**Session 2**

**Assignment 2.1**

Student Name: Karthik K

Course: Big Data Hadoop & Spark Training

## Program 1: Addition of Two numbers

Write a java code with the class named ‘acad’ and a method ‘main’. Hard Code the program with two integers and print the sum of those two.

**package NICE;**

**public class acad {**

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

**{**

**// TODO Auto-generated method stub**

**int a=12;**

**int b=13;**

**int c=a+b;**

**System.out.print("The Value of C: ");**

**System.out.println(c);**

**}**

**}**

**Output:**



## Program 2: Addition, Runtime Input

Rewrite the above code, where, inputs are provided by the user at runtime and the output is printed.

**public** **class** acad

{

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

{

// **TODO** Auto-generated method stub

**int** a,b;

**int** add;

Scanner buf=**new** Scanner(System.***in***);

System.***out***.println("Enter the value of a: ");

a=buf.nextInt();

System.***out***.println("Enter the Value of b: ");

b=buf.nextInt();

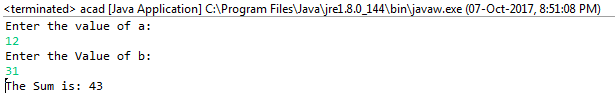
add=a+b;

System.***out***.println("The Sum is: " +add);

}

}

Output:



## Program 3: Addition using the method sum()

Write a program with method name sum() that accepts two parameters from user and print the sum of two numbers. Output format should be as:

**First number is:**

**Second number is:**

**Sum is:**

**public** **class** acad {

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

{

// **TODO** Auto-generated method stub

*sum*();

}

**public** **static** **void** sum()

{

Scanner buf=**new** Scanner(System.***in***);

System.***out***.print("First number is:");

**int** a=buf.nextInt();

System.***out***.print("Second number is:");

**int** b=buf.nextInt();

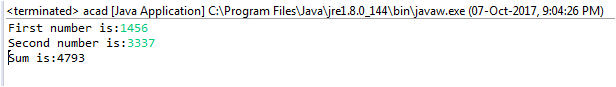
**int** c=a+b;

System.***out***.print("Sum is:" +c);

}

}

Output:



## Program 4: Odd and Even

Write a program to accept two numbers from **stdin** and find all the odd as well as even numbers present in between them.

**public** **class** OddEven

{

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

{

**int** n;

Scanner buf=**new** Scanner(System.***in***);

System.***out***.print("Enter the no. Of Elements: ");

n=buf.nextInt();

**int** a[]=**new** **int**[n];

System.***out***.println("Enter all elements: ");

**for** (**int** i=0; i<n; i++)

{

a[i]=buf.nextInt();

}

System.***out***.print("Odd numbers:");

**for**(**int** i = 0 ; i < n ; i++)

{

**if**(a[i] % 2 != 0)

{

System.***out***.print(a[i]+" ");

}

}

System.***out***.println("");

System.***out***.print("Even numbers: ");

**for**(**int** i = 0 ; i < n ; i++)

{

**if**(a[i] % 2 == 0)

{

System.***out***.print(a[i]+" ");

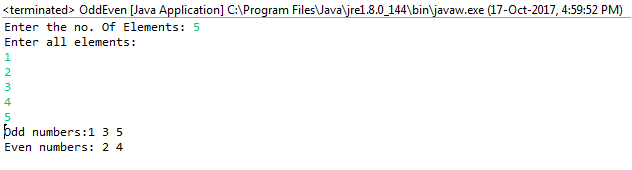
}

}

}

}

Output:



## Program 5: Multiplication

Joe is scared to go to school. When her dad asked the reason, joe said she is unable to complete the task given by her teacher. The task was to find the “first 10 multiples” of the number entered from stdin . Eg:

Input: 3

O/p:

3 x 1 = 3

3 x 2 = 6

………

3 x 10 = 30

Help Joe in completing the task!

**public** **class** Multplication {

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

{

// **TODO** Auto-generated method stub

Scanner buf=**new** Scanner(System.***in***);

System.***out***.print("Enter The Input Value: ");

**int** n=buf.nextInt();

**for** (**int** i=1;i<=10;i++)

{

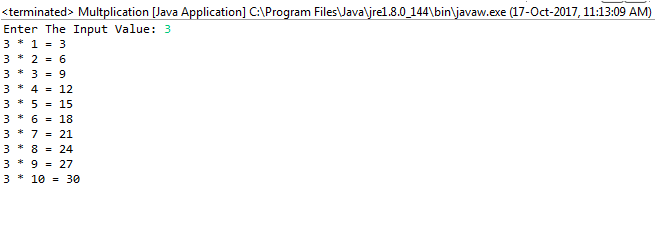
System.***out***.println(n+" \* "+i+" = "+n\*i);

}

}

}

Output:



## Program 6: Method Over Loading

### Method overloading

If two or more method in a class have same name but different parameters, it is known as method overloading. Overloading always occur in the same class (unlike method overriding).

Method overloading is one of the ways through which java supports polymorphism. Method overloading can be done by changing number of arguments or by changing the data type of arguments. If two or more method have same name and same parameter list but differs in return type are not said to be overloaded method

Note: Overloaded method can have different access modifiers.

**Write a program consisting method sum() and demonstrate the concept of method overloading using this method.**

**public** **class** MethodOverLoading

{

**void** sum(**int** a, **int** b)

{

System.***out***.println("Sum is" + (a + b));

}

**void** sum(**float** a, **float** b)

{

System.***out***.println("Sum is" + (a + b));

}

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

{

MethodOverLoading cal=**new** MethodOverLoading();

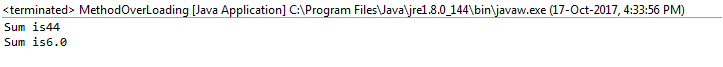
cal.sum (22,22);

cal.sum (2.3f,3.7f);

}

}

Output:



## Program 7: Can methods be overloaded based on the return types

**Can you overload a method with same return type? Explain your answer with proper logic.**

In Java, method overloading is not possible by changing the return type of the method because there may occur ambiguity.

Consider the following example,

**public** **class** MethodOLRT

{

**int** sum(**int** a, **int** b)//Compile Time Error - Duplicate method sum(int, int)

{

**return** a+b;

}

**void** sum(**int** a, **int** b)//Compile Time Error - Duplicate method sum(int, int)

{

System.***out***.println(a+b);

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

MethodOLRT obj=**new** MethodOLRT();

**int** result=obj.sum(10, 10);

}

}

The method can be overloaded only on the following basis:

1. The method name are same (including the case. viz. sum and sum are not the same)

2. The signatures (argument) of the method. Return type cannot be used for the method overloading.

A method cannot be overloaded on the bases of return types because the compiler will check only the name of the method and the arguments. So a method cannot be overloaded on the bases of return types.

## Program 8: Descending Order

Write a program in java using Arrays that sorts the element in descending order.

**public** **class** DescendingOrder

{

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

{

**int** n, temp;

Scanner buf=**new** Scanner(System.***in***);

System.***out***.print("Enter no of Elements: ");

n=buf.nextInt();

**int** a[]=**new** **int**[n];

System.***out***.println("Enter all the elements: ");

**for** (**int** i=0; i<n; i++)

{

a[i]=buf.nextInt();

}

**for** (**int** i=0; i<n; i++)

{

**for** (**int** j=i+1; j<n; j++)

{

**if**(a[i]<a[j])

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

System.***out***.print("Descending Order: ");

**for** (**int** i=0; i<n-1; i++)

{

System.***out***.print(a[i] + ",");

}

System.***out***.print(a[n-1]);

}

}

Output:

