COCSIT

LANGUAGE BASIC PART I

COCSIT

Language Basics

Objectives

In this module you will learn about :

- Path and Classpath
- Command line arguments
- Keywords
- Basic data types
- Types of Operators.

Langugage Basics




A Simple Java Program – The Set Up

- Before we learn about writing a simple java program, let us understand what preparations are needed for running a java program from console

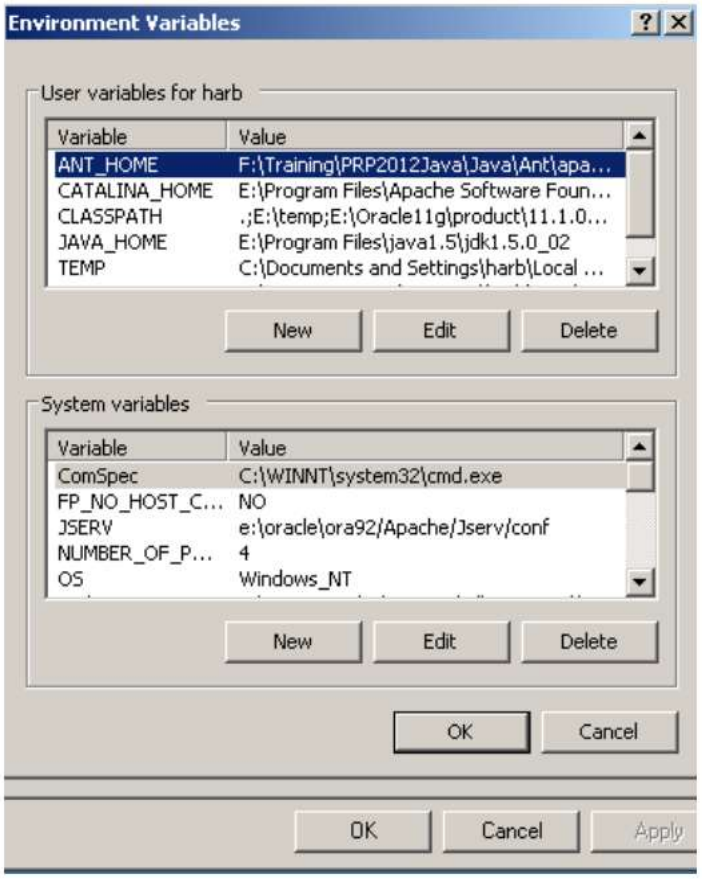
- We have to define PATH and CLASSPATH parameters and create necessary directories(folders) where we will be storing all our program files

PATH

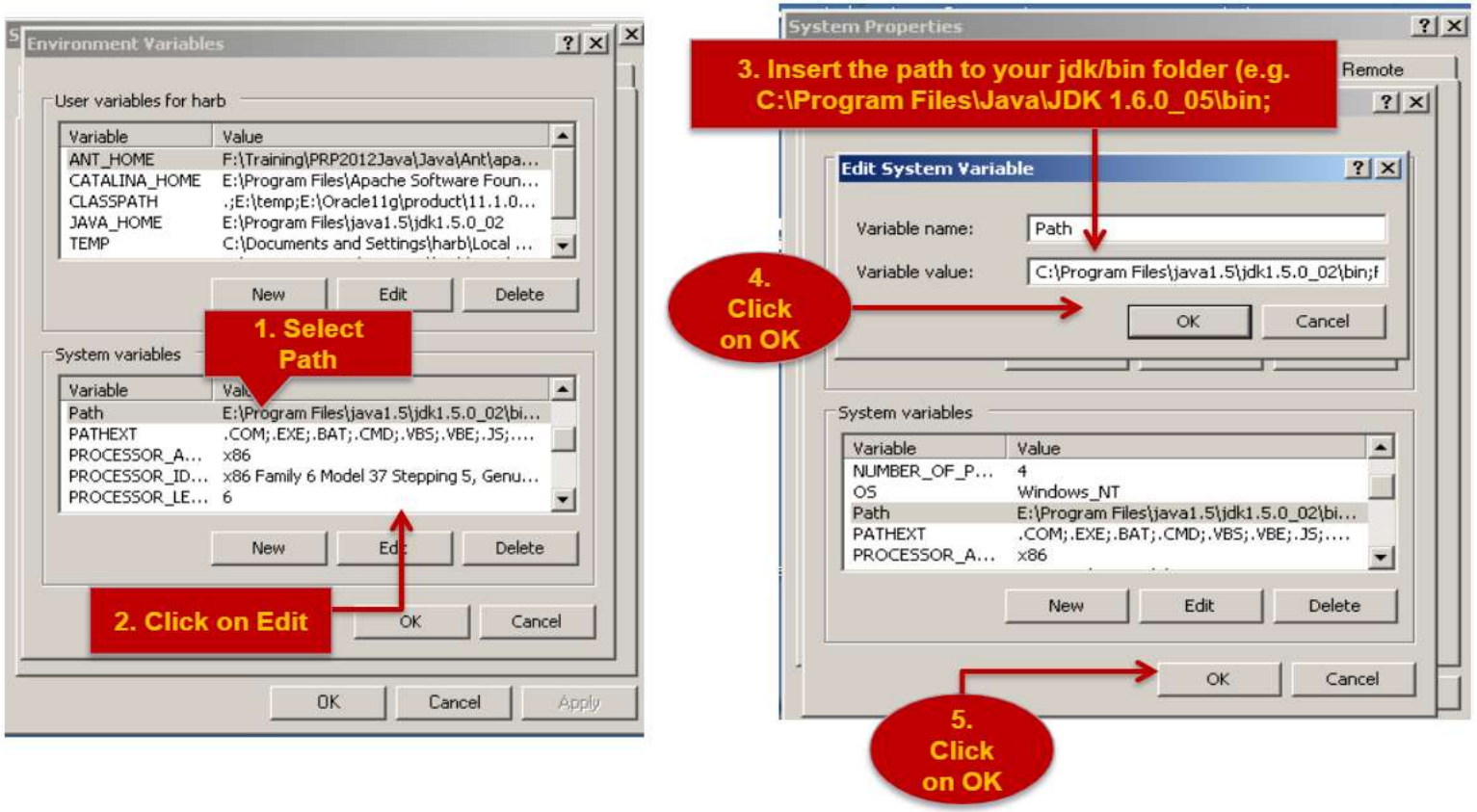
- PATH is an *environmental variable* in DOS(Disk Operating System), Windows and other operating systems like Unix.
- PATH tells the operating system which directories(folders) to search for executable files, in response to commands issued by a user .
- It is a convenient way of executing files without bothering about providing the absolute path to the folder, where the file is located.

How to set PATH ?

1. Right Click My Computer
2. Select Properties
3. You will get to see the Properties Page of My Computer
4. Select Advanced Tab
5. Select Environment Variables
6. You will see Environment Variables Page as displayed here



How to set PATH ? (Contd.).

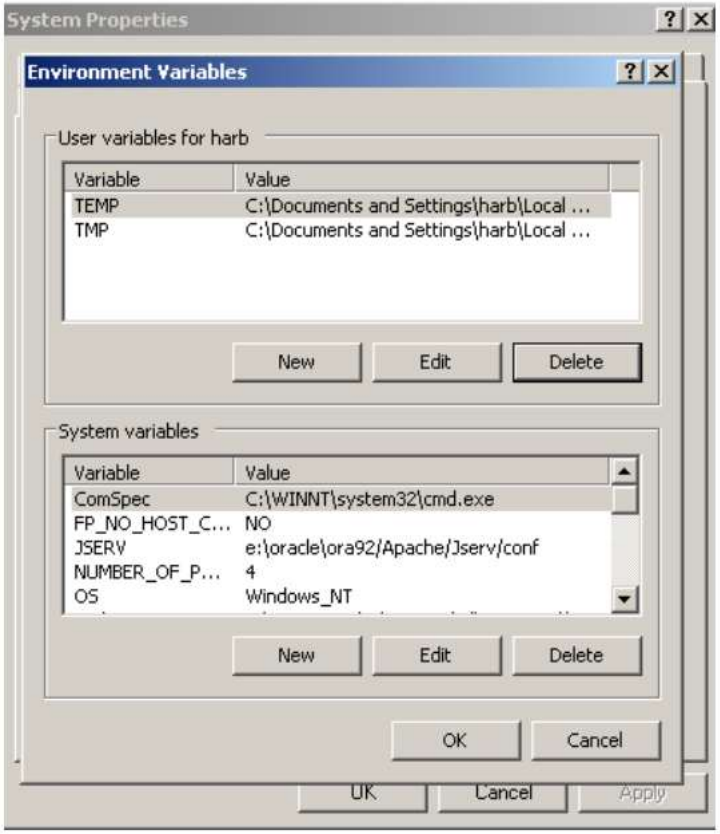


CLASSPATH

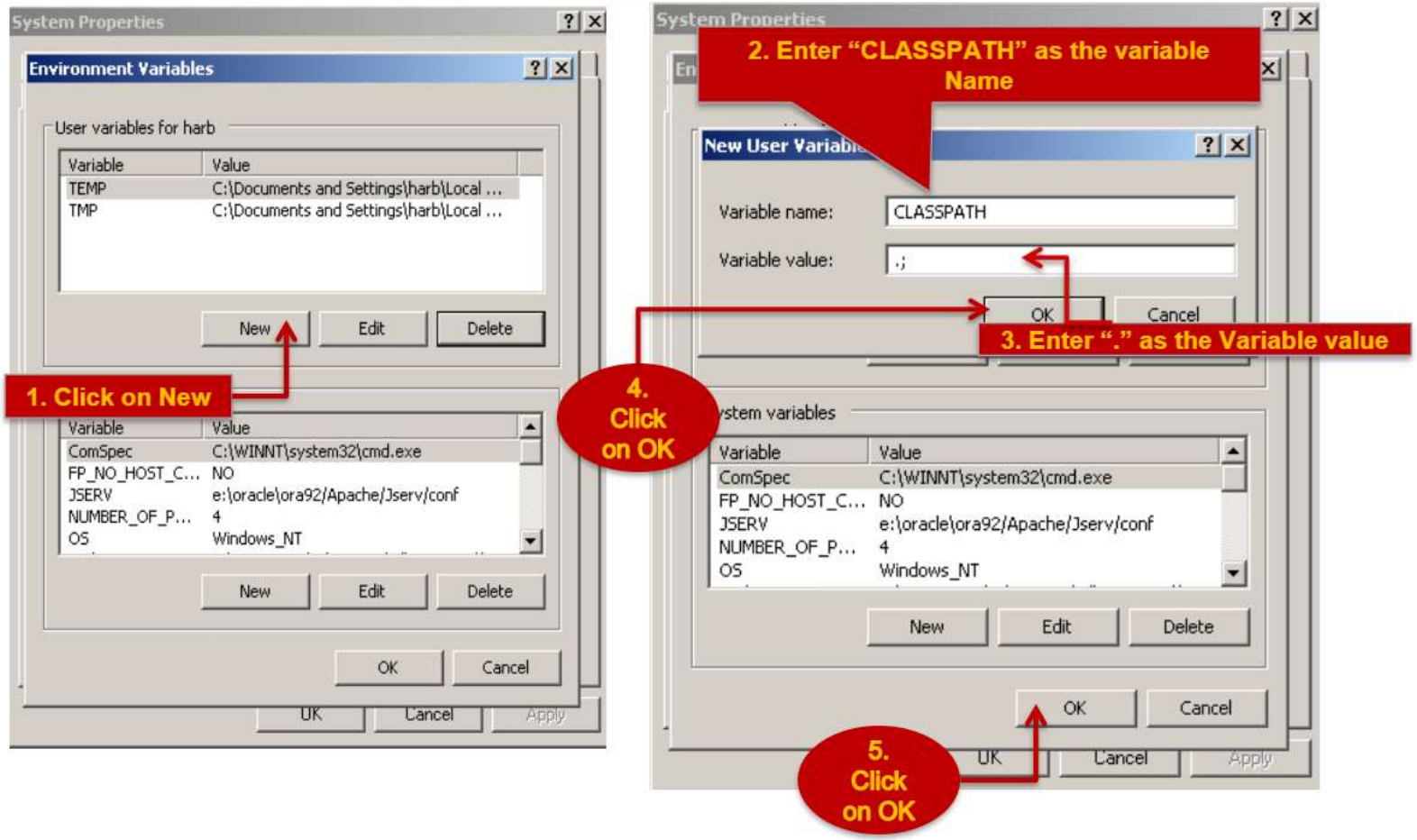
- CLASSPATH is a parameter which tells the JVM or the Compiler, where to locate classes that are not part of Java Development ToolKit(JDK).
- CLASSPATH is set either on command-line or through environment variable.
- CLASSPATH set on command-line is temporary in nature, while the environment variable is permanent.

How to set CLASSPATH ?

1. Right Click My Computer
2. Select Properties
3. You will get to see the Properties Page of My Computer
4. Select Advanced Tab
5. Select Environment Variables
6. You will see Environment Variables Page as displayed here



How to set CLASSPATH ? (Contd.).



A Simple Java Program

Our first Java Program:

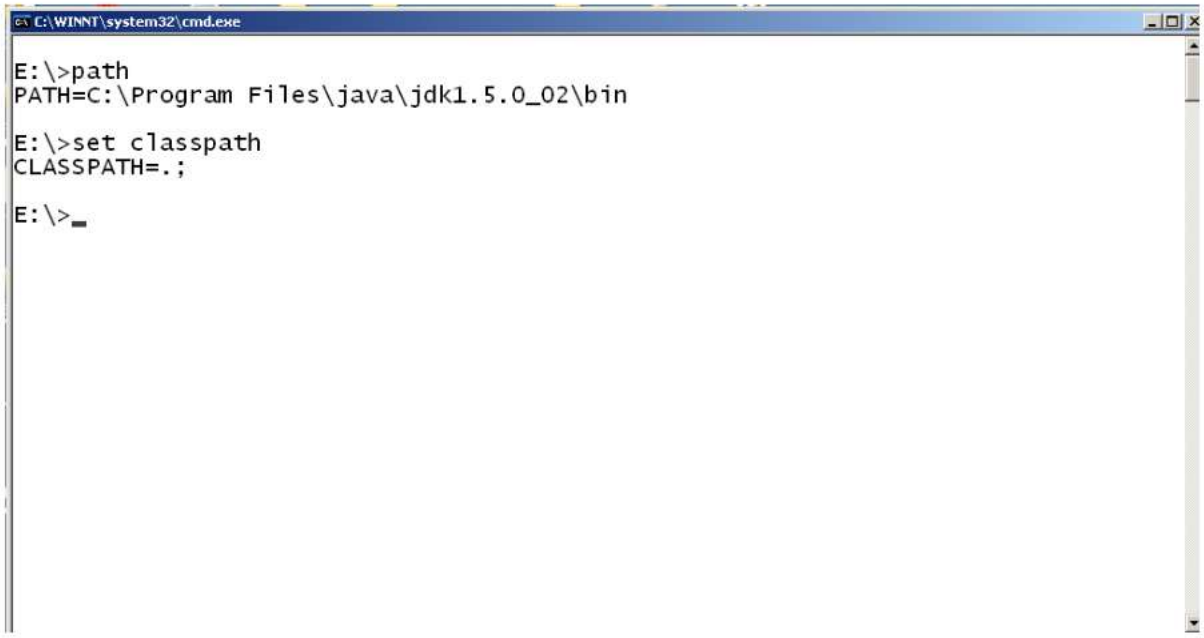
```
public class Welcome {
     public static void main(String args[]) {
          System.out.println("Welcome..!");
    }
}
```

This program displays the output "Welcome..!" on the console

Create source file	:	Welcome.java
Compile	:	javac Welcome.java
Execute	:	java Welcome

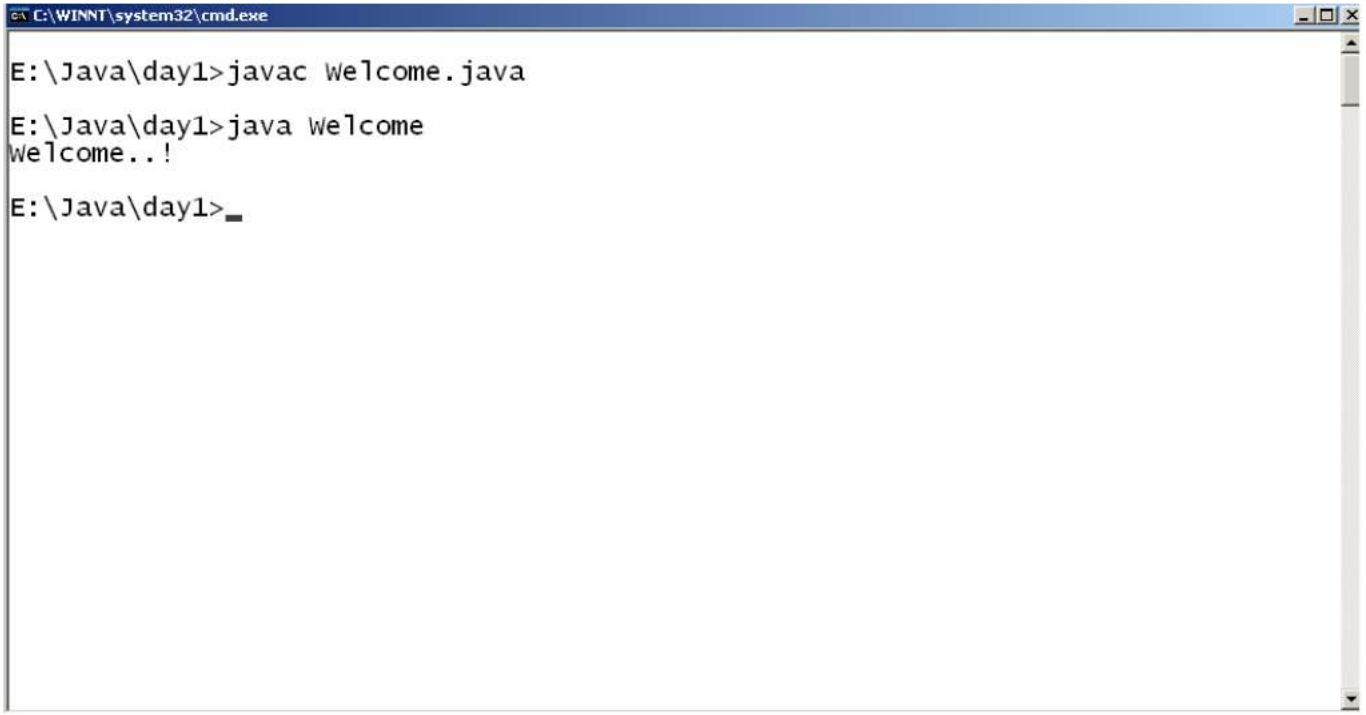
Executing your first Java Program

- Before executing the program, just check whether the PATH and the CLASSPATH parameters are properly set, by typing in the commands as shown in the screen below:



Executing your first Java Program (Contd.).

- Now compile and execute your program as given below :



Quiz

Sample.java file contains class A, B and C. How many .class files will be created after compiling Sample.java ? What is your observation ?

Sample.java

```
class A {
    void m1() { }
}
class B {
    void m2() { }
}
class C {
    void m3() { }
}
```

Quiz(Contd..)

What will be the result if you try to compile and execute the following program ?

Reason out :

Sample.java

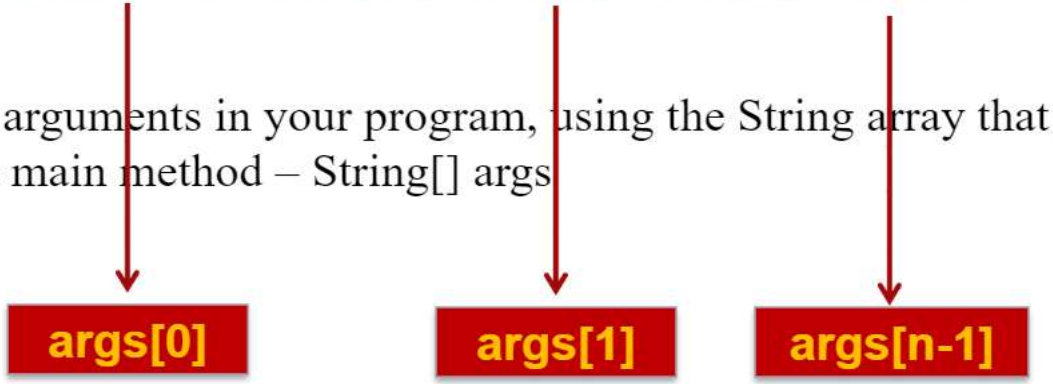
```
class Sample {
    public static void main() {
     System.out.println("Welcome");
    }
}
```

a. Compilation Error
b. Runtime Error
c. The program compiles and executes successfully but prints nothing.
d. It will print "Welcome"

Accessing command line arguments

When you execute a java program, you can pass command line arguments in the following way :

```
java Simple <argument1> <argument2>….<argument-n>
```



You can access these arguments in your program, using the String array that you have passed as an argument to the main method – String[] args

Passing command line arguments

```
class Simple {
   static public void main(String[] args) {
     System.out.println(args[0]);
   }
}
```

When we compile the above code successfully and execute it as Java Simple Wipro, the output will be :

```
Wipro
```

Accessing numeric command line arguments

```
class Simple {
  static public void main(String[] args) {
    int i1 = Integer.parseInt(args[0]);
    int i2 = Integer.parseInt(args[1]);
    System.out.println(i1+i2);
  }
}
```

When we compile the above code successfully and execute it as Java Simple 10 20, the output will be :

30

Finding length of an Array

- How to find the number of command line arguments that a user may pass while executing a java program?

- The answer to the above question lies in **args.length**, where args is the String array that we pass to the main method and length is the property of the Array Object

Example on Finding length of an Array

```
class FindNumberOfArguments {
    public static void main(String[ ] args) {
        int len = args.length;
        System.out.println(len);
    }
}
```

If you compile the above code successfully and execute it as java FindLength A B C D E F, the result will be

6

And if you execute it as java FindLength Tom John Lee, the result will be

3

Quiz

What will be the result if you try to compile and execute the following code without passing any command line argument?

```
class Sample {
    public static void main(String [ ] args) {
            int len = args.length;
            System.out.println(len);
    }
}
```

a. Compilation Error

b. Runtime Error

c. The program compiles and executes successfully but prints nothing.

d. The program compiles and executes successfully & prints **0**.

Good Programming Practices

Naming Conventions

Class Names

- Class names should be **nouns**, in mixed case with the first letter of each internal word capitalized
- Class names should be simple and descriptive
- Eg: class **Student**, class **TestStudent**

Variable Names

- The variables are in mixed case with a lowercase first letter
- Variable names should not start with underscore _ or dollar sign $ characters, even though both are allowed
- Variable names should be small yet meaningful
- One-character variable names should be avoided except for temporary "throwaway" variables
- Eg: int y,myWidth;

Good Programming Practices(Contd.).

Naming Conventions

Method Names

- Methods should be **verbs**, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized
- Eg: void run(), void getColor()

Comments

Block Comments

- Block comments are used to provide descriptions of files, methods, data structures and algorithms
- Block comments may be used at the beginning of each file and before each method

```
/*
    Here is a block comment
*/
```

Good Programming Practices(Contd.).

Comments

Single line Comment

- Single line comments can be written using // Single line

Number per Line

- One declaration per line is recommended

```
int height;
int width;
```

is preferred over

```
int height,width;
```

Do not put different types on the same line

```
int height,width[];      // Not recommended
```

The Java API

- An application programming interface(API), in the framework of java, is a collection of prewritten packages, classes, and interfaces with their respective methods, fields and constructors

- The Java API, included with the JDK, describes the function of each of its components

- In Java programming, many of these components are pre-created and commonly used

The Java Buzzwords

- Simple
- Object-Oriented
  - Supports encapsulation, inheritance, abstraction, and polymorphism
- Distributed
  - Libraries for network programming
  - Remote Method Invocation
- Architecture neutral
  - Java Bytecodes are interpreted by the JVM

The Java Buzzwords (Contd.).

- Secure
  - Difficult to break Java security mechanisms
  - Java Bytecode verification
  - Signed Applets
- Portable
  - Primitive data type sizes and their arithmetic behavior specified by the language
  - Libraries define portable interfaces
- Multithreaded
  - Threads are easy to create and use

COCSIT

Language Basics

- Keywords
- Data Types
- Variables
- Operators
- Conditional Statements
- Loops

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

Quiz

What will be the result, if we try to compile and execute the following code?

```
class Test {
    public static void main(String [ ] ar) {
        int for=2;
        System.out.println(for);
    }
}
```

Primitive Data Types

Data Type	Size (in bits)	Minimum Range	Maximum Range	Default Value (for fields)
byte	8	-128	+127	0
short	16	-32768	+32767	0
int	32	-2147483648	+2147483647	0
long	64	-9223372036854775808	+9223372036854775807	0L
float	32	1.40E-45	3.40282346638528860e+38	0.0f
double	64	4.94065645841246544e-324d	1.79769313486231570e+308d	0.0d
char	16		0 to 65,535	'\u0000'
boolean	1	NA	NA	false

Quiz

What will be the result, if we try to compile and execute the following code?

```
class Test {
    public static void main(String [ ]  ar) {
        byte b=128;
        System.out.println(b);
    }
}
```

Quiz(Contd..)

What will be the result, if we try to compile and execute the following code?

```
class Test {
    public static void main(String ar[]) {
        float f=1.2;
        boolean b=1;
        System.out.println(f);
        System.out.println(b);
    }
}
```

Quiz(Contd..)

What will be the result, if we try to compile and execute the following code?

```
class Test {
    public static void main(String ar[]) {
        double d=1.2D;
        System.out.println(d);
    }
}
```


Quiz(Contd..)

What will be the result, if we try to compile and execute the following code?

```
class Test {
    public static void main(String [ ] args) {
        int 9A=10;
        System.out.println(9A);
    }
}
```

Quiz(Contd..)

What will be the result, if we try to compile and execute the following code?

```
class Test {
    public static void main(String [ ] args) {
        int x;
        System.out.println(x);
    }
}
```

Quiz

1. Match the following table:

DATA TYPES	SIZE(bytes)
char	4
byte	2
int	1
double	8

Types of Variables

The Java programming language defines the following kinds of Variables:

- Local Variables
  - Tied to a method
  - Scope of a local variable is within the method
- Instance Variables (Non-static)
  - Tied to an object
  - Scope of an instance variable is the whole class
- Static Variables
  - Tied to a class
  - Shared by all instances of a class

Quiz

What will be the output for the below code ?

```
public class Sample {
    public static void main() {
     int i_val = 10, j_val = 20;
     boolean chk;
     chk = i_val < j_val;
     System.out.println("chk value: "+chk);
    }
}
```


Thanking
You