Verify a number is Even/Odd

Java

```
import java.util.Scanner;

public class EvenOdd{
public static void main(String[] args){
Scanner in = new Scanner(System.in);
System.out.println("Enter a number which you want to check whether that is even or odd");
int n = in.nextInt();

if(n%2==0){
System.out.println(n+" is an even number.");
}else{
System.out.println(n+" is an odd number.");

}
}
}
```

Swapping Numbers without using 3rd variable

```
import java.util.Scanner;

public class Swapping{
public static void main(String[] args){
Scanner in = new Scanner(System.in);
System.out.println("Enter the 1st number: ");
int x = in.nextInt();
System.out.println("Enter the 2nd number: ");
int y = in.nextInt();

System.out.println("Initial value of x: "+x+" and y: "+y);

x = x+y;
y = x-y;
x = x-y;
```

```
System.out.println("After swapping value of x: "+x+" and y: "+y);
}
}
```

Factorial of a number

Java

```
import java.util.Scanner;

public class Factorial{
public static void main(String[] args){
Scanner in = new Scanner(System.in);
System.out.println("Enter the number whose factorial you want: ");
int n = in.nextInt();
int f =1;
for(int i=n; i&gt;0; i--){
f = f*i;
}
System.out.println("Factorial of "+n+" is "+f);
}
}
```

How to get the prime numbers between a given range.

Java

```
package javaTutorial;

import java.util.ArrayList;
import java.util.Scanner;

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

```
System.out.println("Enter a number from which you want prime number: ");
int p1 = in.nextInt();
System.out.println("Enter one more number till which you want prime number: ");
int p2 = in.nextInt();
ArrayList&lt;Integer&gt; prime = new ArrayList&lt;Integer&gt;();
int i=2;
for(int p=p1; p&lt;=p2; p++){
i=2;
for(; i&lt;10; i++){
if(p%i==0 &amp;&amp; p!=i){
break;
}
}
if(i==10){
prime.add(p);
}
}
System.out.println("Prime numbers between "+p1+" and "+p2+" are: ");
for(int j=0; j&lt;prime.size(); j++){
System.out.print(prime.get(j).toString()+", ");
}
}
}
```

Check a number is prime or not.

Note- A number is prime if it is not divisible by any other number except itself.

Java

```
import java.util.Scanner;

public class PrimeNumber{
public static void main(String[] args){
Scanner in = new Scanner(System.in);
System.out.println("Enter a number greater than 2 which you want to check whether that number is prime or not: ");
int p = in.nextInt();
int i=2;
for(; i&lt;10; i++){
```

```
if(p%i==0 &amp;amp;&amp;amp; p!=i){
System.out.println("Entered number "+p+" is not a prime number.");
break;
}
}
if(i==10){
System.out.println("Entered number "+p+" is a prime number.");
}
}
}
```

Check if a number is Armstrong or not.

Note- A number is armstrong if the sum of the cubes of digit of number is equal to the number.
ex- 407 = 4*4*4 + 0*0*0 + 7*7*7
Java

```
import java.util.Scanner;

public class ArmstrongNum{
public static void main(String[] args){
Scanner in = new Scanner(System.in);
System.out.println("Enter a number which you want to check whether that is armstrong or not: ");
int n = in.nextInt();
int a = n, r=0, s=0;

while(a!=0){
r = a%10;
a = a/10;
s = s + r*r*r;
}
if(s==n){
System.out.println("Number "+n+" is an armstrong number.");
}else{
System.out.println("Number "+n+" is not an armstrong number.");
}
}
}
```

Floyd Triangle

Note- Floyd Triangle is like

1

2 3

4 5 6

7 8 9 10

————

Code-

Java

```
import java.util.Scanner;

public class FloydTriangle{
public static void main(String[] args){
Scanner in = new Scanner(System.in);
System.out.println("Enter the number of rows which you want in your Floyd Triangle: ");
int r = in.nextInt();
int n=0;
for(int i=0; i&lt;r; i++){
for(int j=0; j&lt;=i; j++){
System.out.print(++n+" ");
}
System.out.println();
}
}
}
```

Palindrome of String or reverse a String.

Java

```
import java.util.Scanner;

public class PalindromeString{
public static void main(String[] args){
Scanner in = new Scanner(System.in);
System.out.println("Enter the string which you want to check whether that is palindrome or not: ");
```

```
String s = in.next();
String r = "";
for(int i=s.length()-1; i&gt;=0; i--){
r = r+s.charAt(i);
}
System.out.println("Reverse of entered string "+s+" is "+r);
if(r.equals(s)){
System.out.println("String "+s+" is palindrome.");
}else{
System.out.println("String "+s+" is not palindrome.");
}
}
}
```

Binary Search

```
import java.util.Arrays;
import java.util.Scanner;

public class BinarySearch{
public static void main(String[] args){
Scanner in = new Scanner(System.in);
System.out.println("Enter the size of the array which should be greater than zero else it will throw InputMismatchException :
");
int size = in.nextInt();
int[] array = new int[size];
System.out.println("Enter the elements of the array: ");
for(int i=0; i&lt;size; i++){
array[i] = in.nextInt();
}
System.out.println("Enter the search element: ");
int s = in.nextInt();

Arrays.sort(array); //binary search will work on sorted array only so sort first
int first, last, middle;
first=0;
last = size-1;
middle = (first+last)/2;
int i=0;
```

```
for(; i&lt;size; i++){
if(s&gt;array[middle]){
first = middle+1;
}else if(s&lt;array[middle]){
last = middle-1;
}else{
printArray(array);
System.out.println("Element "+s+" found in the array.");
break;
}
middle= (first+last)/2;
}
if(i==size){
printArray(array);
System.out.println("Element "+s+" is not found in the array");
}
}
public static void printArray(int[] a){
System.out.println("Array of elements: ");
System.out.print("{");
for(int i:a){
System.out.print(i+",");
}
System.out.println("}");
}
}
```

Bubble Sort

Java

```
package SeleniumMakeItEasy;

import org.apache.commons.lang3.ArrayUtils;

public class BubbleSort{
public static void main(String[] args){
int[] a = {2,3,2,5,3,3,6,1,2,5};
int l = a.length;

for(int i=0;i&lt;l; i++){
```

```
for(int j=0; j&lt;l-1; j++){
if(a[j]&gt;a[j+1]){
a[j] = a[j] + a[j+1];
a[j+1] = a[j] - a[j+1];
a[j] = a[j] - a[j+1];
}else if(a[j]==a[j+1]){
a = ArrayUtils.remove(a,j);
l = a.length;
}
}
}
for(int s: a){
System.out.println(s);
}

}
}
```

Reverse number program in java

Posted by: instance of java Posted date: **18:31** / comment : 0

```
1.  package com.instaceofjavaforus;
    public class Reversenum {
     public static void main(String[] args) {

    int rev=0;
    int num=1234;
    while(num>0){
     int rem=num%10;
     rev=rem+(rev*10);
     num=num/10;

    }
    }
    }
```

Program to print prime numbers in java

Posted by: instance of java Posted date: **18:18** / comment : 0

```
package com.instaceofjava;

public class primenumbers {

public static void main(String[] args) {

// TODO Auto-generated method stub

int num=50;

int count=0;

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

count=0;

for(int j=2;j<=i/2;j++){

if(i%j==0){

count++;
break;
}

}

if(count==0){

System.out.println(i);

}

}

}

}
```

Reverse a string without using string function in java

Posted by: instance of java Posted date: **18:22** / comment : 0

```
package com.instaceofjava;

public class ReverseString {

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

```
6.
7.  String str="Hello world";
8.  String revstring="";
9.
10.for(int i=str.length()-1;i>=0;--i){
11.revstring +=str.charAt(i);
12.}
13.
14.System.out.println(revstring);
15.}
16.}
```

Check whether string is palindrome or not

Posted by: instance of java Posted date: **18:26** / comment : 1

Solution #1:

```
1.  package com.instaceofjava;
2.
3.  public class PalindromeDemo{
4.
5.  public static void main(String[] args) {
6.
7.  String str="MADAM";
8.  String revstring="";
9.
10.for(int i=str.length()-1;i>=0;--i){
11.revstring +=str.charAt(i);
12.}
13.
14.System.out.println(revstring);
15.
16.if(revstring.equalsIgnoreCase(str)){
17.System.out.println("The string is Palindrome");
18.}
19.else{
20.System.out.println("Not Palindrome");
21.}
22.
23.}
24.}
```

Fibonacci series without using recursion in java

Posted by: instance of java Posted date: **18:34** / comment : 0

```
1. package com.instaceofjava;

   public class Fibanaciwithoutrecursive {

   public static void main(String[] args) {

   int n1=0;

   int n2=1;

   System.out.println(n1);

   System.out.println(n2);

   for(int i=0;i<=100;i++){

   int sum=n1+n2;

   if(sum<=100){

   n1=n2;

   n2=sum;

   System.out.println(sum);

   }
   }
   }
```

Fibonacci series using recursion in java

Posted by: instance of java Posted date: **18:40** / comment : 0

```
1. package com.instaceofjavaforus;
     public class FibanacciRecursive {
    public void fibanci(int n1,int n2){
     int sum=0;
     if(n1==0){
     System.out.println(n1+"\n"+n2);
     }
     sum=n1+n2;
```

```
      if(sum<=100){
       System.out.println(sum);
       n1=n2;
       n2=sum;

       fibanci(n1,n2);

      }
     }

     public static void main(String[] args) {

         FibanacciRecursive fb=new FibanacciRecursive();
         fb.fibanci(0,1);
     }

    }
```

Sort the string using String Methods?

Posted by: instance of java Posted date: **18:44** / comment : 0

```
1.  package com.instanceofjava;
2.
3.  public class SortString {
4.
5.  public static void main(String[] args) {
6.
7.     String original = "edcba";
8.
9.    char[] chars = original.toCharArray();
10.
11.  Arrays.sort(chars);
12.
13.  String sorted = new String(chars);
14.  System.out.println(sorted);
15.}
16.
17.}
```

Sorting string without using string Methods?

Posted by: instance of java Posted date: **18:46** / comment : 0

```
1.  package com.Instanceofjava;
```

```
2.  public class SortString {
3.
4.  public static void main(String[] args) {
5.
6.  String original = "edcba";
7.  int j=0;
8.  char temp=0;
9.
10.   char[] chars = original.toCharArray();
11.
12.   for (int i = 0; i <chars.length; i++) {
13.
14.       for ( j = 0; j < chars.length; j++) {
15.
16.        if(chars[j]>chars[i]){
17.             temp=chars[i];
18.            chars[i]=chars[j];
19.            chars[j]=temp;
20.        }
21.
22.    }
23.
24. }
25.
26. for(int k=0;k<chars.length;k++){
27. System.out.println(chars[k]);
28. }
29.
30. }
31.
```

}

How to find largest element in an array with index and value using array?

Posted by: instance of java Posted date: **18:58** / comment : 0

```
1.  package com.instanceofjava;
2.  public class Array {
3.
4.  public static void main(String[] args) {
5.
6.  int arr[]={1,120,56,78,87};
7.  int largest=arr[0];
8.  int smallest=arr[0];
9.  int small=0;
10. int index=0;
11.
12. for(int i=1;i<arr.length;i++){
```

```
13.
14.if(arr[i]>largest){
15.
16.largest=arr[i];
17.index=i;
18.
19.}
20.else if(largest>arr[i]){
21.
22.smallest=arr[i];
23.small=i;
24.
25.}
26.}
27.
28.System.out.println(largest);
29.System.out.println(index);
30.System.out.println(smallest);
31.System.out.println(small);
32.
33.}
34.
35.}
```

what is the difference between method overloading and method overriding?

Posted by: instance of java Posted date: **12:16** / comment : 0

- **Method overloading** is nothing but defining multiple fuctionalities with same name with
  different parameters is known as method overloading
- while compile time itself we know which method is execute
- Overloading gives better performance compared to overriding. The reason is that the binding of overridden methods is being done at runtime.
- Argument list should be different while doing method overloading.
- Polymorphism not applies to overloading. Return type of overloaded methods should be same.
  static binding is used for overloaded methods

```
public class A{
public void show(int a){
S.o.p("indhu");
}
public void show(int a,int b){
S.o.p("sindhu");
```

```
}

public void show(float a){
S.o.p("lavanya");
}
public static void main(String args[]){

A a=new A();
a.show(10);
a.show(1,2);
a.show(1.2);
}
}
```

- Method overriding is nothing but defining multiple finctionalities with same name with same definition in super class and sub class is known as Method overriding.
  While Run time only now which method is Executed.
- Overriding gives slower performance compared to overloading.The reason is that the binding of overridden methods is being done at run time.
- Argument list should be same while doing method overriding.
  Polymorphism applies to overriding.
- In method overriding the return type of overriding method can be different from overridden
  method.
- Dynamic binding used for method overriding methods

```
public class A{
public void display(){
S.o.p("Indhu");
}
}
Public class B extends A{

public void display(){
S.o.p("Sindhu");
}
public sttaic void main(String args[]){
A a =new B();
a.display();
}
}
```

Sort integer array using Bubble Sort in java

Posted by: instance of java Posted date: **19:02** / comment : 0

```
1.  package com.instaceofjava;
2.
3.  public class Bubblesort {
4.
5.  public static void main(String[] args) {
6.
7.  int a[]={23,5,6,66,1};
8.
9.  int n=a.length;
10.
11.         int temp=0;
12.
13.for(int i=0;i<n;i++){
14.
15.for(int j=1;j<(n-i);j++){
16.
17.if(a[j-1]>a[j]){
18.    temp=a[j];
19.   a[j]=a[j-1];
20.   a[j-1]=temp;
21.
22.   }
23.
24.}
25.
26.}
27.
28.for(int k=0;k<n;k++){
29.System.out.println(a[k]);
30.}
31.
32.}
33.
34.}
```

Object cloning in java example

Posted by: instance of java Posted date: **19:07** / comment : 0

```
1.  package com.instanceofjava;
2.
3.  public class Employee implements Cloneable {
4.
5.          int a=0;
```

```
6.          String name="";
7.          Employee (int a,String name){
8.          this.a=a;
9.          this.name=name;
10.}
11.
12.public Employee clone() throws CloneNotSupportedException{
13.
14.return (Employee ) super.clone();
15.
16.}
17.
18.public static void main(String[] args) {
19.
20.          Employee e=new Employee (2,"Indhu");
21.            System.out.println(e.name);
22.
23.try {
24.
25. Employee b=e.clone();
26. System.out.println(b.name);
27.
28.}
29. catch (CloneNotSupportedException e1) {
30.
31.e1.printStackTrace();
32.}
33.}
34.
35.}
```

Method overriding example program

Posted by: instance of java Posted date: **19:09** / comment : 1

```
1.  package com.oops;
2.
3.  class A
4.  {
5.
6.  void msg(){
7.
8.   System.out.println("Hello");
9.
10.}
11.
```

```
12.}
```

write a program to create singleton class?

Posted by: instance of java Posted date: **15:50** / comment : 0

```
package com.instanceofjava;

public class Singleton {

 static Singleton obj;
 private  Singleton(){
 }

 public static Singleton getInstance(){
  if(obj!=null){
   return  obj;
  }
  else{
   obj=new Singleton();
  return obj;
  }
 }

 public static void main(String[] args) {

  Singleton obj=Singleton.getInstance();
  Singleton obj1=Singleton.getInstance();

  if(obj==obj1){
  System.out.println("indhu");
  }
  else{
   System.out.println("Sindhu");
  }
              System.out.println(obj==obj1);

 }
}
```

Print numbers in pyramid shape

Posted by: instance of java Posted date: **14:12** / comment : 0

```
1.  package com.instanceofjava;
2.
3.  public class Pyramidshape {
4.
```

```
5.   public static void main(String[] args) {
6.
7.    int num=10;
8.
9.    for (int i = 1; i < num; i++) {
10.
11.    for (int j = 1; j<=num-i;j++) {
12.
13.     System.out.print(" ");
14.    }
15.
16.   for (int k = 1; k <= i; k++) {
17.    System.out.print(""+k+" ");
18.   }
19.
20.   for (int l =i-1; l >0; l--) {
21.    System.out.print(""+l+" ");
22.   }
23.
24.    System.out.println();
25.   }
26.
27.}
```

Check armstrong number or not

Posted by: instance of java Posted date: **20:09** / comment : 0

```
1.  package com.instanceofjavaTutorial;
2.  import java.util.Scanner;
3.
4.  public class ArmstrongNumber{
5.
6.  public static void main(String args[])
7.   {
8.
9.          int n, sum = 0, temp, r;
10.
11.         Scanner in = new Scanner(System.in);
12.
13.         System.out.println("Enter a number to check if it is an Armstrong number
    or not");
14.         n = in.nextInt();
15.
16.         temp = n;
17.         while( temp != 0 )
18.         {
```

```
19.
20.           r = temp%10;
21.           sum = sum + r*r*r;
22.           temp = temp/10;
23.
24.         }
25.
26.         if ( n == sum )
27.           System.out.println(n+"is an Armstrong number.");
28.         else
29.           System.out.println(n+"  is not an Armstrong number.");
30.
31.       }
32.}
```

Common class

```
package com.instanceofjavaforus;

public class Common {

    int x;

    boolean flag=true;
    //if flag is true producer thread has to produce
    // if flag is false consumer thread has to produce

    synchronized public void produce(int i){

        if(flag){

            x=i;
            System.out.println("producer thread has produced "+i);
            flag=false;
            notify();
            try {
                wait();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
     }


    synchronized public int consume(){
        if(flag){
            try {
                wait();
            } catch (InterruptedException e) {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
    flag=true;
    notify();
    return x;
  }

}

package com.instanceofjavaforus;

public class ProducerThread extends Thread {
  Common c;

  ProducerThread(Common c){
    this.c=c;
  }

  public void run(){
    int i=0;
    while(true){

      i++;
      c.produce(i);


      try {
        Thread.sleep(600);
      } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
    }
  }
}
```

Remove duplicates from an array java

Posted by: instance of java Posted date: **11:09** / comment : 0

Removing duplicate elements form an array using collections:

```
1.  package com.instanceofjavaTutorial;
```

```
2.
3.  import java.util.Arrays;
4.  import java.util.HashSet;
5.  import java.util.List;
6.  import java.util.Set;
7.
8.  public class RemoveDupArray {
9.
10.   public static void main(String[] args) {
11.
12.  // A string array with duplicate values
13.    String[] data = { "E", "C", "B", "E", "A", "B", "E", "D", "B", "A" };
14.
15.System.out.println("Original array        : " + Arrays.toString(data));
16.
17. List<String> list = Arrays.asList(data);
18. Set<String> set = new HashSet<String>(list);
19.
20.   System.out.print("After removing duplicates: ");
21.   String[] resarray= new String[set.size()];
22.    set.toArray(resarray);
23.
24. for (String ele: resarray) {
25.
26. System.out.print(ele + ", ");
27.
28. }
29.
30.}
31.
32.}
```

Convert Byte Array to String

Posted by: instance of java Posted date: **09:01** / comment : 2

- To convert byte array to string we have a constructor in String class which is taking byte array as an argument.
- So just we need to pass byte array object to string as argument while creating String class object.

```
1.    package com.instanceofjavaforus;
2.
3.
4.  public class ByteArrayToString {
5.      /*
6.      * This method converts a byte array to a String object.
7.        */
```

```
8.
9.      public static void convertByteArrayToString() {
10.
11.         byte[] byteArray = new byte[] {78,73, 67,69};
12.         String value = new String(byteArray);
13.        System.out.println(value);
14.
15.     }
16.
17.     public static void main(String[] args) {
18.         ByteArrayToString.convertByteArrayToString();
19.     }
20.
21.}
```

Print 1 to 10 without using loop in java?

Posted by: instance of java Posted date: **15:01** / comment : 0

- Basically to display numbers from 1 to 10 or a series we will use for , while or do while loop
- So here is the programs to do same thing without using loop.
- This is possible in two ways
- First one is to display by printing all those things using system.out.println.
- second one is by using recursive method call lets see these two programs

Solution #1:

```
1.  package com.instanceofjavaTutorial;
2.
3.  class Demo{
4.
5.  public static void main(String args[]) {
6.

7.     System.out.println(1);
8.     System.out.println(2);
9.     System.out.println(3);
10.    System.out.println(4);
11.    System.out.println(5);
12.    System.out.println(6);
13.    System.out.println(7);
14.    System.out.println(8);
15.    System.out.println(9);
16.    System.out.println(10);
17.
18.}
19.
```

```
20.}
```

add two matrices

Posted by: instance of java Posted date: **20:48** / comment : 0

```
1.  package com.instanceofjava;
2.
3.  import java.util.Scanner;
4.
5.  class Add2Matrix
6.  {
7.
8.     public static void main(String args[])
9.     {
10.
11.       int rows, cols, c, d;
12.
13.       Scanner in = new Scanner(System.in);
14.
15.       System.out.println("Please Enter number of rows and columns");
16.
17.       rows = in.nextInt();
18.       cols  = in.nextInt();
19.
20.       int first[][] = new int[rows][cols];
21.       int second[][] = new int[rows][cols];
22.       int sum[][] = new int[rows][cols];
23.
24.       System.out.println("Please Enter elements of first matrix");
25.
26.       for (  c = 0 ; c < rows ; c++ )
27.          for ( d = 0 ; d < cols ; d++ )
28.             first[c][d] = in.nextInt();
29.
30.       System.out.println("Please Enter elements of second matrix");
31.
32.       for ( c = 0 ; c < rows ; c++ )
33.          for ( d = 0 ; d < cols ; d++ )
34.             second[c][d] = in.nextInt();
35.
36.       for ( c = 0 ; c < rows ; c++ )
37.          for ( d = 0 ; d < cols ; d++ )
38.             sum[c][d] = first[c][d] + second[c][d];  //replace '+' with '-' to subtract
    matrices
39.
40.       System.out.println("Sum of entered matrices:-");
41.
42.       for ( c = 0 ; c < rows ; c++ )
```

```
43.      {
44.        for ( d = 0 ; d < cols ; d++ )
45.          System.out.print(sum[c][d]+"\t");
46.        System.out.println();
47.      }
48.   }
49.
50. }
```

Multiply two matrices

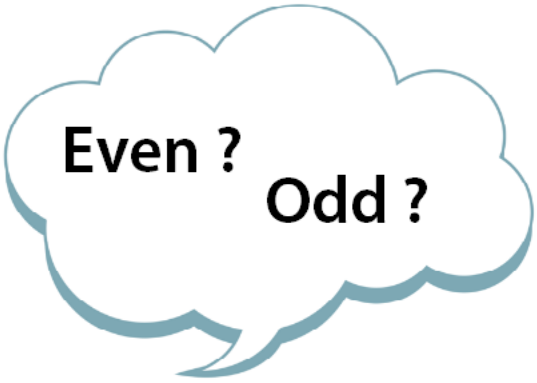
Posted by: instance of java Posted date: **21:21** / comment : 0

```
1.  package com.instanceofjava;
2.
3.  import java.util.Scanner;
4.
5.  class Multiply2Matrices{
6.
7.     public static void main(String args[])
8.     {
9.
10.      int m, n, p, q, sum = 0, c, d, k;
11.
12.      Scanner in = new Scanner(System.in);
13.
14.      System.out.println("Enter the number of rows and columns of first matrix");
15.
16.      m = in.nextInt();
17.      n = in.nextInt();
18.
19.      int first[][] = new int[m][n];
20.
21.     System.out.println("Enter the elements of first matrix");
22.
23.      for ( c = 0 ; c < m ; c++ )
24.        for ( d = 0 ; d < n ; d++ )
25.          first[c][d] = in.nextInt();
26.
27.      System.out.println("Enter the number of rows and columns of second
    matrix");
28.
29.      p = in.nextInt();
30.      q = in.nextInt();
31.
32.     if ( n != p )
33.        System.out.println("Matrices with entered orders can't be multiplied with
    each other.");
34.      else
```

```
    {
      int second[][] = new int[p][q];
      int multiply[][] = new int[m][q];
      System.out.println("Enter the elements of second matrix");

      for ( c = 0 ; c < p ; c++ )
        for ( d = 0 ; d < q ; d++ )
          second[c][d] = in.nextInt();

      for ( c = 0 ; c < m ; c++ )
      {
        for ( d = 0 ; d < q ; d++ )
        {
          for ( k = 0 ; k < p ; k++ )
          {
            sum = sum + first[c][k]*second[k][d];
          }
          multiply[c][d] = sum;
          sum = 0;
        }
      }

      System.out.println("Multiplication of entered matrices:-");

      for ( c = 0 ; c < m ; c++ )
      {
        for ( d = 0 ; d < q ; d++ )
          System.out.print(multiply[c][d]+"\t");
        System.out.print("\n");
      }

    }

  }
}
```

Program Check even or odd without using modulus and division operators

Posted by: instance of java Posted date: **14:49** / comment : 3



- Checking the number even or odd program is very easy. Anybody can solve this but there is a condition we need to see.
- When we get this question in interview "write a program to check given number is even or odd" always continues with "without using modulus and division operators".
- Before going to actual program lets see how to check a number is even or odd by using modulus and division operators.

Program to check number is even or odd by using modulus "%" operator

```
1.  package instanceofjava;
2.  import java.util.Scanner;
3.
4.  public class EvenorOdd {
5.
6.  public static void main(String []args )    {
7.
8.      int number;
9.      Scanner in= new Scanner(System.in);
10.
```

```
11.    System.out.println("Enter a number to check even or odd");
12.    number=in.nextInt();
13.
14.    if((number % 2)==0){
15.        System.out.println(+number+" is Even number");
16.    }else{
17.        System.out.println(+number+" is Odd Number");
18.    }
19.
20.}
21.}
```

How to Add elements to hash map and Display?

Posted by: instance of java Posted date: **18:54** / comment : 0

```
1.  package com.instaceofjava;
2.
3.  import java.util.HashMap;
4.  import java.util.Iterator;
5.  import java.util.Map;
6.  import java.util.Map.Entry;
7.
8.  public class Mapiterator {
9.
10. public static void main(String[] args) {
11.
12. HashMap map=new HashMap();
13.
14. map.put(1, "indhu");
15. map.put("d", "sindhu");
16. map.put("3", "swathi");
17.
18. if(!map.isEmpty()){
19.
20. Iterator it=map.entrySet().iterator();
21.
22. while(it.hasNext()){
23.
24. Map.Entry obj=(Entry) it.next();
25. System.out.println(obj.getValue());
26.
27. }
28.
29. }
30.
```

```
}

}
```

Sort ArrayList in descending order

Posted by: instance of java Posted date: **10:49** / comment : 0

Descending order:

```
1.  package com.instanceofjavaforus;
2.  import java.util.ArrayList;
3.   import java.util.Collections;
4.  import java.util.Comparator;
5.
6.  public class SortArrayListDesc {
7.
8.      public static void main(String[] args) {
9.
10.         //create an ArrayList object
11.         ArrayList arrayList = new ArrayList();
12.
13.         //Add elements to Arraylist
14.         arrayList.add(1);
15.         arrayList.add(2);
16.         arrayList.add(3);
17.        arrayList.add(4);
18.         arrayList.add(5);
19.         arrayList.add(6);
20.
21.         /*
22.        Use static Comparator reverseOrder() method of Collections
23.      utility class to get comparator object
24.        */
25.
26.       Comparator comparator = Collections.reverseOrder();
27.
28.       System.out.println("Before sorting  : "  + arrayList);
29.
30.
31.       /*
32.          use
33.          static void sort(List list, Comparator com) method of Collections class.
34.        */
35.
36.         Collections.sort(arrayList,comparator);
37.         System.out.println("After sorting  : + arrayList);
38.
```

```
39.
40.       }
41.    }
```

Sort object using comparator

Posted by: instance of java Posted date: **19:18** / comment : 0

```
1.  package com.instanceofjava;
2.
3.  public class Employee
4.  {
5.
6.   int id;
7.   String name;
8.
9.   public Employee(int id, String name) {
10.
11.   this.id=id;
12.   this.name=name;
13.
14. }
15.
16.public int getId() {
17.
18.return id;
19.
20.}
21.
22. public void setId(int id) {
23.   this.id = id;
24. }
25.
26. public String getName() {
27.   return name;
28. }
29.
30.public void setName(String name) {
31.
32.   this.name = name;
33.
34.}
35.
36.}
```

Count the number of occurrences of a character in a String?

Posted by: instance of java Posted date: **18:52** / comment : 0

```
1.  package com.instaceofjava;
2.
3.  public class StringCount {
4.
5.  public static void main(String[] args)
6.  {
7.
8.  String str = "abcdcabcdacbdadbca";
9.
10.     String findStr = "b";
11.     int lastIndex = 0;
12.     int count = 0;
13.
14.     while (lastIndex != -1) {
15.
16.      lastIndex = str.indexOf(findStr, lastIndex);
17.
18.      if (lastIndex != -1) {
19.       count++;
20.       lastIndex += findStr.length();
21.
22.      }
23.     }
24.     System.out.println(count);
25.}
26.
27.}
```

Can we overload static methods in java

Posted by: instance of java Posted date: **19:20** / comment : 0

- Yes. We can overload static methods in java.
- Method overriding is not possible but method overloading is possible for static methods.
- Before that lets see about method overloading in java.

Method overloading:

- Defining multiple methods with same name and with different arguments is known as method overloading.
- Multiple methods with same name and different arguments so compile time itself we can tell which method is going to get executed based on method call.
- Method overloading also known as compile time polymorphism.

Static methods - method overloading

- Its always possibles static method overloading.
- Defining multiple static methods with same name and different arguments will possible.
- By this we can define multiple main methods in our class with different arguments but only default main method will be called by the JVM remaining methods we need to call explicitly.
- Lets see an example java program which explains static method overloading.

```
1.  class StaticMethodOverloading{
2.
3.  public static void staticMethod(){
4.
5.  System.out.println("staticMethod(): Zero arguments");
6.
7.  }
8.
9.  public static void staticMethod(int a){
10.
11.System.out.println("staticMethod(int a): one argument");
12.
13.}
14.
15.public static void staticMethod(String str, int x){
16.
17.System.out.println("staticMethod(String str, int x): two arguments");
18.
19.}
20.
21.public static void main(String []args){
22.
23.  StaticMethodOverloading.staticMethod();
24.  StaticMethodOverloading.staticMethod(12);
25.  StaticMethodOverloading.staticMethod("Static method overloading",10);
26.
27.}
28.}
```

Can we override private method in java

Posted by: instance of java Posted date: **17:20** / comment : 0

- We can not override private methods in java.
- The basic inheritance principle is when we are extending a class we can access all non private members of a class so private members are private to that class only we can not access anywhere outside the class if we declare anything as private.
- Know more information about access specifiers here

```
1.  Class A{
2.
3.    int a;
4.   private int b;
5.
6.  public void print(){
7.
8.  System.out.println("print method of super class A");
9.
10.}
11.
12.private void add(){
13.
14.System.out.println("add method of super class A");
15.
16.}
17.}
18.Class B extends A{
19.
20.public void show(){
21.
22. System.out.println("show method of sub class B");
23.
24.}
25.public static void main(String[] args){
26.
27.B obj= new B();
28.obj.a=30;
29.
30.obj.print();
31.obj.show();
32.
33. System.out.println("a="obj.a);
34.//obj.b=20;  compile time error. The field Super.a is not visible
35.//obj.add(); compile time error : The method show() from the type Super is not
    visible
36.
37.}
```

```
38.}
```

Can we call sub class methods using super class object

Posted by: instance of java Posted date: **16:23** / comment : 0

- Common coding interview question everybody facing in interviews is can we call sub class method using super class object? or can we call child class method using parent class? or can a super class call a subclass' method.
- The answer is No. but still we can say yes. so we need to what are all those scenarios of calling sub class method from super class and lets discuss about which is the actual possible case to call sub class method using super class object.
- Before that let me explain what is inheritance and how the super and sub classes related each other.

Inheritance:

- Getting the properties from one class object to another class object is known as inheritance.
- Two types of inheritance
- Getting the properties from one class object to another with level wise and with some priorities is known as multilevel inheritance.
- Getting the properties from one class object to another class object with same priority is known as multiple inheritance
- Java does not supports multiple inheritance

Read this:

Why Java does not supports multiple inheritance

Creating super class object and calling methods

```
1.  //Multilevel inheritance program
2.  Class A{
3.
4.  int a;
5.
6.  public void print(){
7.
8.  System.out.println("print method of super class A");
9.
10.}
```

```
11.
12.}
13.Class B extends A{
14.
15.public void show(){
16.
17. System.out.println("show method of sub class B");
18.
19.}
20.public static void main(String[] args){
21.
22.A obj= new A();
23.obj.a=10;
24.
25.obj.print();
26. //obj.show(); // compile time error
27.
28. System.out.println("a="obj.a);
29.
30.}
31.}
```

return type in java example

Posted by: instance of java Posted date: **18:02** / comment : 0

Return type in java:

- Basically return type is used in java methods.
- Method signature includes this return type.
- public int show(){ // }
- we will use methods to do a particular task after completion of task if we want to return something to the calling place these return types will be used.
- Based on the type of data to be returned will mention it as int , char , float double etc as return type in method signature and return statement should be the last statement of the method body.

Type of declaration of methods based on return type and arguments:

1.Method with out return type and without arguments.

```
1.  package com.instanceofjava;
2.  class sample{
3.
4.  public void add(){
5.
6.  int a=40;
7.  int b=50;
8.  int c=a+b;
9.  System.out.println(c);
10.
11.}
12.
13.public static void main(String args[]) // ->method prototype.
14.{
15.
16.sample obj= new sample();
17.obj.add();
18.
19.
20. }
21.}
```

Can we override static methods in java

Posted by: instance of java Posted date: **16:58** / comment : 0

- Exact answer is NO. We can not override static methods.
- Before discussing this topic lets see what is static in java.

Static methods in java:

- Static means class level if we declare any data or method as static then those data(variables) and methods belongs to that class at class level.
- Static variables and static methods belongs to class level.
- Static methods are not the part of objects state . Belongs to class
- Without using object we can call these static methods.

here the detailed explanation on static methods

Instance methods in java:

- Instance methods will be called at run time.
- Instance variables and instance methods are object level. variables values may change from one object to another where as static variable values are class level so if we access by using any object and changes that values will effect to all objects means its class level variable.

- Lets see about overriding in java.

Method overriding in java:

- Defining the super class method in sub class with same signature.
- Even though in inheritance all properties of supers class can access in sub class we have an option of overriding super class methods in sub class known as method overriding.
- So by this every-time sub-most object method will be called.

Instance methods - Method Overriding

```
1. class SuperClassDemo{
2.
3. public void instanceMethod(){
4.
5. System.out.println("SuperClassDemo instanceMethodcalled");
6.
7. }
8.
9. }
```

PROGRAM: WRITE A PROGRAM TO FIND MAXIMUM REPEATED WORDS FROM A FILE.

Description:

Write a program to read words from a file. Count the repeated or duplicated words. Sort it by maximum repeated or duplicated word count.

Code:

?

```
package com.java2novice.algos;


import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.StringTokenizer;
import java.util.Map.Entry;

public class MaxDuplicateWordCount {

    public Map<String, Integer> getWordCount(String fileName){

        FileInputStream fis = null;
        DataInputStream dis = null;
        BufferedReader br = null;
        Map<String, Integer> wordMap = new HashMap<String, Integer>();
        try {
            fis = new FileInputStream(fileName);
            dis = new DataInputStream(fis);
            br = new BufferedReader(new InputStreamReader(dis));
            String line = null;
            while((line = br.readLine()) != null){
                StringTokenizer st = new StringTokenizer(line, " ");
                while(st.hasMoreTokens()){
                    String tmp = st.nextToken().toLowerCase();
                    if(wordMap.containsKey(tmp)){
```

```
                        wordMap.put(tmp, wordMap.get(tmp)+1);
                    } else {
                        wordMap.put(tmp, 1);
                    }
                }
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally{
            try{if(br != null) br.close();}catch(Exception ex){}
        }
        return wordMap;
    }

    public List<Entry<String, Integer>> sortByValue(Map<String, Integer> wordMap){

        Set<Entry<String, Integer>> set = wordMap.entrySet();
        List<Entry<String, Integer>> list = new ArrayList<Entry<String, Integer>>(set
        Collections.sort( list, new Comparator<Map.Entry<String, Integer>>()
        {
            public int compare( Map.Entry<String, Integer> o1, Map.Entry<String, Integ
            {
                return (o2.getValue()).compareTo( o1.getValue() );
            }
        } );
        return list;
    }

    public static void main(String a[]){
```

```
            MaxDuplicateWordCount mdc = new MaxDuplicateWordCount();
            Map<String, Integer> wordMap = mdc.getWordCount("C:/MyTestFile.txt");
            List<Entry<String, Integer>> list = mdc.sortByValue(wordMap);
            for(Map.Entry<String, Integer> entry:list){
                System.out.println(entry.getKey()+" ==== "+entry.getValue());
            }
        }
    }
```

PROGRAM: WRITE A PROGRAM TO FIND OUT DUPLICATE CHARACTERS IN A STRING.

Description:

Write a program to find out duplicate or repeated characters in a string, and calculate the count of repeatation.

Code:

?

```
package com.java2novice.algos;

import java.util.HashMap;
import java.util.Map;
import java.util.Set;

public class DuplicateCharsInString {

    public void findDuplicateChars(String str){

        Map<Character, Integer> dupMap = new HashMap<Character, Integer>();
        char[] chrs = str.toCharArray();
```

```
            for(Character ch:chrs){
                if(dupMap.containsKey(ch)){
                    dupMap.put(ch, dupMap.get(ch)+1);
                } else {
                    dupMap.put(ch, 1);
                }
            }
            Set<Character> keys = dupMap.keySet();
            for(Character ch:keys){
                if(dupMap.get(ch) > 1){
                    System.out.println(ch+"--->"+dupMap.get(ch));
                }
            }
        }

        public static void main(String a[]){
            DuplicateCharsInString dcs = new DuplicateCharsInString();
            dcs.findDuplicateChars("Java2Novice");
        }
    }
```

PROGRAM: WRITE A PROGRAM TO FIND TOP TWO MAXIMUM NUMBERS IN A ARRAY.

Description:

Write a program to find top two maximum numbers in the given array. You should not use any sorting functions. You should iterate the array only once. You should not use any kind of collections in java.

Code:

```
?
    package com.java2novice.algos;

    public class TwoMaxNumbers {

        public void printTwoMaxNumbers(int[] nums){
            int maxOne = 0;
            int maxTwo = 0;
            for(int n:nums){
                if(maxOne < n){
                    maxTwo = maxOne;
                    maxOne =n;
                } else if(maxTwo < n){
                    maxTwo = n;
                }
            }
            System.out.println("First Max Number: "+maxOne);
            System.out.println("Second Max Number: "+maxTwo);
        }

        public static void main(String a[]){
            int num[] = {5,34,78,2,45,1,99,23};
            TwoMaxNumbers tmn = new TwoMaxNumbers();
            tmn.printTwoMaxNumbers(num);
        }
    }
```

PROGRAM: WRITE A PROGRAM TO FIND PERFECT NUMBER OR NOT.

Description:

A perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself. Equivalently, a perfect number is a number that is half the sum of all of its positive divisors.
The first perfect number is 6, because 1, 2 and 3 are its proper positive divisors, and 1 + 2 + 3 = 6. Equivalently, the number 6 is equal to half the sum of all its positive divisors:
(1 + 2 + 3 + 6) / 2 = 6.

Code:

?

```
package com.java2novice.algos;


public class IsPerfectNumber {


    public boolean isPerfectNumber(int number){


        int temp = 0;
        for(int i=1;i<=number/2;i++){
            if(number%i == 0){
                temp += i;
            }
        }
        if(temp == number){
            System.out.println("It is a perfect number");
            return true;
        } else {
            System.out.println("It is not a perfect number");
            return false;
        }
    }
```

```
        public static void main(String a[]){
            IsPerfectNumber ipn = new IsPerfectNumber();
            System.out.println("Is perfect number: "+ipn.isPerfectNumber(28));
        }
    }
```

PROGRAM: WRITE A PROGRAM TO CONVERT DECIMAL NUMBER TO BINARY FORMAT.

Description:

Write a program to convert decimal number to binary format using numeric operations. Below example shows how to convert decimal number to binary format using numeric operations.

Code:

```
?
    package com.java2novice.algos;

    public class DecimalToBinary {

        public void printBinaryFormat(int number){
            int binary[] = new int[25];
            int index = 0;
            while(number > 0){
                binary[index++] = number%2;
                number = number/2;
            }
            for(int i = index-1;i >= 0;i--){
                System.out.print(binary[i]);
            }
        }
```

```
        public static void main(String a[]){
            DecimalToBinary dtb = new DecimalToBinary();
            dtb.printBinaryFormat(25);
        }
    }
```

PROGRAM: WRITE A PROGRAM TO CREATE DEADLOCK BET

Description:

Deadlock describes a situation where two or more threads are blocked forever, waiting for each other. Deadlocks can occur in Java when the synchronized keyword causes the executing thread to block while waiting to get the lock, associated with the specified object. Since the thread might already hold locks associated with other objects, two threads could each be waiting fo the other to release a lock. In such case, they will end up waiting forever.

Code:

?

```
package com.java2novice.algos;

public class MyDeadlock {

    String str1 = "Java";
    String str2 = "UNIX";

    Thread trd1 = new Thread("My Thread 1"){
        public void run(){
            while(true){
                synchronized(str1){
                    synchronized(str2){
```

```
                        System.out.println(str1 + str2);
                    }
                }
            }
        };

        Thread trd2 = new Thread("My Thread 2"){
            public void run(){
                while(true){
                    synchronized(str2){
                        synchronized(str1){
                            System.out.println(str2 + str1);
                        }
                    }
                }
            }
        };

        public static void main(String a[]){
            MyDeadlock mdl = new MyDeadlock();
            mdl.trd1.start();
            mdl.trd2.start();
        }
    }
```

PROGRAM: WRITE A PROGRAM TO CHECK THE GIVEN NUMBER IS BINARY NUMBER OR NOT?

Description:

The binary numeral system, or base-2 number system, represents numeric values using two symbols: 0 and 1. More specifically, the usual base-2 system is a positional notation with a radix of 2. Because of its straightforward implementation in digital electronic circuitry using logic gates, the binary system is used internally by almost all modern computers.

Code:

```
?
1   package com.java2novice.algos;
2
3   public class MyBinaryCheck {
4
5       public boolean isBinaryNumber(int binary){
6
7           boolean status = true;
8           while(true){
9               if(binary == 0){
10                  break;
11              } else {
12                  int tmp = binary%10;
13                  if(tmp > 1){
14                      status = false;
15                      break;
16                  }
17                  binary = binary/10;
18              }
19          }
20          return status;
21      }
22
23      public static void main(String a[]){
24          MyBinaryCheck mbc = new MyBinaryCheck();
25          System.out.println("Is 1000111 binary? :"+mbc.isBinaryNumber(1000111));
```

```
24            System.out.println("Is 10300111 binary? :"+mbc.isBinaryNumber(10300111));
25        }
26    }
27
28
```

PROGRAM: WRITE A PROGRAM TO FIND SUM OF EACH DIGIT IN THE GIVEN NUMBER USING RECURSION.

Description:

Below example shows how to find out sum of each digit in the given number using recursion logic. For example, if the number is 259, then the sum should be 2+5+9 = 16.

Code:

```
?
1     package com.java2novice.algos;
2
3     public class MyNumberSumRec {
4
5         int sum = 0;
6
7         public int getNumberSum(int number){
8
9             if(number == 0){
10                return sum;
11            } else {
12                sum += (number%10);
                  getNumberSum(number/10);
```

```
            }
            return sum;
        }

        public static void main(String a[]){
            MyNumberSumRec mns = new MyNumberSumRec();
            System.out.println("Sum is: "+mns.getNumberSum(223));
        }
    }
```

PROGRAM: WRITE A PROGRAM FOR BUBBLE SORT IN JAVA

Description:

Bubble sort is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair o adjacent items and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm gets its name from the way smaller elements bubble to the top of the list. Because it only uses comparisons to operate on elements, it is a comparison sort. You can see the code implementation below:

Code:

```
?
1   package com.java2novice.algos;
2
3   public class MyBubbleSort {
4
5       // logic to sort the elements
6       public static void bubble_srt(int array[]) {
            int n = array.length;
```

```
7               int k;
8               for (int m = n; m >= 0; m--) {
9                   for (int i = 0; i < n - 1; i++) {
10                      k = i + 1;
11                      if (array[i] > array[k]) {
12                          swapNumbers(i, k, array);
13                      }
14                  }
15                  printNumbers(array);
16              }
17          }
18
19          private static void swapNumbers(int i, int j, int[] array) {
20
21              int temp;
22              temp = array[i];
23              array[i] = array[j];
24              array[j] = temp;
25          }
26
27          private static void printNumbers(int[] input) {
28
29              for (int i = 0; i < input.length; i++) {
30                  System.out.print(input[i] + ", ");
31              }
32              System.out.println("\n");
33          }
34
            public static void main(String[] args) {
                int[] input = { 4, 2, 9, 6, 23, 12, 34, 0, 1 };
```

```
35            bubble_srt(input);
36
37        }
38    }
39
40
41
```

PROGRAM: WRITE A PROGRAM FOR INSERTION SORT IN JAVA.

Description:

Insertion sort is a simple sorting algorithm that builds the final sorted array one item at a time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort. Every repetition of insertion sort removes an element from the input data, inserting it into the correct position in the already-sorted list, until no input elements remain. The choice of which element to remove from the input is arbitrary, and can be made using almost any choice algorithm. You can see the code implementation below:

Code:

```
?
1     package com.java2novice.algos;
2
3     public class MyInsertionSort {
4
5         public static void main(String[] args) {
6
7             int[] input = { 4, 2, 9, 6, 23, 12, 34, 0, 1 };
8             insertionSort(input);
        }
```

```
9
10      private static void printNumbers(int[] input) {
11
12          for (int i = 0; i < input.length; i++) {
13              System.out.print(input[i] + ", ");
14          }
15          System.out.println("\n");
16      }
17
18      public static void insertionSort(int array[]) {
19          int n = array.length;
20          for (int j = 1; j < n; j++) {
21              int key = array[j];
22              int i = j-1;
23              while ( (i > -1) && ( array [i] > key ) ) {
24                  array [i+1] = array [i];
25                  i--;
26              }
27              array[i+1] = key;
28              printNumbers(array);
29          }
30      }
31  }
32
```

PROGRAM: WRITE A PROGRAM TO GET DISTINCT WORD LIST FROM THE GIVEN FILE.

Description:

Write a program to find all distinct words from the given file. Remove special chars like ".,;:" etc. Ignore case sensitivity.

Code:

```
?
1   package com.java2novice.algos;
2
3   import java.io.BufferedReader;
4   import java.io.DataInputStream;
5   import java.io.FileInputStream;
6   import java.io.FileNotFoundException;
7   import java.io.IOException;
8   import java.io.InputStreamReader;
9   import java.util.ArrayList;
10  import java.util.List;
11  import java.util.StringTokenizer;
12
13  public class MyDistinctFileWords {
14
15      public List<String> getDistinctWordList(String fileName){
16
17          FileInputStream fis = null;
18          DataInputStream dis = null;
19          BufferedReader br = null;
20          List<String> wordList = new ArrayList<String>();
21          try {
22              fis = new FileInputStream(fileName);
23              dis = new DataInputStream(fis);
              br = new BufferedReader(new InputStreamReader(dis));
              String line = null;
```

```
24                while((line = br.readLine()) != null){
25                    StringTokenizer st = new StringTokenizer(line, " ,.;:\"");
26                    while(st.hasMoreTokens()){
27                        String tmp = st.nextToken().toLowerCase();
28                        if(!wordList.contains(tmp)){
29                            wordList.add(tmp);
30                        }
31                    }
32                }
33            } catch (FileNotFoundException e) {
34                e.printStackTrace();
35            } catch (IOException e) {
36                e.printStackTrace();
37            } finally{
38                try{if(br != null) br.close();}catch(Exception ex){}
39            }
40            return wordList;
41        }
42
43        public static void main(String a[]){
44
45            MyDistinctFileWords distFw = new MyDistinctFileWords();
46            List<String> wordList = distFw.getDistinctWordList("C:/sample.txt");
47            for(String str:wordList){
48                System.out.println(str);
49            }
50        }
51    }
```

```
52
53
```

PROGRAM: WRITE A PROGRAM TO GET A LINE WITH MAX WORD COUNT FROM THE GIVEN FILE.

Description:

Below example shows how to find out the line with maximum number of word count in the given file. In case if it has multiple lines with max number of words, then it has to list all those lines.

Code:

```
?
1   package com.java2novice.algos;
2
3   import java.io.BufferedReader;
4   import java.io.DataInputStream;
5   import java.io.FileInputStream;
6   import java.io.FileNotFoundException;
7   import java.io.IOException;
8   import java.io.InputStreamReader;
9   import java.util.ArrayList;
10  import java.util.List;
11
12  public class MaxWordCountInLine {
13
14      private int currentMaxCount = 0;
15      private List<String> lines = new ArrayList<String>();
16
17      public void readMaxLineCount(String fileName){
```

```
17
18          FileInputStream fis = null;
19          DataInputStream dis = null;
20          BufferedReader br = null;
21
22          try {
23              fis = new FileInputStream(fileName);
24              dis = new DataInputStream(fis);
25              br = new BufferedReader(new InputStreamReader(dis));
26              String line = null;
27              while((line = br.readLine()) != null){
28
29                  int count = (line.split("\\s+")).length;
30                  if(count > currentMaxCount){
31                      lines.clear();
32                      lines.add(line);
33                      currentMaxCount = count;
34                  } else if(count == currentMaxCount){
35                      lines.add(line);
36                  }
37              }
38          } catch (FileNotFoundException e) {
39              e.printStackTrace();
40          } catch (IOException e) {
41              e.printStackTrace();
42          } finally{
43              try{
44                  if(br != null) br.close();
              }catch(Exception ex){}
          }
      }
```

```
45
46      public int getCurrentMaxCount() {
47          return currentMaxCount;
48      }
49
50      public void setCurrentMaxCount(int currentMaxCount) {
51          this.currentMaxCount = currentMaxCount;
52      }
53
54      public List<String> getLines() {
55          return lines;
56      }
57
58      public void setLines(List<String> lines) {
59          this.lines = lines;
60      }
61
62      public static void main(String a[]){
63
64          MaxWordCountInLine mdc = new MaxWordCountInLine();
65          mdc.readMaxLineCount("/Users/ngootooru/MyTestFile.txt");
66          System.out.println("Max number of words in a line is: "+mdc.getCurrentMaxCou
67          System.out.println("Line with max word count:");
68          List<String> lines = mdc.getLines();
69          for(String l:lines){
70              System.out.println(l);
71          }
72      }
  }
```

```
73
74
75
76
77
```

PROGRAM: WRITE A PROGRAM TO CONVERT STRING TO NUMBER WITHOUT USING INTEGER.PARSEINT() METHOD.

Description:

Below example shows how to convert string format of a number to number without calling Integer.parseInt() method. We can do this by converting each character into ascii format and form the number.

Code:

```
?
1   package com.java2novice.algos;
2
3   public class MyStringToNumber {
4
5       public static int convert_String_To_Number(String numStr){
6
7           char ch[] = numStr.toCharArray();
8           int sum = 0;
9           //get ascii value for zero
10          int zeroAscii = (int)'0';
11          for(char c:ch){
12              int tmpAscii = (int)c;
13              sum = (sum*10)+(tmpAscii-zeroAscii);
```

```
13              }
14              return sum;
15          }
16
17          public static void main(String a[]){
18
19              System.out.println("\"3256\" == "+convert_String_To_Number("3256"));
20              System.out.println("\"76289\" == "+convert_String_To_Number("76289"));
21              System.out.println("\"90087\" == "+convert_String_To_Number("90087"));
22          }
23      }
24
```

PROGRAM: WRITE A PROGRAM TO FIND TWO LINES WITH

Description:

Write a program to read a multiple line text file and write the 'N' longest lines to the output console, where the file to be read i specified as command line aruguments. The program should read an input file. The first line should contain the value of the number 'N' followed by multiple lines. 'N' should be a valid positive integer.

Code:

```
?
1       package com.longest.lines;
2
3       import java.io.BufferedReader;
4       import java.io.File;
5       import java.io.FileNotFoundException;
        import java.io.FileReader;
```

```
6   import java.io.IOException;
7   import java.util.Comparator;
8   import java.util.Set;
9   import java.util.TreeSet;
10
11  public class Main {
12
13      public static void main(String[] args) {
14
15          BufferedReader br = null;
16          String filePath = args[0];
17          int topList = 0;
18          Set<Entries> liSet = new TreeSet<Entries>(new MyComp());
19          try {
20              br = new BufferedReader(new FileReader(new File(filePath)));
21              String line = br.readLine();
22              topList = Integer.parseInt(line.trim());
23              while((line = br.readLine()) != null){
24                  line = line.trim();
25                  if(!"".equals(line)){
26                      liSet.add(new Entries(line.length(), line));
27                  }
28              }
29              int count = 0;
30              for(Entries ent:liSet){
31                  System.out.println(ent.line);
32                  if(++count == topList){
33                      break;
                    }
                }
            } catch (FileNotFoundException e) {
```

```
34                 // TODO Auto-generated catch block
35                 e.printStackTrace();
36             } catch (IOException e) {
37                 // TODO Auto-generated catch block
38                 e.printStackTrace();
39             }
40         }
41 
42         public static class Entries{
43             Integer length;
44             String line;
45             public Entries(Integer l,String line){
46                 length = l;
47                 this.line = line;
48             }
49         }
50 
51         public static class MyComp implements Comparator<Entries>{
52 
53             @Override
54             public int compare(Entries e1, Entries e2) {
55                 if(e2.length > e1.length){
56                     return 1;
57                 } else {
58                     return -1;
59                 }
60             }
61 
62         }
63     }
64 
```

```
62
63
64
65
66
67
```

PROGRAM: WRITE A PROGRAM TO FIND THE SUM OF THE FIRST 1000 PRIME NUMBERS.

Description:

Write a program to find the sum of the first 1000 prime numbers.

Code:

```
?
1   package com.primesum;
2
3   public class Main {
4
5       public static void main(String args[]){
6
7           int number = 2;
8           int count = 0;
9           long sum = 0;
10          while(count < 1000){
11              if(isPrimeNumber(number)){
12                  sum += number;
                    count++;
```

```
13                  }
14                  number++;
15              }
16              System.out.println(sum);
17          }
18
19          private static boolean isPrimeNumber(int number){
20
21              for(int i=2; i<=number/2; i++){
22                  if(number % i == 0){
23                      return false;
24                  }
25              }
26              return true;
27          }
28      }
29
```

PROGRAM: FIND LONGEST SUBSTRING WITHOUT REPEATING CHARACTERS.

Description:

Given a string, find the longest substrings without repeating characters. Iterate through the given string, find the longest maximum substrings.

Code:

?

```
1   package com.java2novice.algos;
```

```
2
3   import java.util.HashSet;
4   import java.util.Set;
5
6   public class MyLongestSubstr {
7
8       private Set<String> subStrList = new HashSet<String>();
9       private int finalSubStrSize = 0;
10
11      public Set<String> getLongestSubstr(String input){
12          //reset instance variables
13          subStrList.clear();
14          finalSubStrSize = 0;
15          // have a boolean flag on each character ascii value
16          boolean[] flag = new boolean[256];
17          int j = 0;
18          char[] inputCharArr = input.toCharArray();
19          for (int i = 0; i < inputCharArr.length; i++) {
20              char c = inputCharArr[i];
21              if (flag[c]) {
22                  extractSubString(inputCharArr,j,i);
23                  for (int k = j; k < i; k++) {
24                      if (inputCharArr[k] == c) {
25                          j = k + 1;
26                          break;
27                      }
28                      flag[inputCharArr[k]] = false;
29                  }
              } else {
                  flag[c] = true;
              }
```

```
30              }
31              extractSubString(inputCharArr,j,inputCharArr.length);
32              return subStrList;
33          }
34
35          private String extractSubString(char[] inputArr, int start, int end){
36
37              StringBuilder sb = new StringBuilder();
38              for(int i=start;i<end;i++){
39                  sb.append(inputArr[i]);
40              }
41              String subStr = sb.toString();
42              if(subStr.length() > finalSubStrSize){
43                  finalSubStrSize = subStr.length();
44                  subStrList.clear();
45                  subStrList.add(subStr);
46              } else if(subStr.length() == finalSubStrSize){
47                  subStrList.add(subStr);
48              }
49
50              return sb.toString();
51          }
52
53          public static void main(String a[]){
54              MyLongestSubstr mls = new MyLongestSubstr();
55              System.out.println(mls.getLongestSubstr("java2novice"));
56              System.out.println(mls.getLongestSubstr("java_language_is_sweet"));
57              System.out.println(mls.getLongestSubstr("java_java_java_java"));
              System.out.println(mls.getLongestSubstr("abcabcbb"));
          }
      }
```

```
58
59
60
61
62
63
```

PROGRAM: WRITE A PROGRAM TO REMOVE DUPLICATES FROM SORTED ARRAY.

Description:

Given array is already sorted, and it has duplicate elements. Write a program to remove duplicate elements and return new array without any duplicate elements. The array should contain only unique elements.

Code:

```
?
1   package com.java2novice.algos;
2
3   public class MyDuplicateElements {
4
5       public static int[] removeDuplicates(int[] input){
6
7           int j = 0;
8           int i = 1;
9           //return if the array length is less than 2
10          if(input.length < 2){
11              return input;
12          }
            while(i < input.length){
```

```
13                  if(input[i] == input[j]){
14                      i++;
15                  }else{
16                      input[++j] = input[i++];
17                  }
18              }
19              int[] output = new int[j+1];
20              for(int k=0; k<output.length; k++){
21                  output[k] = input[k];
22              }
23
24              return output;
25          }
26
27          public static void main(String a[]){
28              int[] input1 = {2,3,6,6,8,9,10,10,10,12,12};
29              int[] output = removeDuplicates(input1);
30              for(int i:output){
31                  System.out.print(i+" ");
32              }
33          }
34      }
35
```

PROGRAM: HOW TO SORT A STACK USING A TEMPORARY S

Description:

You have a stack with full of integers. Sort it in the ascending order using another temporary array by using all stack functionality.

Code:

```
?
1   package com.java2novice.algo;
2
3   import java.util.Stack;
4
5   public class StackSort {
6
7       public static Stack<Integer> sortStack(Stack<Integer> input){
8
9           Stack<Integer> tmpStack = new Stack<Integer>();
10          System.out.println("=============== debug logs ================");
11          while(!input.isEmpty()) {
12              int tmp = input.pop();
13              System.out.println("Element taken out: "+tmp);
14              while(!tmpStack.isEmpty() && tmpStack.peek() > tmp) {
15                  input.push(tmpStack.pop());
16              }
17              tmpStack.push(tmp);
18              System.out.println("input: "+input);
19              System.out.println("tmpStack: "+tmpStack);
20          }
21          System.out.println("=============== debug logs ended ================");
22          return tmpStack;
23      }
```

```
24
25            Stack<Integer> input = new Stack<Integer>();
26            input.add(34);
27            input.add(3);
28            input.add(31);
29            input.add(98);
30            input.add(92);
31            input.add(23);
32            System.out.println("input: "+input);
33            System.out.println("final sorted list: "+sortStack(input));
34        }
35    }
36
37
```

Algorithm	Average Time	Worst Time	Space
Bubble sort	n^2	n^2	1
Selection sort	n^2	n^2	1
Insertion sort	n^2	n^2	
Quick sort	n log(n)	n^2	
Merge sort	n log(n)	n log(n)	depends

Stack: What is stack? Stack is a linear data structure which implements data on last in first out criteria. Here is a java program to implement stack using linked list.

```
class Stack{
    Node top;
```

```
    public Node peek(){
        if(top != null){
            return top;
        }

        return null;
    }

    public Node pop(){
        if(top == null){
            return null;
        }else{
            Node temp = new Node(top.val);
            top = top.next;
            return temp;
        }
    }

    public void push(Node n){
        if(n != null){
            n.next = top;
            top = n;
        }
    }
}
```

Queue: Queue is a data structure, that uses First in First out(FIFO) principle. Queue can be implemented by stack, array and linked list. Here is java program to implement queue using linked list.

view source
print?

```
class Queue{
    Node first, last;

    public void enqueue(Node n){
        if(first == null){
            first = n;
            last = first;
        }else{
            last.next = n;
            last = n;
        }
```

```
    }

    public Node dequeue(){
        if(first == null){
            return null;
        }else{
            Node temp = new Node(first.val);
            first = first.next;
            return temp;
        }
    }
}
```