coding interview

java interview questions and answers

Top 60 Core Java Interview Questions and Answers
&
Top 20 Most Asked Java Programming Questions

We are sharing Top 60 Core Java Interview Questions and Answers

And Top 20 java interview Programming questions; these questions are frequently asked by the recruiters. Java questions can be asked from any core java topic. So we try our best to provide you the java interview questions and answers for experienced &

fresher which should be in your to do list before facing java questions in technical interview.

This Page Was Intentionally Left Blank

Top 60 Most Asked Core Java Interview Questions!

1. What is the most important feature of Java?

Java is a platform independent language.

2. What do you mean by platform independence?

Platform independence

means that we can write and compile the java code in one platform (For example: Windows) and can execute the class in any other supported platform (Linux,Solaris,etc).

3. What is a JVM?

JVM is Java Virtual Machine which is a run time environment for the compiled java class files.

4. Are JVM’s platforms independent?

JVM's are not platform independent. JVM's are platform specific run time implementation provided by the vendor.

5. What is the difference between a JDK and a JVM?

JDK is Java Development Kit

which is for development purpose and it includes execution environment also. But JVM is purely a run time environment and hence you will not be able to compile your source files using a JVM.

6. What is a pointer and does Java support pointers?

Pointer is a reference handle to a memory location.

Improper handling of pointers leads to memory leaks and reliability issues hence Java doesn't support the usage of pointers.

7. What is the base class of all classes?

java.lang.Object

8. Does Java support multiple inheritance?

Java doesn't support multiple inheritance.

9. Is Java a pure object oriented language?

Java uses primitive data types and hence is not a

pure object oriented language.

10. Are arrays primitive data types?

In Java, Arrays are objects.

1. What is difference between Path and Classpath?

Path and Classpath are operating system level environment variables. Path is used define where the system can find the

executable (.exe) files and classpath is used to specify the location .class files.

12. What are local variables?

Local variables are those which are declared within a block of code like methods. Local variables should be initialized before accessing them.

13. What are instance

variables?

Instance variables are those which are defined at the class level. Instance variables need not be initialized before using them as they are automatically initialized to their default values.

14. How to define a constant variable in Java?

The variable should be

declared as static and final. So only one copy of the variable exists for all instances of the class and the value can't be changed also.

static final int MAX_LENGTH = 50; is an example for constant.

15. Should a main() method be compulsorily declared in all java classes?

No not required. main() method should be defined only if the source class is a java application.

16. What is the return type of the main() method?

Main() method doesn't return anything hence declared void.

17. Why is the main() method declared static?

main() method is called by the JVM even before the instantiation of the class hence it is declared as static.

18. What is the argument of main()

method?

main() method accepts an array of String object as an argument.

19. Can a main() method be overloaded?

Yes. You can have any number of main() methods with different method signature and implementation in the class.

20. Can a main() method

be declared final?

Yes. Any inheriting class will not be able to have it's own default main() method.

21. Does the order of public and static declaration matter in main() method?

No. It doesn't matter but void should always come before main().

22. Can a source file

contain more than one class declaration?

Yes a single source file can contain any number of Class declarations but only one of the class can be declared as public.

23. What is a package?

Package is a collection of

related classes and interfaces. package declaration should be first statement in a java class.

24. Which package is imported by default?

java.lang package is imported by default even without a package declaration.

25. Can a class declared

as private be accessed outside it's package?

Not possible.

26. Can a class be declared as protected?

The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declared protected, however methods and fields in a interface cannot be

declared protected.

27. What is the access scope of a protected method?

A protected method can be accessed by the classes within the same package or by the subclasses of the class in any package.

28. What is the purpose of declaring a variable as

final?

A final variable's value can't be changed. final variables should be initialized before using them.

29. What is the impact of declaring a method as final?

A method declared as final can't be overridden. A sub-class can't have the same method signature with a different implementation.

30. I don't want my class to be inherited by any other class. What should i do?

You should declare your class as final. But you can't define your class as final, if it is an abstract class. A class declared as final can't be extended by any other class.

31. Can you give few

examples of final classes defined in Java API?

java.lang.String, java.lang.Math are final class

32. How is final different from finally and finalize()?

final is a modifier which can be applied to a class or a method or a variable. final class can't be inherited, final method can't be overridden

and final variable can't be changed.

Finally is an exception handling code section which gets executed whether an exception is raised or not by the try block code segment.

finalize() is a method of Object class which will be executed by the JVM just before garbage collecting object to give a final chance

for resource releasing activity.

33. Can a class be declared as static?

We cannot declare top level class as static, but only inner class can be declared static.

```
public class Test
{
    static class InnerClass
    {
        public static void
```

```
InnerMethod()
        {
System.out.println("Static
Inner Class!"); }
    }
    public static void
main(String args[])
    {

Test.InnerClass.InnerMethod(
    }
}
//output: Static Inner Class!
```

34. When will you define a method as static?

When a method needs to be accessed even before the creation of the object of the class then we should declare the method as static.

35. What are the

restrictions imposed on a static method or a static block of code?

A static method should not refer to instance variables without creating an instance and cannot use "this" operator to refer the instance.

36. I want to print "Hello" even before main() is executed. How

will you achieve that?

Print the statement inside a static block of code. Static blocks get executed when the class gets loaded into the memory and even before the creation of an object. Hence it will be executed before the main() method. And it will be executed only once.

37. What is the importance of static

variable?

static variables are class level variables where all objects of the class refer to the same variable. If one object changes the value then the change gets reflected in all the objects.

38. Can we declare a static variable inside a method?

Static variables are class

level variables and they can't be declared inside a method. If declared, the class will not compile.

39. What is an Abstract Class and what is its purpose?

A Class which doesn't provide complete implementation is defined as an abstract class. Abstract classes enforce abstraction.

40. Can a abstract class be declared final?

Not possible. An abstract class without being inherited is of no use and hence will result in compile time error.

41. What is use of a abstract variable?

Variables can't be declared

as abstract. Only classes and methods can be declared as abstract.

42. Can you create an object of an abstract class?

Not possible. Abstract classes can't be instantiated.

43. Can a abstract class be defined without any abstract methods?

Yes it's possible. This is

basically to avoid instance creation of the class.

44. Class C implements Interface I containing method m1 and m2 declarations. Class C has provided implementation for method m2. Can i create an object of Class C?

No not possible. Class C should provide implementation for all the

methods in the Interface I.
Since Class C didn't provide implementation
for m1 method, it has to be declared as abstract.
Abstract classes can't be instantiated.

45. Can a method inside a Interface be declared as final?

No not possible. Doing so will result in compilation

error. public and abstract are the only applicable modifiers for method declaration in an interface.

46. Can an Interface implement another Interface?

Interfaces doesn't provide implementation hence a interface cannot implement another interface.

47. Can an Interface extend another Interface?

Yes an Interface can inherit another Interface, for that matter an Interface can extend more than one Interface.

48. Can a Class extend

more than one Class?

Not possible. A Class can extend only one class but can implement any number of Interfaces.

49. Why is an Interface be able to extend more than one Interface but a Class can't extend more than one Class?

Basically Java doesn't allow multiple inheritance, so a

Class is restricted to extend only one Class. But an Interface is a pure abstraction model and doesn't have inheritance hierarchy like classes (do remember that the base class of all classes is Object). So an Interface is allowed to extend more than one Interface.

50. Can an Interface be

final?

Not possible. Doing so will result in compilation error.

51. Can a class be defined inside an Interface?

Yes it's possible.

52. Can an Interface be defined inside a class?

Yes it's possible.

53. What is a Marker

Interface?

An Interface which doesn't have any declaration inside but still enforces a mechanism.

54. Which object oriented Concept is achieved by using overloading and overriding?

Polymorphism.

55. Why does Java not support operator overloading?

Operator overloading makes the code very difficult to read and maintain. To maintain code simplicity, Java doesn't support operator overloading.

56. Can we define private and protected modifiers for variables in interfaces?

No.

57. What is Externalizable?

Externalizable is an Interface that extends Serializable Interface. And sends data into Streams in Compressed Format. It has

two
methods, writeExternal(Obje
out) and readExternal(Object
in)

58. What modifiers are allowed for methods in an Interface?

Only public and abstract mod
are allowed for methods in
interfaces.

59. What is a local,

member and a class variable?

Variables declared within a method are "local" variables. Variables declared within the class i.e not within any methods are "member" variables (global variables). Variables declared within the class i.e not within any methods and are defined as "static" are class variables.

60. What is an abstract method?

An abstract method is a method whose implementation is deferred to a subclass.

20 Most Asked Programming Questions!

Q.1) PROGRAM TO CHECK UNIQUE NUMBER IN JAVA

```
import java.util.*;
import java.io.*;

  public class IsUnique
{
```

```
public static boolean
isUniqueUsingHash(String word)
{
char[] chars = word.toCharArray();

Set<Character> set = new
HashSet<Character>();
for (char c : chars)

if (set.contains(c))
return false;
 else
set.add(c);
return true;

}

public static boolean
```

```
isUniqueUsingSort(String word)
    {
    char[] chars = word.toCharArray();

if (chars.length <= 1)
return true;
Arrays.sort(chars);




char temp = chars[0];

for (int i = 1; i < chars.length; i++)
```

```
{
if (chars[i] == temp)
return false;
temp = chars[i];
}

return true;

}

public static void main(String[] args)
throws IOException
{
System.out.println(isUniqueUsingHash(
? "Unique" : "Not Unique");
System.out.println(isUniqueUsingSort("
? "Unique" : "Not Unique");
```

```
}

}
```

Output:

```
Unique
Not Unique
```

Q.2) PROGRAM TO FIND PERMUTATION OF A STRING

```
import java.util.*;
import java.io.*;

public class CheckPermutations
    {

    public static boolean
isPermutation(String    s1,
String s2)
```

```
        {
                char[] a =
 s1.toCharArray();


 char[]
   b =
s2.toCharArray();

Arrays.sort(a);
Arrays.sort(b);
             if
(a.length   !=
b.length)
          return
false;
        for (int
```

```
i = 0; i <
a.length;
i++)
        {

      if (a[i] != b[i]) return false;

        }

        return true;

        }



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

```
        System.out.println(isPermutation("a
"cba") ? "It is a permutation" :
        "It is not a permutation");

    System.out.println(isPermutation("test
    "estt") ? "It is a permutation" : "It is
    not a permutation");

    System.out.println(isPermutation("test
    "estt") ? "It is a permutation" : "It is
    not a  permutation");

        }

        }
```

Output:

```
It is a permutation
It is a permutation
It is not a  permutation
```

Q.3) PROGRAM TO PUT HTML LINKS AROUND URLS STRINGS

```
import java.util.*;
    import java.io.*;

  public class URLify
    {
public static char[] URLify(char[]
chars, int len)
{
  int spaces = countSpaces(chars, len);
```

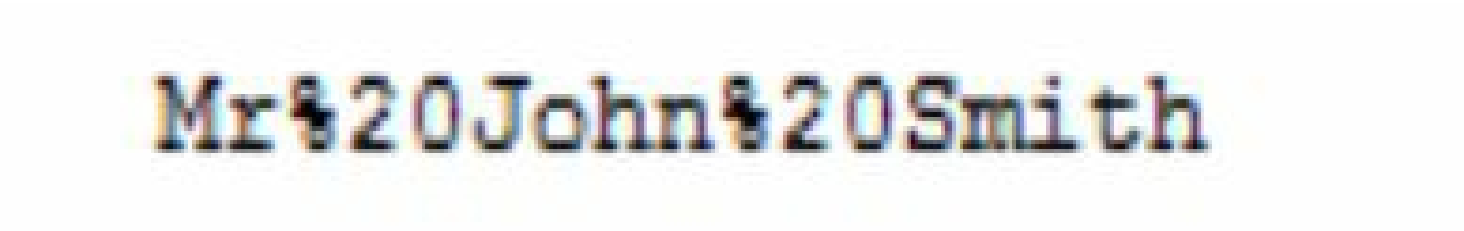
```
int end = len - 1 + spaces *2;
for (int i = len - 1; i >= 0; i-)
{
 if (chars[i] == ' ')
{

chars[end - 2] ='%';
chars[end - 1] ='2';
chars[end] = '0';
 end -= 3;

}
else
{
chars[end] =chars[i]; end--;
}

}
```

```
return chars;
}
```

```
static int countSpaces(char[] chars, int
len)
{
int count = 0;

for (int i = 0; i < len;i++)
```

```
if (chars[i] == ' ') count++;

return count;

}

public static void main(String[] args)
throws IOException
{
   char[] chars = "Mr John Smith
".toCharArray();
System.out.println(URLify(chars,
13));

}

}
```

Output:



Q.4) PROGRAM TO CHECK PALINDROME PERMUTATIONS OF A STRING

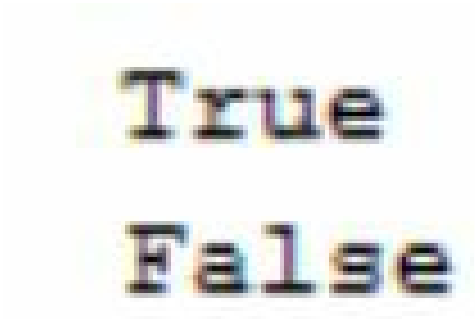
```
import java.util.*;


  public class PalindromePermutation
{

public static boolean
```

```
permuteHash(String str)
{
Map<Character, Integer> map = new
HashMap<Character, Integer>();
for (int i = 0; i < str.length(); i++) {

Character c =
Character.toLowerCase(str.charAt(i));

if (!Character.isLetter(c))
continue;

if (map.containsKey(c))
map.put(c, map.get(c) + 1);
else
map.put(c, 1);
}
```

```
int odd = 0;

for (Character key : map.keySet())
if (map.get(key) % 2 != 0)
odd++;




if (odd > 1)
return false;
else
return true;
}
```

```
public static void main(String[] args)
{
System.out.println(permuteHash("Tact
Coa") ? "True" : "False");
System.out.println(permuteHash("test"
? "True" : "False");
  }
```

Output:



Q.5) PROGRAM TO COMPRESS STRING

```
public class
StringCompression
{
public String
compress(String input)
{
char[] cs =
```

```
input.toCharArray();
char temp = cs[0];
int i = 0, j = 0, count = 0,
len = cs.length;
while(j < len)
{
cs[i++] = temp;
while (j < len && temp ==
cs[j])
{
j++;
count++;
}
```

```
if(j < len)
temp = cs[j++];
cs[i++] =
String.valueOf(count).charAt
count = 1;
}
return new String(cs, 0, i);
}
```

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

StringCompression
compression = new
StringCompression();
System.out.println(compressi
}
}
```

Output:

Q.6) PROGRAM TO ROTATE MATRIX

```
import java.util.*;

public class RotateMatrix {

public static void rotate(int[][]
matrix)
{
int n = matrix.length;

for (int layer = 0; layer < n / 2;
layer++)
{
```

```
int first = layer;
int last = n - 1 - layer;
for (int i = first; i < last; i++)
{
int offset = i - first;
int top = matrix[first][i];

// left -> top

matrix[first][i] = matrix[last-offset]
[first];
// bottom -> left
matrix[last-offset][first] =
matrix[last][last-offset];
// right -> bottom matrix[last][last-
offset] = matrix[i][last];
// top -> right matrix[i][last] = top;
}
```

```
}

}
```

```
public static void main(String[] args)
{
int[][] arr = new int[][]
{

{1, 2, 3, 4, 5},
{6, 7, 8, 9, 10},
```

```
{11, 12, 13, 14, 15},
{16, 17, 18, 19, 20},
{21, 22, 23, 24, 25}

};

rotate(arr);

for (int[] a : arr)
System.out.println(Arrays.toString();
   }
```

Output:

```
[21, 16, 11, 6, 1]
[22, 17, 12, 7, 2]
[23, 18, 13, 8, 3]
[24, 19, 14, 9, 4]
[25, 20, 15, 10, 5]
```

Q.7) PROGRAM TO CONVERT ALL 1 INTO ZERO MATRIX.

```
import java.util.*;

  public class ZeroMatrix {

public static void zero(int[][] matrix)
{
int m = matrix.length;

int n = matrix[0].length;
```

```
boolean[] row = new boolean[m];
boolean[] col = new boolean[n];

for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
if (matrix[i][j] == 0) {
row[i] = true;
col[j] = true;
}
}
}

for (int i = 0; i < m; i++) {
  for (int j = 0; j < n; j++) {
if (row[i] == true || col[j] == true)
matrix[i][j] = 0;
}
}
```

```
return;

}




public static void main(String[] args)
{
int[][] matrix = new int[][]
{
{0, 1, 1, 1, 0},
{1, 0, 1, 0, 1},
{1, 1, 1, 1, 1},
{1, 0, 1, 1, 1},
{1, 1, 1, 1, 1}
};
```

```
for (int i = 0; i < matrix.length; i++)
System.out.println(Arrays.toString(mat
zero(matrix);
System.out.println();

for (int i = 0; i < matrix.length; i++)
System.out.println(Arrays.toString(mat
}
}
```

Output:

```
[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 0, 0]
[0, 0, 1, 0, 0]
```

Q.8) PROGRAM TO ROTATE STRING.

```
import java.util.*;

 public class StringRotation {

public static boolean
isRotation(String s1, String s2)
{
if (s1.length() != s2.length())
return false;
String s3 = s1 + s1;
return s3.contains(s2);

}
```

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

System.out.println(isRotation("waterbottl
"erbottlewat") ? "True" :
    "False");
    System.out.println(isRotation("waterbo
    "erbottlewat") ? "True" : "False");
    System.out.println(isRotation("pandees
    "deeshpand") ? "True" : "False");
    }
```

Output:

```
True
False
False
```

Q.9) PROGRAM TO ADD TWO NUMBERS WITHOUT USING PLUS ('+') SIGN

```
public class AddWithoutPlus
{
public static int add(int a, int b)
{
if (b > a)
{
int temp = b;
b = a;
```

```
a = temp;
}

int carry = 0;
while (b != 0)
{
carry = a & b;
a = a ^ b;
b = carry << 1;
}
return a;

}

public static int recursiveAdd(int a,
int b)
{
if (b == 0)
```

```
return a;
else

return recursiveAdd(a ^ b, (a & b) <<
1);
}




public static void main(String[] args)
{
int a = 12;

int b = 34;
System.out.println(add(a, b));
```

```
System.out.println(recursiveAdd(b, a));

}
```

Output:

Q.10) PROGRAM TO REMOVE DUPLICATES CHARACTER FROM A STRING

```
import java.util.*;

public class RemoveDups {
public static class Node {
Node next;

char val;

public Node(char val)
```

```
{
this.val = val;
}

public String toString() {

StringBuilder sb = new
StringBuilder();
Node temp = this;

while (temp != null)
{
sb.append(temp.val);
temp = temp.next;
}

return sb.toString();
```

```
}
}
```

```
public static void
removeDupes(Node node)
{
Set<Character> set = new
HashSet<Character>();
set.add(node.val);

Node prev = node;
```

```
Node temp = node.next;

while (temp != null)
{
if (set.contains(temp.val))
{
prev.next = temp.next;
}

else
{
set.add(temp.val);

prev = temp;
}

temp = temp.next;
```

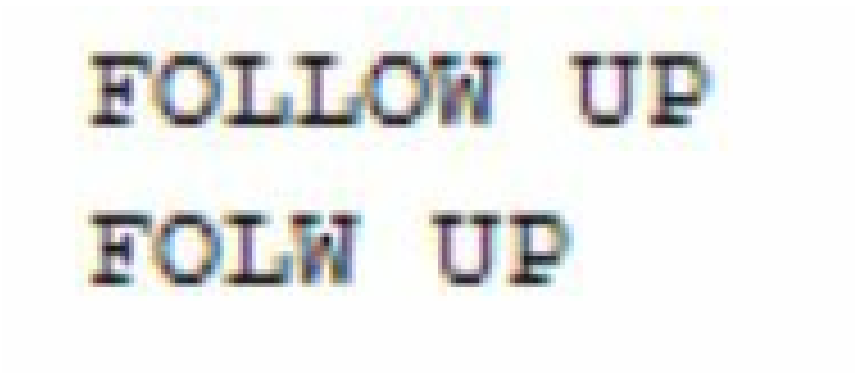
```
}

}
```

```
public static void main(String[] args)
{
Node a = new Node('F');
```

```
Node b = new Node('O');
Node c = new Node('L');
Node d = new Node('L');
Node e = new Node('O');
Node f = new Node('W');
Node g = new Node(' ');
Node h = new Node('U');
Node i = new Node('P');

a.next = b;
b.next = c;
c.next = d;
d.next = e;
e.next = f;
f.next = g;
g.next = h;
h.next = i;
```

```
System.out.println(a);
removeDupes(a);
System.out.println(a);
}
}
```

Output:



Q.11) PROGRAM TO RETURN A CHARACTER FROM THE STRING.

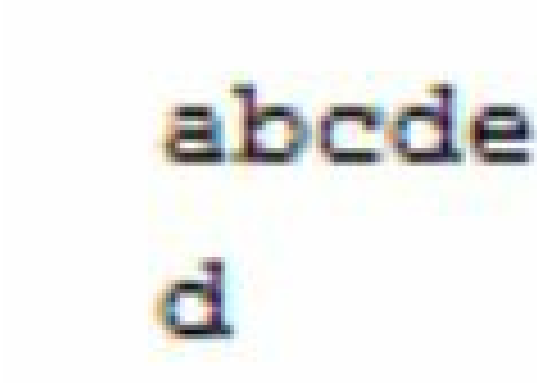
```
import java.util.*;


public class ReturnKth {
  public static class Node {
    Node next;
    char val;
    public Node(char val) {
      this.val = val;
    }
    public String toString() {
```

```
      StringBuilder sb = new
StringBuilder();
      Node temp = this;
      while (temp != null) {
        sb.append(temp.val);
        temp = temp.next;
      }
      return sb.toString();
    }
  }
  public static Node returnKth(Node
node, int k) {
    k--;
    Node first = node;
    Node last = node;
    for (int i = 0; i < k; i++)
      last = last.next;
    while (last.next != null) {
```

```
      last = last.next;
      first = first.next;
    }

  return first;
  }

  public static void main(String[]
args) {
    Node a = new Node('a');
    Node b = new Node('b');
    Node c = new Node('c');
    Node d = new Node('d');
```

```
    Node e = new Node('e');
    a.next = b;
    b.next = c;
    c.next = d;
    d.next = e;
    System.out.println(a);
    System.out.println(returnKth(a,
2).val);
  }
}
```

Output:



Q.12) PROGRAM TO REMOVE MIDDLE CHARACTER FROM A STRING

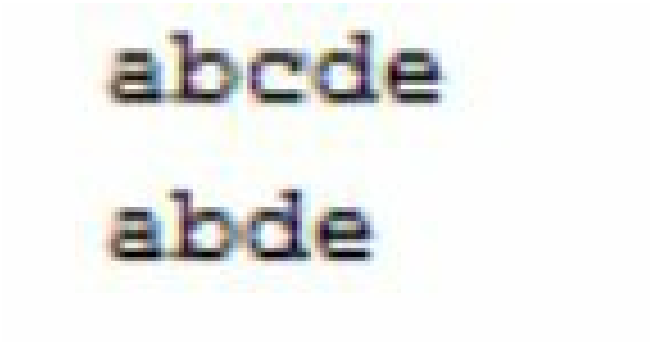
```
import java.util.*;

public class DeleteMiddle
{
public static class Node
{
Node next;
char val;

public Node(char val)
```

```
{
this.val = val;
}

public String toString()
{
StringBuilder sb = new StringBuilder();
Node temp = this;
while (temp != null)
{
sb.append(temp.val);
temp = temp.next;
}
return sb.toString();
}
}
```

```
public static boolean
deleteMiddle(Node node) {
if (node == null || node.next == null)
{
return false;
}
else
{
node.val = node.next.val;
node.next = node.next.next;
return true;
}
}

public static void main(String[] args ) {
   Node a = new Node('a');
```

```
    Node b = new Node('b');
    Node c = new Node('c');
    Node d = new Node('d');
     Node e = new Node('e');
a.next = b;
b.next = c;
c.next = d;
d.next = e;
System.out.println(a);
deleteMiddle(c);
System.out.println(a);
}
}
```

Output:


abcde
abde

Q.13) WRITE A PROGRAM FOR BUBBLE SORT IN JAVA

```
public class MyBubbleSort {

    // logic to sort the elements
    public static void bubble_srt(int
array[])
{
        int n = array.length;
        int k;
        for (int m = n; m >= 0; m--) {
            for (int i = 0; i < n - 1; i++) {
                k = i + 1;
                if (array[i] > array[k]) {
```

```
                    swapNumbers(i, k,
array);
                }
            }
            printNumbers(array);
        }
    }

    private static void
swapNumbers(int i, int j, int[] array)
 {
        int temp;
        temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }
```

```
    private static void
printNumbers(int[] input) {

        for (int i = 0; i < input.length;
i++) {
            System.out.print(input[i] + ",
");
        }
        System.out.println("\n");
    }

    public static void main(String[]
args) {
        int[] input = { 4, 2, 9, 6, 23, 12,
34, 0, 1 };
```

```
        bubble_srt(input);

    }
}
```

Output:

2, 4, 6, 9, 12, 23, 0, 1, 34,

2, 4, 6, 9, 12, 0, 1, 23, 34,

2, 4, 6, 9, 0, 1, 12, 23, 34,

2, 4, 6, 0, 1, 9, 12, 23, 34,

2, 4, 0, 1, 6, 9, 12, 23, 34,

2, 0, 1, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,

Q.14) WRITE A PROGRAM FOR INSERTION SORT IN JAVA.

```
public class MyInsertionSort {

    public static void main(String[]
args)
{
        int[] input = { 4, 2, 9, 6, 23, 12,
34, 0, 1 };
        insertionSort(input);
    }
```

```
    private static void
printNumbers(int[] input)
{

        for (int i = 0; i < input.length;
i++) {
            System.out.print(input[i] + ",
");
        }
        System.out.println("\n");
    }

    public static void insertionSort(int
array[])
 {
        int n = array.length;
        for (int j = 1; j < n; j++) {
```

```
            int key = array[j];
            int i = j-1;
            while ( (i > -1) && ( array [i]
> key ) ) {
                array [i+1] = array [i];
                i--;
            }

            array[i+1] = key;
            printNumbers(array);
        }

}

}
```

Output:

2, 4, 9, 6, 23, 12, 34, 0, 1,

2, 4, 9, 6, 23, 12, 34, 0, 1,

2, 4, 6, 9, 23, 12, 34, 0, 1,

2, 4, 6, 9, 23, 12, 34, 0, 1,

2, 4, 6, 9, 12, 23, 34, 0, 1,

2, 4, 6, 9, 12, 23, 34, 0, 1,

0, 2, 4, 6, 9, 12, 23, 34, 1,

0, 1, 2, 4, 6, 9, 12, 23, 34,

Q.15) WRITE A PROGRAM TO IMPLEMENT HASHCODE AND EQUALS.

Description:

The hashcode of a Java Object is simply a number, it is 32-bit signed int, that allows an object to be managed by a hash-based data structure. We know that hash code is an unique id number allocated to an object by JVM. But actually speaking,

Hash code is not an unique number for an object.

If two objects are equals then these two objects should return same hash code. So we have to implement hashcode() method of a class in such way that if two objects are equals, ie compared by equal() method of that class, then those two objects must return same hash code. If you are overriding hashCode you need to override equals method also.

The below example shows how to override equals and hashcode methods. The class Price overrides equals and hashcode. If you notice the

hashcode implementation, it always generates unique hashcode for each object based on their state, ie if the object state is same, then you will get same hashcode. A HashMap is used in the example to store Price objects as keys. It shows though we generate different objects, but if state is same, still we can use this as key.

```
import java.util.HashMap;

public class MyHashcodeImpl {

    public static void main(String a[])
{
        HashMap<Price, String> hm =
new HashMap<Price, String>();
        hm.put(new Price("Banana",
20), "Banana");
        hm.put(new Price("Apple", 40),
"Apple");
        hm.put(new Price("Orange",
30), "Orange");
        //creating new object to use as
key to get value
```

```
        Price key = new
Price("Banana", 20);
        System.out.println("Hashcode of
the key: "+key.hashCode());
        System.out.println("Value from
map: "+hm.get(key));
    }
}

class Price{
    private String item;
    private int price;
    public Price(String itm, int pr){
        this.item = itm;
        this.price = pr;
    }
    public int hashCode(){
        System.out.println("In
```

```
hashcode");
        int hashcode = 0;
        hashcode = price*20;
        hashcode += item.hashCode();
        return hashcode;
    }


    public boolean equals(Object obj)
{
        System.out.println("In equals");
        if (obj instanceof Price) {
            Price pp = (Price) obj;
            return
(pp.item.equals(this.item) &&
pp.price == this.price);
```

```
}


else {
            return false;
        }
    }

    public String getItem() {
        return item;
    }
    public void setItem(String item) {
        this.item = item;
    }
    public int getPrice() {
        return price;
    }
    public void setPrice(int price) {
```

```
        this.price = price;
    }
    public String toString(){
        return "item: "+item+"  price:
"+price;
    }
}
```

Output:

```
In hashcode
In hashcode
In hashcode
In hashcode
Hashcode of the key: 1982479637
In hashcode
In equals
Value from map: Banana
```

Q.16) HOW TO GET DISTINCT ELEMENTS FROM AN ARRAY BY AVOIDING DUPLICATE ELEMENTS?

```
public class MyDisticntElements {

    public static void
printDistinctElements(int[] arr){

        for(int i=0;i<arr.length;i++){
            boolean isDistinct = false;
            for(int j=0;j<i;j++){
                if(arr[i] == arr[j]){
                    isDistinct = true;
```

```
                    break;
                }
            }
            if(!isDistinct){
                System.out.print(arr[i]+"
");
            }
        }
    }

    public static void main(String a[])
{

        int[] nums = {5,2,7,2,4,7,8,2,3};
        MyDisticntElements.printDistinc
    }
}
```

Output:

Q.17) WRITE A PROGRAM TO FIND THE SUM OF THE FIRST 1000 PRIME NUMBERS.

```
public class Main {
    public static void main(String
args[]){
        int number = 2;
        int count = 0;
        long sum = 0;
        while(count < 1000){
            if(isPrimeNumber(number)){
                sum += number;
                count++;
            }
            number++;
```

```
        }
        System.out.println(sum);
    }

    private static boolean
isPrimeNumber(int number){
        for(int i=2; i<=number/2; i++){
            if(number % i == 0){
                return false;
            }
        }
        return true;
    }
}
```

Output:

Q.18) WRITE A PROGRAM TO REMOVE DUPLICATES FROM SORTED ARRAY.

```
public class MyDuplicateElements
  {
   public static int[]
removeDuplicates(int[] input)
    {

        int j = 0;
        int i = 1;
        //return if the array length is less
than 2
```

```
        if(input.length < 2)
            {
            return input;
        }

        while(i < input.length)
          {
            if(input[i] == input[j])
            {
                i++;
            }else
            {
                input[++j] = input[i++];
            }
        }
        int[] output = new int[j+1];
        for(int k=0; k<output.length;
k++){
```

```
            output[k] = input[k];
        }

        return output;
    }



public static void main(String a[]){
        int[] input1 =
 {2,3,6,6,8,9,10,10,10,12,12};
        int[] output =
 removeDuplicates(input1);
        for(int i:output){
            System.out.print(i+" ");
        }
    }
}
```

Output:

Q.19) FIND LONGEST SUBSTRING WITHOUT REPEATING CHARACTERS .

```
import java.util.HashSet;
import java.util.Set;

public class MyLongestSubstr {
```

```
    private Set<String> subStrList =
new HashSet<String>();
    private int finalSubStrSize = 0;

    public Set<String>
getLongestSubstr(String input){
        //reset instance variables
        subStrList.clear();
        finalSubStrSize = 0;
        // have a boolean flag on each
character ascii value
        boolean[] flag = new
boolean[256];
        int j = 0;
        char[] inputCharArr =
input.toCharArray();
        for (int i = 0; i <
```

```
inputCharArr.length; i++) {
            char c = inputCharArr[i];
            if (flag[c]) {
                extractSubString(inputChar
                for (int k = j; k < i; k++) {
                    if (inputCharArr[k] ==
c) {
                        j = k + 1;
                        break;
                    }
                    flag[inputCharArr[k]] =
false;
                }


} else {
                flag[c] = true;
            }
```

```
}

        extractSubString(inputCharArr,j,
        return subStrList;
    }


    private String
extractSubString(char[] inputArr, int
start, int end){

        StringBuilder sb = new
StringBuilder();
        for(int i=start;i<end;i++){
            sb.append(inputArr[i]);
        }
        String subStr = sb.toString();
```

```
        if(subStr.length() >
finalSubStrSize){
            finalSubStrSize =
subStr.length();
            subStrList.clear();
            subStrList.add(subStr);
        } else if(subStr.length() ==
finalSubStrSize){
            subStrList.add(subStr);
        }

        return sb.toString();
    }
```

```
    public static void main(String a[])
{
        MyLongestSubstr mls = new
MyLongestSubstr();
        System.out.println(mls.getLonges
        System.out.println(mls.getLonges
        System.out.println(mls.getLonges
        System.out.println(mls.getLonges
    }
}
```

Output:

[a2novice]
[uage_is]
[_jav, va_j]
[cab, abc, bca]

Q.20) HOW TO SORT A STACK USING A TEMPORARY STACK?

```
import java.util.Stack;

public class StackSort
{

    public static Stack<Integer>
sortStack(Stack<Integer> input)
    {
        Stack<Integer> tmpStack = new
Stack<Integer>();
        System.out.println("==========
debug logs ==================");
```

```
        while(!input.isEmpty())
        {
            int tmp = input.pop();
            System.out.println("Element
taken out: "+tmp);
            while(!tmpStack.isEmpty()
&& tmpStack.peek() > tmp)
            {
                input.push(tmpStack.pop());
            }
            tmpStack.push(tmp);
            System.out.println("input:
"+input);
            System.out.println("tmpStack:
"+tmpStack);
            }
        System.out.println("==========
debug logs ended
```

```
==========");
        return tmpStack;
    }

    public static void main(String a[])
    {
        Stack<Integer> input = new
Stack<Integer>();
        input.add(34);
        input.add(3);
        input.add(31);
        input.add(98);
```

```
        input.add(92);
        input.add(23);
        System.out.println("input:
"+input);
        System.out.println("final sorted
list: "+sortStack(input));
    }
}
```

Output:

```
input: [34, 3, 31, 98, 92, 23]
========== debug logs
==========
Element taken out: 23
input: [34, 3, 31, 98, 92]
tmpStack: [23]
```

Element taken out: 92
input: [34, 3, 31, 98]
tmpStack: [23, 92]
Element taken out: 98
input: [34, 3, 31]
tmpStack: [23, 92, 98]
Element taken out: 31
input: [34, 3, 98, 92]
tmpStack: [23, 31]
Element taken out: 92
input: [34, 3, 98]
tmpStack: [23, 31, 92]
Element taken out: 98
input: [34, 3]
tmpStack: [23, 31, 92, 98]
Element taken out: 3
input: [34, 98, 92, 31, 23]
tmpStack: [3]

Element taken out: 23
input: [34, 98, 92, 31]
tmpStack: [3, 23]
Element taken out: 31
input: [34, 98, 92]
tmpStack: [3, 23, 31]
Element taken out: 92
input: [34, 98]
tmpStack: [3, 23, 31, 92]
Element taken out: 98
input: [34]
tmpStack: [3, 23, 31, 92, 98]
Element taken out: 34
input: [98, 92]
tmpStack: [3, 23, 31, 34]
Element taken out: 92
input: [98]
tmpStack: [3, 23, 31, 34, 92]

Element taken out: 98
input: []
tmpStack: [3, 23, 31, 34, 92, 98]
========== debug logs ended ==========
final sorted list: [3, 23, 31, 34, 92, 98]

Thank You!