

Features of JAVA



Java Program Structure

- In the Java programming language:
 - A program is made up of one or more *classes*
 - A class contains one or more *methods*
 - A method contains program *statements*
- These terms will be explored in detail throughout the course
- A Java application always contains a method called `main`

Lincoln.java

```
public class Lincoln
{
    //-----
    // Prints a presidential quote.
    //-----
    public static void main (String[] args)
    {
        int a;
        System.out.println ("A quote by Abraham Lincoln:");

        System.out.println ("Whatever you are, be a good one.");
    }
}
```

Java Program Structure

```
// comments about the class
```

```
public class MyProgram  
{
```

class header



The diagram illustrates the structure of a Java program. It shows a code snippet with a comment line, a class declaration, and an opening curly brace. An orange arrow points from the text 'class header' to the 'public class MyProgram' line. A large orange curly brace on the left side of the code, spanning from the opening brace to the closing brace, is labeled 'class body'.

class body

Comments can be placed almost anywhere

```
}
```

Java Program Structure

```
// comments about the class
```

```
public class MyProgram  
{
```

```
    // comments about the method
```

```
    public static void main (String[] args)
```

```
    {
```



method body

```
    }
```

method header

Understanding Path structure

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, the path is not required to be set because all the tools will be available in the current directory.

However, if you have your Java file outside the JDK/bin folder, it is necessary to set the path of JDK.
There are two ways to set the path in Java:

Temporary

Permanent

1) How to set the Temporary Path of JDK in Windows

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

Open the command prompt

Copy the path of the JDK/bin directory

Write in command prompt: set path=copied_path

For Example:

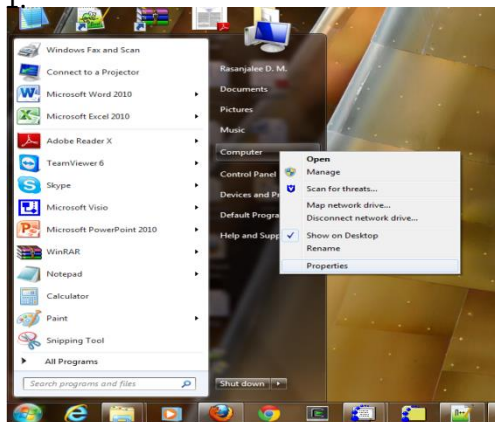
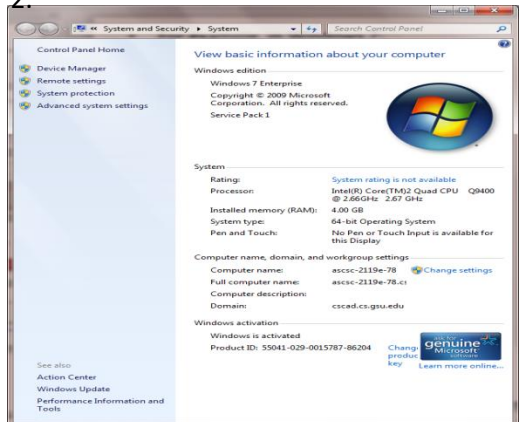
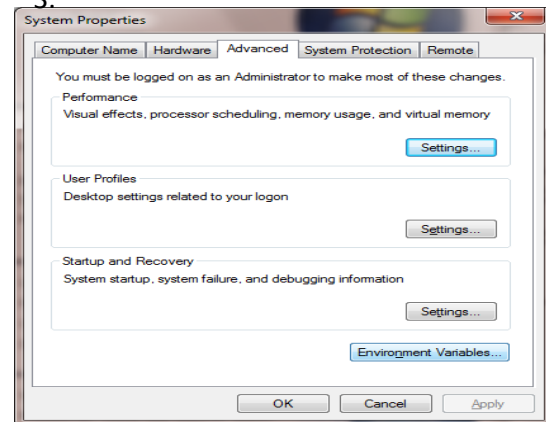
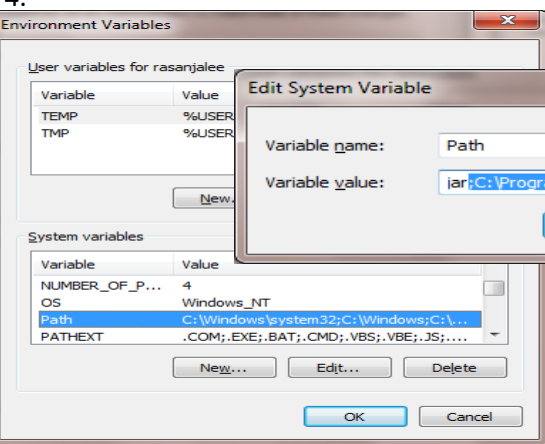
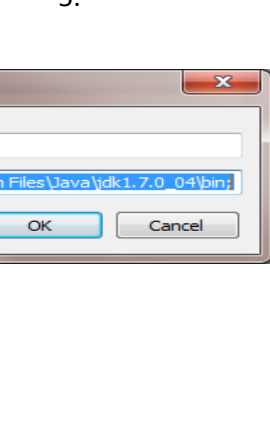
set path=C:\Program Files\Java\jdk1.6.0_23\bin

Let's see it in the figure given below:

Understanding class path

- For example, in Windows 7:

REF: <http://docs.oracle.com/javase/7/docs/webnotes/install/windows/jdk-installation-windows.html#Check>

- 1. Windows 7 desktop with Start menu open. The 'Show on Desktop' option is selected under the 'Computer' menu.
- 2. Windows 7 System Properties window showing system information. The 'Advanced' tab is selected, displaying details about the Windows edition, system, and hardware.
- 3. Windows 7 System Properties window showing environment variables. The 'Advanced' tab is selected, and the 'Environment Variables...' button is visible.
- 4. Windows 7 Environment Variables dialog box. The 'System variables' tab is selected, and the 'Path' variable is highlighted in the list.
- 5. Windows 7 Edit System Variable dialog box. The 'Variable name' is 'Path' and the 'Variable value' is 'jar;C:\Program Files\Java\jdk1.7.0_04\bin;'. The 'OK' button is visible.

Add the location of the bin folder of the JDK installation for the PATH variable in **System Variables** in Environmental variables.

The following is a typical value for the PATH variable:

`C:\WINDOWS\system32;C:\WINDOWS;C:\Program Files\Java\jdk1.7.0\bin`

Java Data Types



Data Types

Primitive

numeric

integer

byte

short

int

long

floating point

double

float

non - numeric

character

boolean

Non Primitive

strings

arrays

user defined
classes

Data Types

- Variables are nothing but **reserved memory locations** to store values. This means that when you create a variable you **reserve some space in memory**.
- All variables must first be declared before they can be used.
- Based on the **data type** of a variable, **the operating system allocates memory** and **decides what can be stored in the reserved memory**.
- Therefore, by assigning different data types to variables, you can store **integers**, **decimals**, or **characters** in these variables.
- The Java programming language is **statically-typed**.

Data Types (Cont.)

- This involves stating the variable's type and name
 - Foreg: `int a=1;`
- Doing so tells your program that
 - **a field named "a" exists, holds numerical data, and**
 - **has an initial value of "1".**
- A variable's data type determines the values it may contain, plus the operations that may be performed on it.
- There are two categories of data types available in Java:
 - **Primitive Data Types**
 - **Reference/Object Data Types**

Primitive Data Types (Cont.)

Data Type	Size (in Bytes)	Description	Example
float (Floating point Type)	4	<ul style="list-style-type: none">❑ Store single-precision floating point numbers.❑ Default value is 0.0f	float pi=3.14f;
double (Floating point Type)	8	<ul style="list-style-type: none">❑ Store double-precision floating point numbers.❑ Default value is 0.0d.	double pi=3.14; double pi=3.14d;
boolean	Not defined	<ul style="list-style-type: none">❑ Represents one bit of information.❑ Only two possible values: true and false.❑ Used for simple flags that track true/false conditions.❑ Default value is false.	boolean flag = true; boolean flg = false;
char	2	Single character	char ch='a';

Primitive Data Types

- There are 8 primitive data types supported by Java.
- Primitive data types are **predefined by the language and named by a key word.**

Data Type	Size (in Bytes)	Description	Example
byte	1	<ul style="list-style-type: none">❑ Store signed integer.❑ Save memory.❑ Default value is 0	byte a = -128; byte b = 127;
short	2	<ul style="list-style-type: none">❑ Store signed integer.❑ Default value is 0.	short s = -32768; short r = 32767;
int	4	<ul style="list-style-type: none">❑ Large enough for the numbers❑ Default value is 0.	int a = - 2147483648; int b = 2147483647;
long	8	<ul style="list-style-type: none">❑ Range of values wider than those provided by int.❑ Default value is 0L.	long a = 123L; long b = -123L;

Default Values

- **Fields** that are declared but not initialized will be set to a reasonable default value by the compiler but not to the local variables.
- The following chart summarizes the default values for the different data types.

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Reference Data Types

- Reference variables are created using **defined constructors of the classes**.
- They are used to **access objects**.
- These variables are declared to be of a **specific type** that cannot be changed.
 - ❖ **For example:** Employee, Student etc.
- It includes:
 - ❖ **Class objects**
 - ❖ **Various type of array variables**
- Default value of any reference variable is **null**.
- Used to refer to **any object of the declared type**.
E.g. : **Animal** animal = new **Animal**("giraffe");