, load balancing and clustering. In java, EJB is used for creating enterprise applications.

4) Mobile Application

An application created for mobile devices is called Mobile Applications. Currently Android and Java ME are used for creating mobile applications.

In addition to the above applications, there are other applications that use OOP concepts as follow:

5. Neural Networks –The neural system of the human body is simulated into the machine, so that optimizations are carried out according to it.

6. Real Times –These Systems work bases on the time. Examples airline, rockets, etc;

7. **Expert System** –Expert's knowledge is integrated with system, that system will respond as the expert in the particular domain.

8. Database management systems – Data is constructed into tables and collection of table is called database. Java is used to process the data that is available in the database, such as insertion, deletion, display etc;.

6. 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. Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describe 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. This name was inspired by the Oak Tree that stood outside Golsling's Office.

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. During 1991 to 1995 many people around the world contributed to the growth of the Oak, by adding the new features. **Bill Joy, Arthur Van Hoff**, **Jonathan Payne, Frank Yellin, and Tim Lindholm** were key contributors to the original prototype.

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 chosen 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). Java coffee was consumed in large quantities by the GreenTeam.

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**.

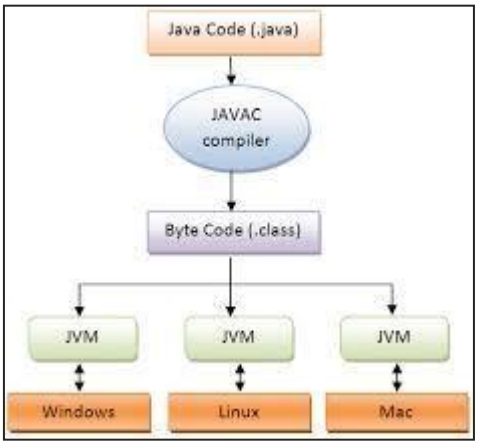
12) JDK 1.0 released in(January 23, 1996).

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 1. 5 (30th Sep, 2004)
8. Java SE 1.6 (11th Dec, 2006)
9. Java SE 1.7 (28th July, 2011)
10. Java SE 1.8 (18th March, 2014)

7. Java Virtual Machine



The key that allows Java to solve both the **security and the portability** problems is the ***byte code***. The output of Java Compiler is not directly executable. Rather, it contains highly optimized set of instructions. This set of instructions is called, "**byte code**". This byte code is designed to be executed by Java Virtual Machine (JVM). The JVM also called as the ***Interpreter*** for byte code. JVM also helps to solve many problems associated with web-based programs.

Translating a Java program into byte code makes it much easier to run a program in a wide variety of environments (or platforms) because only the JVM needs to be implemented for each platform. Once the run-time package exists for a given system, any Java program can run on it. Remember, although the details of the JVM will differ from platform to platform, all understand the same Java ***Byte Code***. Thus, the execution of byte code by the JVM is the easiest way to create truly portable programs.

The fact that a Java program is executed by the JVM also helps to make it ***secure***. Because the JVM is in control, it can contain the program and prevent it from generating side effects outside of the system.

In general, when a program is compiled to an intermediate form and then interpreted by a virtual machine, it ***runs slower than*** it would run if compiled to executable code. However, with Java, the differential between the two is not so great. Because byte code has been highly optimized, the use of byte code enables the JVM to execute programs much faster than you might expect

To give on-the-fly performance, the Sun began to design HotSpot Technology for Compiler, which is called, Just-In-Time compiler. The JIT, Compiler also produces output immediately after compilation.

8. 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.

- Simple
- Secure
- Portable
- Object-oriented
- Robust
- Multithreaded
- Architecture-neutral
- Interpreted
- High performance
- Distributed
- Dynamic

Simple

Java was designed to be easy for the professional programmer to learn and use effectively. 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 unreferenced objects because there is Automatic Garbage Collection in java.

Secure

Once the byte code generated, the code can be transmitted to other computer in the world without knowing the internal details of the source code.

Portable

The byte code can be easily carried from one machine to other machine.

Object Oriented

Everything in Java is an Object. The object model in Java is simple and easy to extend, while primitive types, such as integers, are kept as high-performance non-objects.

Robust

The multi-plat-formed environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. Thus, the ability to create robust programs was given a high priority in the design of Java. Java also frees from having worry about many errors. Java is Robust in terms of **memory management and mishandled exceptions**. Java provides automatic memory management and also provides well defined exception handling mechanism.

Multithreaded

Java was designed to meet the real-world requirement of creating interactive, networked programs. To accomplish this, Java supports multithreaded programming, which allows you to write programs that do many things **simultaneously**.

Architecture-neutral

The Java designers made several hard decisions in the Java language and the Java Virtual Machine in an attempt to alter this situation. Their goal was "**write once; run anywhere, any time, forever.**" To a great extent, this goal was accomplished.

Interpreted and High Performance

Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java byte code. This code can be executed on any system that implements the Java Virtual Machine. Most previous attempts at cross-platform solutions have done so at the expense of performance. As explained earlier, the Java byte code was carefully designed so that it would be easy to translate directly into native machine code for very high performance by using a just-in-time compiler.

Distributed

Java is designed for the distributed environment of the Internet because it handles TCP/IP protocols. In fact, accessing a resource using a URL is not much different from accessing a file. Java also supports ***Remote Method Invocation (RMI)***. This feature enables a program to invoke methods across a network.

Dynamic
Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time. This makes it possible to dynamically link code in a safe and expedient manner.

9. Program Structures

Simple Java Program

Example.java

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

Entering the Program

We can use any text editor such as "**notepad**' or "**dos text editor**". The source code is typed and is saved with **".java**" as extension. The source code contains one or more class definitions. The program name will be same as class name in which main function is written. This in not compulsory, but by convention this is used. ***The source file is officially called as compilation unit***. We can even uses our choice of interest name for the program. If we use a different name than the class name, then compilation is done with program name, and running is done with class file name. To avoid this confusion and organize the programs well, it is suggested to put the same name for the program and class name, but not compulsory.

Compiling the Program

To compile the program, first execute the compiler, "**javac**", specifying the name of the source file on the command line, as shown bellow:

c:\>javac Example.java

The javac compiler creates the file called "**Example.class**", that contains the byte code version of the source code. This byte code is the intermediate representation of the source code that contains the instructions that the Java Virtual Machine (JVM) will execute. Thus the output of the javac is not the directly executable code.

To actually run the program, we must use Java interpreter, called "java". This is interpreter the "**Example.class**" file given as input.
When the program is run with java interpreter, the following output is produced:

```
Hello World
```

Description of the every line of the program

The first line contains the keyword **class** and **class name**, which actually the basic unit for encapsulation, in which data and methods are declared.

Second line contains "{" which indicates the beginning of the class.

Third line contains the
public static void main(String args[])

where public is access specifier, when a member of a class is made public it can be accessed by the outside code also. The main function is the beginning of from where execution starts. Java is case-sensitive. "**Main**" is different from the "**main**". In main there is one parameter, String args, which is used to read the command line arguments.

Fourth line contains the "{", which is the begining of the main function.

Fifth line contains the statement

System.out.println("Hello World");

Here "**System**" is the predefined class, that provides access to the system, and **out** is the output stream that is used to connect to the console. The **println()**, is used to display string passed to it. This can even display other information to.

10.Installation of JDK 1.6

Installing the JDK Software

If you do not already have the JDK software installed or if `JAVA_HOME` is not set, the Java CAPS installation will not be successful. The following tasks provide the information you need to install JDK software and set `JAVA_HOME` on UNIX or Windows systems.

The following list provides the Java CAPS JDK requirements by platform.

Solaris

JDK5: At least release 1.5.0_14

JDK6: At least release 1.6.0_03

IBM AIX

JDK5: The latest 1.5 release supported by IBM AIX

Linux (Red Hat and SUSE)

JDK5: At least release 1.5.0_14

JDK6: At least release 1.6.0_03

Macintosh

JDK5: The latest 1.5 release supported by Apple

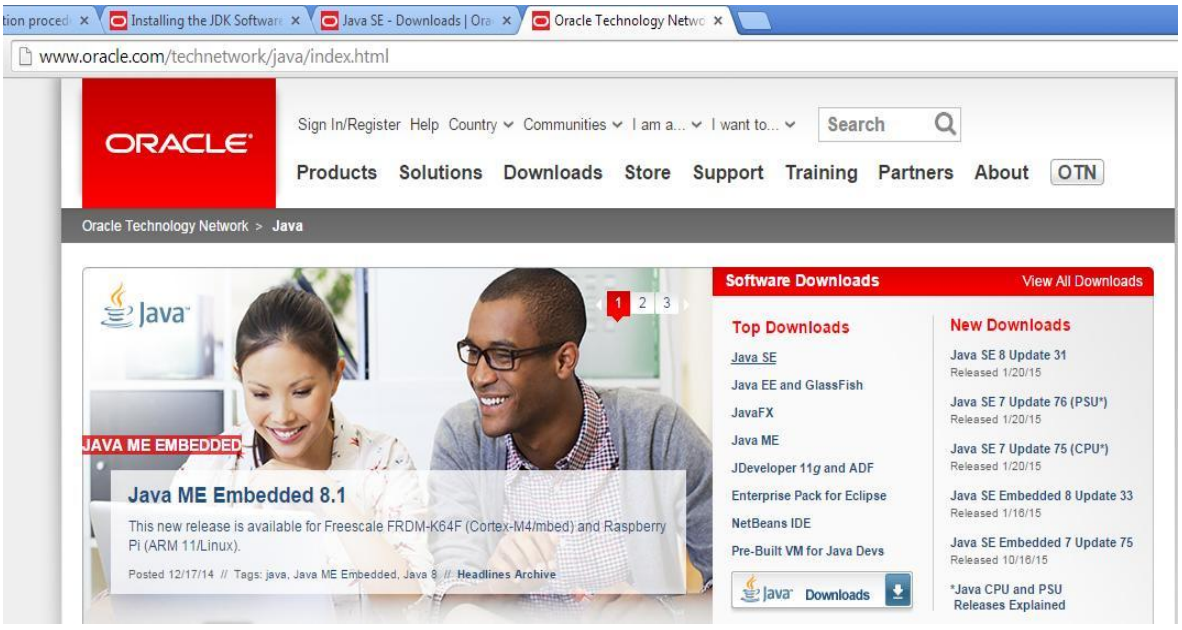
Microsoft Windows

JDK5: At least release 1.5.0_14

JDK6: At least release 1.6.0_03

To Install the JDK Software and Set `JAVA_HOME` on a Windows System

1. Install the JDK software.
   a. Go to `http://java.sun.com/javase/downloads/index.jsp`.

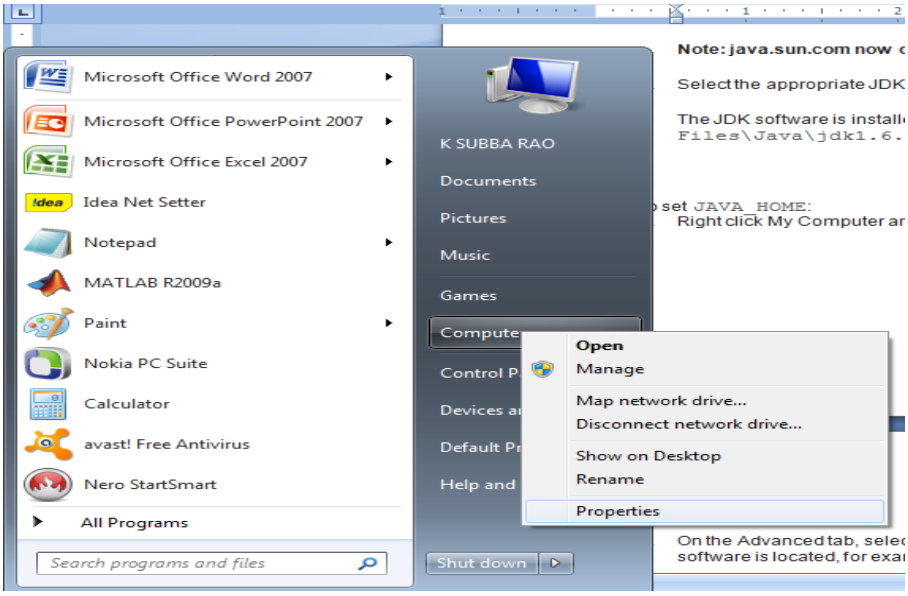


Note: java.sun.com now owned by oracle corporation

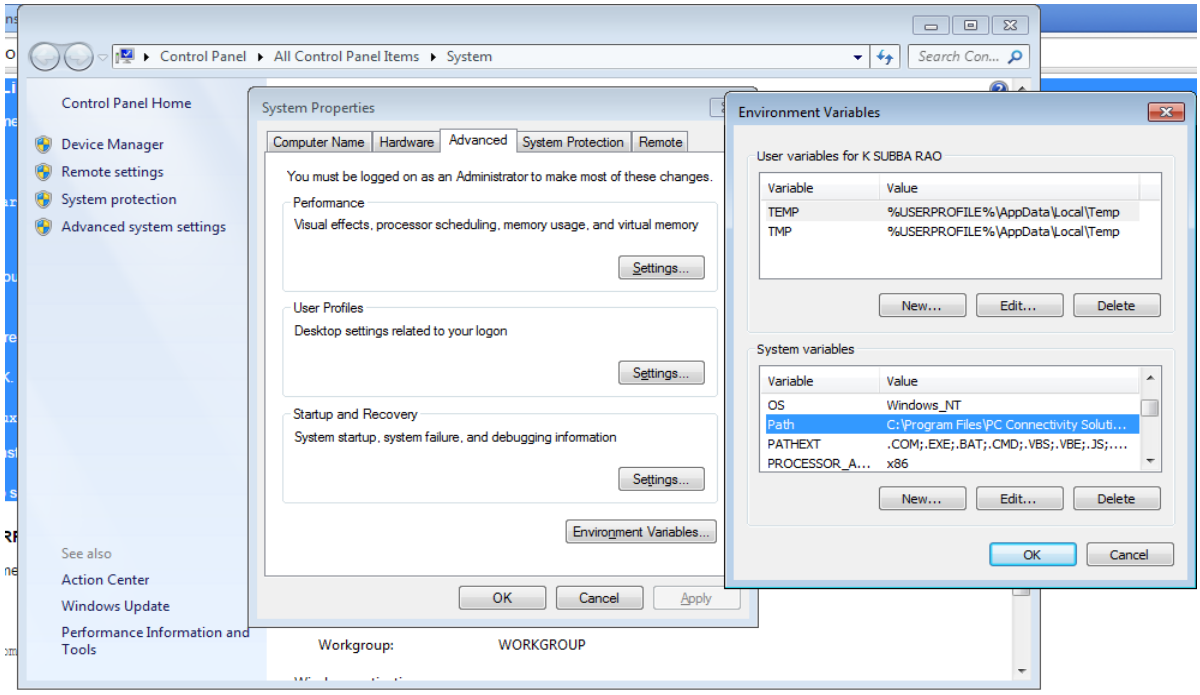
b. Select the appropriate JDK software and click Download.

The JDK software is installed on your computer, for example, at `C:\Program Files\Java\jdk1.6.0_02`. You can move the JDK software to another location if desired.

2. To set `JAVA_HOME`:
   a. Right click My Computer and select Properties.



b. On the Advanced tab, select Environment Variables, and then edit `JAVA_HOME` to point to where the JDK software is located, for example, `C:\Program Files\Java\jdk1.6.0_02`.



Installation of the 32-bit JDK on Linux Platforms

This procedure installs the Java Development Kit (JDK) for 32-bit Linux, using an archive binary file (`.tar.gz`).

These instructions use the following file:

```
jdk-8uversion-linux-i586.tar.gz
```

1. Download the file.

   Before the file can be downloaded, you must accept the license agreement. The archive binary can be installed by anyone (not only root users), in an location that you can write to. However, only the root user can install the JDK into the system location.

2. Change directory to the location where you would like the JDK to be installed, then move the `.tar.gz` archive binary to the current directory.

3. Unpack the tarball and install the JDK.

4. `% tar zxvf jdk-8uversion-linux-i586.tar.gz`

   The Java Development Kit files are installed in a directory called `jdk1.8.0_version` in the current directory.

5. Delete the `.tar.gz` file if you want to save disk space.