

**JAVA PROGRAMMING**  
**COURSE NAME: MCA – 5<sup>TH</sup> SEMESTER**  
**LAB MANUAL**

## WEEK 1:

### (b) Setting the PATH variable to the appropriate directory location.

#### 1. Setting the PATH variable temporarily:

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="Path of the bin folder in java"
- Example:

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

Setting the PATH variable permanently in Java:

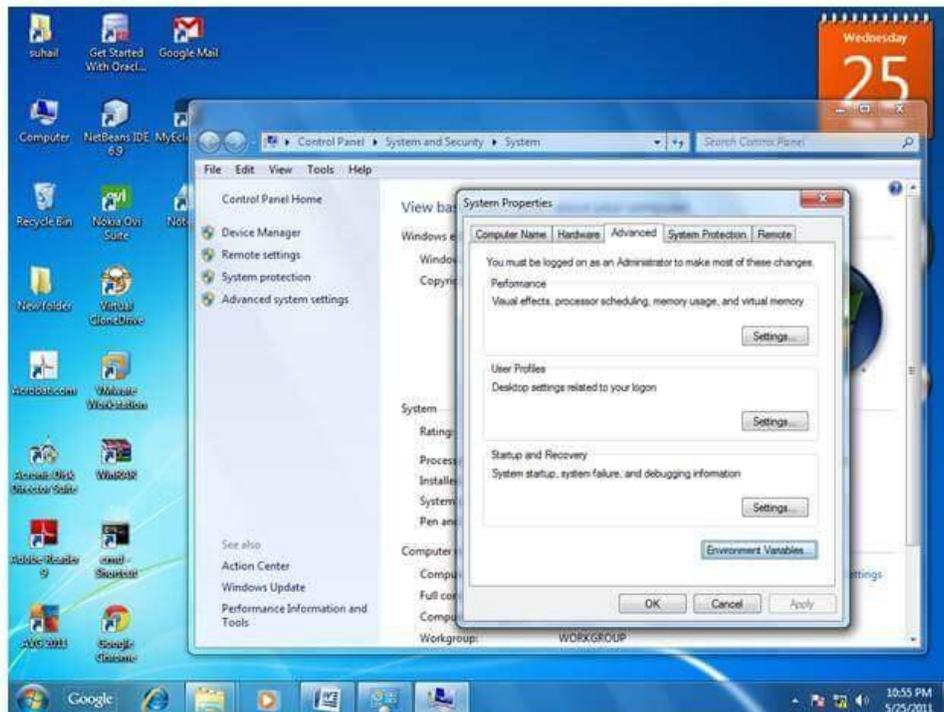
#### 1) Go to MyComputer properties



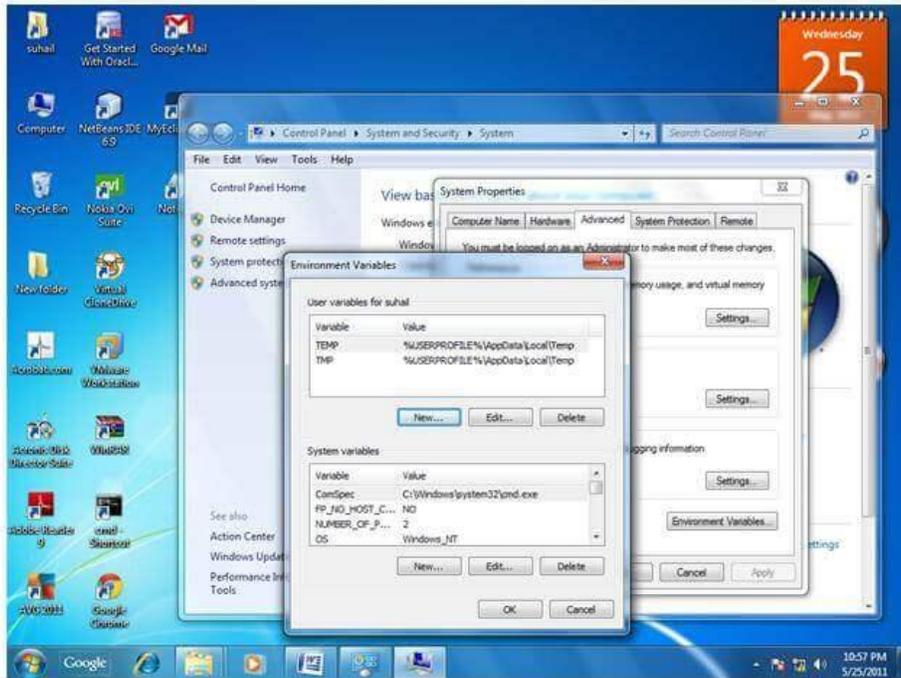
## 2) Click on the advanced tab



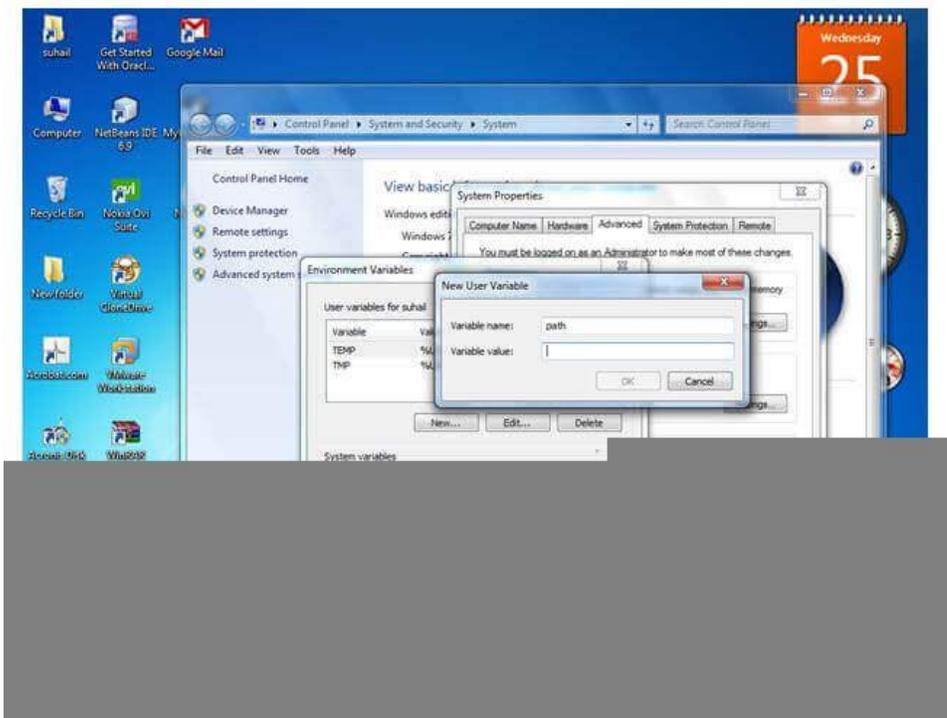
## 3) Click on environment variables



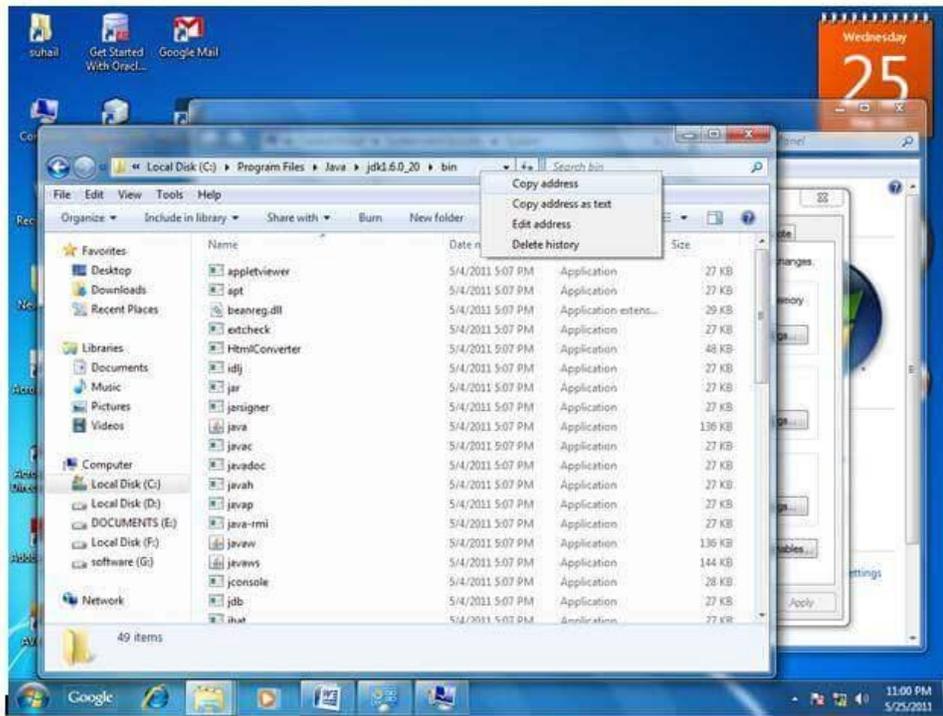
#### 4) Click on the new tab of user variables



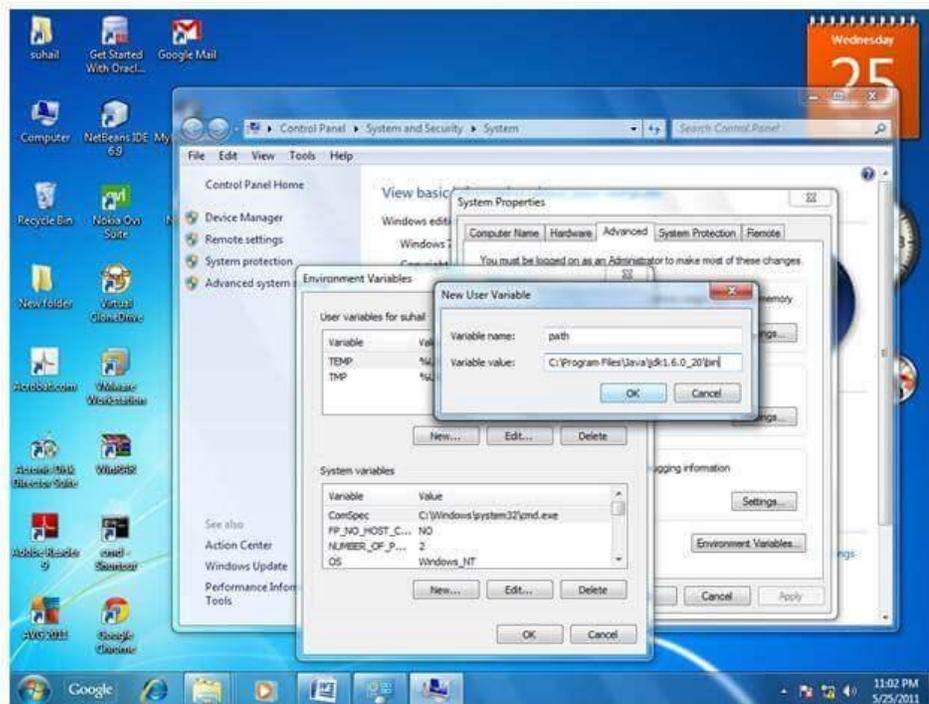
#### 5) Write the path in the variable name



**6) Copy the path of bin folder**



**7) Paste path of bin folder in the variable value**



**8) Click on ok button till all the tabs are closed.**

WEEK 2:

**(a) Program to display “HelloWorld!” on the screen.**

```
/*My first Java Program*/
class first
{
    public static void main(String args[])
    {
        System.out.println("Hello World!");
    }
}
```

**(b) Write a Java program that receives two integer numbers via keyboard, does their summation, and displays the result. Ensure that only integer values are processed.**

```
//program to determine the summation of two integers inputted via keyboard
import java.util.Scanner;
class summation
{
    public static void main(String args[])
    {
        Scanner input=new Scanner(System.in);
        System.out.print("Enter first number");
        int number1=input.nextInt();
        System.out.print("Enter second number");
        int number2=input.nextInt();
        int result;
        result=number1+number2;
        System.out.println("Result obtained after addition is: " +result);
    }
}
```

**(c) Write a Java program that creates a 2D integer array with 5 rows and varying number of columns in each row. Using ‘for each’ variant of for loop display each element of every row.**

**//Using for each variant of the for loop to display elements of a 2D array**

```
import java.util.Scanner;
class twodarray
{
    public static void main(String args[])
    {
```

```

int twoD[][]=new int[5][];
twoD[0]=new int[3];
twoD[1]=new int[5];
twoD[2]=new int[4];
twoD[3]=new int[6];
twoD[4]=new int[2];
int k=0;
Scanner input=new Scanner(System.in);
System.out.print("Enter array elements");
for(int i=0;i<5;i=i+1)
{
    k=0;
    for(int j:twoD[i])
    {
        twoD[i][k++]=input.nextInt();
    }
}
//for eachvariant to print the array
for(int i=0;i<5;i=i+1)
{
    for(int j:twoD[i])
    {
        System.out.print(j+" ");
    }
    System.out.println("");
}
}
}

```

**Do it yourself:**

- (a) Write a Java program that prints the season name corresponding to its month number using *If-else* and *switch-case* statements.
- (b) Write a Java program that sorts (using *bubble sort*) an integer array using *for* loop.
- (c) Write a Java program that calculates factorial of a number (inputted via keyboard) recursively.

## WEEK 3:

- (a) Write a Java program in which a Class overloads a method *sum()*, which takes 2 parameters. The overloaded methods should perform summation of either integer or floating-point values.

```
// Java program to demonstrate working of method overloading in Java.
```

```
import java.util.Scanner;
class Sum
{
    // Overloaded sum(). This sum takes two int parameters
    public int sum(int x, int y)
    {
        return (x + y);
    }

    // Overloaded sum(). This sum takes two double parameters
    public double sum(double x, double y)
    {
        return (x + y);
    }

    public static void main(String args[])
    {
        int i=1,option;
        Scanner input=new Scanner(System.in);
        Sum s=new Sum();
        while(i==1)
        {
            System.out.println("Press (1) to add two integers");
            System.out.println("Press (2) to add two floating point numbers");
            System.out.println("Press (3) to exit");
            option=input.nextInt();
            switch(option)
            {
                case 1:
                    System.out.print("Enter first number");
                    int number1=input.nextInt();
                    System.out.print("Enter second number");
                    int number2=input.nextInt();
```

```

System.out.println("The sum of numbers=" +s.sum(number1,number2));
break;

case 2:
System.out.print("Enter first number");
double num1=input.nextDouble();
System.out.print("Enter second number");
double num2=input.nextDouble();
System.out.println("The sum is equal to " +s.sum(num1,num2));
break;

case 3:
i=0;
break;
} //switch
} //while
} //main
} //class ends

```

### Do it Yourself:

- (a) Write a Java program that creates a Class, namely *Student*.
- i. Ensure that *Age* instance variable of the Class is never accessed directly, and its value is never less than 4 and greater than 40 for any Object of the Class (use methods to validate and assign the value).
  - ii. Ensure that the constructor always assigns a unique value to *Enrollment\_No* instance variable for every Object of the Class (use a *static* class variable for counting objects, say *Object\_Counter*).
  - iii. Ensure that when an Object is removed, the *Object\_Counter* is automatically decremented (use *finalize()*), and whenever required the variable can only be accessed using a method even without an Object reference (make the counter private and use a *static* method to access it).

## WEEK 4

Write a Java program that creates a Class namely *A* that has a private instance variable and method, a protected instance variable and method, a default instance variable and method, and a public instance variable and method. Create another Class say *B* that inherits from *A*.

- a. Show that all except private members are inherited.

```
import java.lang.*;
class A
{
    private int a;
    protected int b;
    int c;
    public int d;

    private void show_a()
    {
        System.out.println("Value of private variable a =" +a);
    }

    protected void show_b()
    {
        System.out.println("Value of protected variable b =" +b);
    }

    void show_c()
    {
        System.out.println("Value of default variable c =" +c);
    }

    public void show_d()
    {
        System.out.println("Value of public variable d =" +d);
    }

}

class B extends A
{

    /*void update_a()
    {
        a=a+10;
        System.out.println("Value of private variable a =" +a); wont compile as a is a private
        member of the super class A
    }*/
}
```

```

void update_b()
{
b=b+10;
System.out.println("Value of protected variable b after update =" +b);
}

void update_c()
{
c=c+7;
System.out.println("Value of default variable c after update =" +c);
}

void update_d()
{
d=b+c;
System.out.println("Value of public variable d after update =" +d);
}

public static void main(String args[])
{
B obj1=new B();
//obj1.show_a(); //wont compile as show_a is a private method
obj1.show_b();
obj1.show_c();
obj1.show_d();
//obj1.update_a();
obj1.update_b();
obj1.update_c();
obj1.update_d();
}
}

```

**ii. Show that an inherited instance variable can be shadowed (with the same or weaker access visibility) but can be accessed using *super* keyword in the sub-class.**

```

import java.lang.*;
class A
{
private int a;
protected int b;
int c;
public int d;

private void show_a()
{

```

```

System.out.println("Value of private variable a =" +a);
}

protected void show_b()
{
System.out.println("Value of protected variable b =" +b);
}

void show_c()
{
System.out.println("Value of default variable c =" +c);
}

public void show_d()
{
System.out.println("Value of public variable d =" +d);
}

}

class B extends A
{

private int b=30;

void update_b()
{
b=b+10; //local variable b accessed
System.out.println("Value of protected variable b after update =" +b); //output b=40
b=super.b+10; //superclass variable b accessed
System.out.println("Value of protected variable b after update =" +b); //output b=10
}

void update_c()
{
c=c+7;
System.out.println("Value of default variable c after update =" +c);
}

void update_d()
{
d=b+c;
System.out.println("Value of public variable d after update =" +d);
}
}

```

```

public static void main(String args[])
{
    B obj1=new B();
    obj1.show_b();
    obj1.show_c();
    obj1.show_d();
    obj1.update_b();
    obj1.update_c();
    obj1.update_d();
}
}

```

**iii. Show that the reference variable of type A or B can't access an overridden method of A in the Object of B.**

```

import java.lang.*;
class A
{
    void callme()
    {
        System.out.println("Inside class A's callme method");
    }
}

```

```

class B2 extends A
{
    void callme()
    {
        System.out.println("Inside class B's callme method");
    }
}

```

```

public static void main(String args[])
{
    A a=new B();
    B b=new B();
    b.callme(); //B's callme method called when a reference variable of type A refers to an object of B
    a.callme(); //B's callme method called
}
}

```

**Do it yourself:**

- i. Show that an inherited method can be overridden (with the same or weaker access visibility) but can be accessed using *super* keyword in the sub-class.
- ii. Show that the reference variable of type A can access a shadowed data member of A in the Object of B.