**2.1**

**a.**

class acad{

public static void main(string args[]) {

int a =2 ,b=3,c;

c=a+b;

System.out.println(“Sum =”+c);

}

}

**b.**

class acad{

public static void main(string args[]) {

int a ,b,c;

Scanner s = new Scanner(System.in);

a = s.nextInt();

b= s.nextInt();

c=a+b;

System.out.println(“Sum =”+c);

}

}

**c.**

class acad{

public static void sum(int a,int b){

int c=a+b;

Sysytem.out.println(“First number is:”+ a);

Sysytem.out.println(“Second number is:”+ b);

Sysytem.out.println(“Sum is:”+ c);

}

public static void main(string args[]) {

int a ,b;

Scanner s = new Scanner(System.in);

a = s.nextInt();

b= s.nextInt();

sum(a,b);

}

**d.** public class acad2 {

public static void main(String args[]) {

StringBuffer sb = new StringBuffer(); //initiate buffer to store even values

StringBuffer sb1 = new StringBuffer();//initiate buffer to store odd values

Scanner sc =new Scanner(System.in); //scan input from user

int a = sc.nextInt(); //declare variable and scan value from user

int b =sc.nextInt(); //declare variable and scan value from user

for (int i = a; i <=b; i++) { // run loop within range of input

if(i%2==0) //check if number is even

{ sb.append(i); sb.append(" "); //store in buffer sb }

else { sb1.append(i);

sb1.append(" "); //if odd store in buffer sb1 } }

System.out.println("The even number are"); //print values in buffer sb System.out.println(sb.toString());

System.out.println("The odd number are"); //print values in buffer sb2 System.out.println(sb1.toString()); }

**e.**

public class acad2 {

public static void main(String args[]) {

Scanner sc= new Scanner(System.in); // to scan input from user System.out.println("Input"); //print input

int no = sc.nextInt(); //scan value from user to print multiples System.out.println("Output");

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

System.out.println(no+"x"+i+"="+i\*no); //print 10 multiples of no

}}

**f.**

Method Overloading- If a class has multiple methods having same name but different in parameters, it is known as Method Overloading. it can be done be achieved by either changing data type or changing number of argument. Lets see by exanple

1. Changing data type- here method named sum is used to perform addition of int type as well as string type of inputs. method performs the same action without errors

import java.util.Scanner; public class acad2 { public static void sum(int a , int b) //create method sum sum two int { System.out.println(a+b);

} public static void sum(String s ,String s1) //another method named sum for string { String s4 = s+s1; System.out.println(s4); }

public static void main(String args[]) { Scanner sc= new Scanner(System.in);

int a = sc.nextInt(); //scan input a as int

int b = sc.nextInt(); //scan input b as int

String st = sc.next(); //scan input st as string

String st1 = sc.next(); //scam input st1 as string

sum(a,b);//sum with integer type as input

sum(st,st1);// sum with String type as input

}

} 2.by changing number of arguments-here we passed two arguments to perform then changed it to three, the method run without errors.

import java.util.Scanner; class acad2{  
static int add(int a,int b){return a+b;} //pass two arguments to operate on..case1 static int add(int a,int b,int c){return a+b+c;}//change no. of arguments passed..case1 public static void main(String[] args){  
System.out.println(acad2.add(11,15)); //print sum for case1 System.out.println(acad2.add(11,15,19)); //print sum for case2 }}

**g.**

Yes we can overload method with same return type but the argument list should be diffrent but method overloading is not possible by changing the return type of the method only because of ambiguity. Basically method overloading is ,If a class has multiple methods having same name but different in parameters, it is known as Method Overloading. it can be done be achieved by either changing data type or changing number of argument.

In code below we passed two int and summed their result in method with return type as int , but in another method we passed we two char's and returned them with add by adding. in case1 it gave me addition of two ints and in second it gives addition of ASCII values of char's passed. hence, Here we can see that the method sum has the same return type but argument list is different one gives actual sum and the other gives the ascii value of the character.

import java.util.Scanner;

public class acad2 { public static int sum(int a , int b) //create method sum to sum two int { return a+b; //return sum

}

public static int sum(char c,char t) //method with return type as int but sum char { int n =(int)c; int p=(int) t; return n+p; //return sum;  
}

public static void main(String args[])

{

Scanner sc= new Scanner(System.in);

int a = sc.nextInt(); //scan input from user

int b = sc.nextInt();

System.out.println("output");

System.out.println(sum(a,b));//sum with integer type as input

System.out.println(sum('c','r'));// sum of ASCII values

}

}

**h.**

import java.util.Arrays; import java.util.Collections; import java.util.Scanner;

public class acad2 { public static void main(String args[]) { Scanner sc= new Scanner(System.in);

Integer[] sortArray = new Integer[5]; //scan 5 elements from user

for (int i = 0; i < 5; i++) {

sortArray[i]=sc.nextInt();

}

Arrays.sort(sortArray, Collections.reverseOrder()); //sort array and reverse

System.out.println("Output");

for (int i = 0; i < sortArray.length; i++)

{

System.out.println(sortArray[i]);//print array

}

}

}