How to reverse a String in Java?

It might be surprising, but there is no reverse() utility method in the String class. But, it’s a very simple task. We can create a character array from the string and then iterate it from the end to start. We can append the characters to a string builder and finally return the reversed string.

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

            String str = "123";

            System.out.println(reverse(str));

 }


 public static String reverse(String in) {

 if (in == null)

      throw new IllegalArgumentException("Null is not valid input");

StringBuilder out = new StringBuilder();
char[] chars = in.toCharArray();

for (int i = chars.length - 1; i >= 0; i--)

          out.append(chars[i]);


 return out.toString();

}

}
```

Bonus Points: Adding null check in the method and using StringBuilder for appending the characters.

Easy to Miss: The indexing in Java starts from 0. So, we have to start at “chars.length – 1” in the for loop.

How to swap two numbers without using a third variable?

It’s a slightly tricky question. It’s a three steps process. It’s better visualized in code.

```
int a = 10;
int b = 20;


b = b + a;     // now b is sum of both

the numbers



a = b - a;     // b - a = (b + a) - a = b (a is

swapped)



b = b - a; // (b + a) - b = a (b is

swapped)
```

We can’t return multiple variables in Java. Since Java is Pass-by-Value and these are primitive data types, their values won’t change. For example, below swap function will not change the input integer values.

```
public static void swapNumbers(int a, int b) {
   b = b + a;
           a = b - a;
           b = b - a;
}

public static void main(String[] args) {
           int a = 10;
           int b = 20;
           swapNumbers(a, b);
```

```
        System.out.printf("a is %d, b is %d", a, b);
 // a is 10, b is 20
}
```

Java Program to check if a vowel is present in the string?

We can use regular expression to check if the string contains vowels or not.

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

                System.out.println(stringContainsVowels("Hello")); // true
                System.out.println(stringContainsVowels("TV")); // false

}

public static boolean stringContainsVowels(String input) {

        return input.toLowerCase().matches(".*[aeiou].*");


 }
}
```

Java program to check if the given number is Prime?

We can write a simple program to divide the given number “n” from 2 to n/2 and check the remainder. If the remainder is 0, then it’s not a prime number.

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

                            System.out.println(isPrime(19)); // true

                            System.out.println(isPrime(49)); // false

 }


 public static boolean isPrime(int n) {

      if (n == 0 || n == 1) {

            return false;     }


     if (n == 2) {

            return true;   }


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

         if (n % i == 0) {

               return false;   }  }


  return true;


}

}
```

But, this is not very memory and time-efficient. For a given number N, if there is a prime number M between 2 to √N (square root of N) that evenly divides it, then N is not a prime number.

Fibonacci Series using recursion

We can use a for loop to print fibonacci series.

```
public static void printFibonacciSeries(int count) {
        int a = 0;
        int b = 1;
        int c = 1;
        for (int i = 1; i <= count; i++) {
                System.out.print(a + ", ");
                a = b;
                b = c;
                c = a + b;
        }
}
```

The fibonacci number is generated by adding the previous two numbers – F(N) = F(N-1) + F(N-2). We can use recursion to print fibonacci series.

```
public class FibonacciNumbers {
        public static int fibonacci(int n) {
                if (n <= 1)
                        return n;
                return fibonacci(n - 1) + fibonacci(n - 2);
        }

        public static void main(String args[]) {
                int n = 10;
                System.out.println(fibonacci(n));
        }
}
```

Check if a List of integers contains only odd numbers?

We can use for loop and check each element one by one if they are odd or not.

```
public static boolean onlyOddNumbers(List<Integer> list) {
        for (int i : list) {
                if (i % 2 == 0)
                        return false;}return true;}
```

If the list is huge, we can use parallel stream for faster processing.

```
public static boolean onlyOddNumbers(List<Integer> list) {
               return list
                        .parallelStream() // parallel stream for faster processing
                        .anyMatch(x -> x % 2 != 0); // return as soon as any
elements match the condition
}
```

Palindrome Check

A palindrome string is one whose reverse is also the same string. So we can reverse the input string and check if both strings are equal or not. We can also use the String charAt(int index) method to check for palindrome string.

```
boolean checkPalindromeString(String input) {
        boolean result = true;
        int length = input.length();
        for(int i=0; i < length/2; i++) {
```

```
        if(input.charAt(i) != input.charAt(length-i-1)) {
            result = false;
            break;
        }
    }
    return result;
}
```

How to remove Whitespaces from String

We can use Character.isWhitespace() method to remove whitespaces from the string.

```
String removeWhiteSpaces(String input){
    StringBuilder output = new StringBuilder();

    char[] charArray = input.toCharArray();

    for(char c : charArray) {
      if (!Character.isWhitespace(c))
         output.append(c);

    }
  return output.toString();
}
```

How to remove leading and trailing whitespaces from a string?

Java String class contains two methods to remove leading and trailing whitespaces

– trim(), and strip(). The strip() method was added to the String class in Java 11. However, the strip() method uses Character.isWhitespace() method to check if the character is a whitespace. This method uses Unicode code points whereas the trim() method identifies any character having codepoint value less than or equal to 'U+0020' as a whitespace character.

The strip() method is the recommended way to remove whitespaces because it uses the Unicode standard.

```
String s = " abc  def\t";

s = s.strip();
System.out.println(s);
```

Since String is immutable, we have to assign the strip() output to the string.

Sorting an array in Java?

This question requires a deep understanding of sorting in Java. If you look at the Arrays utility class, there are many overloaded sort() methods to sort primitive as well as to object arrays.

If you are sorting a primitive array in the natural order, then it's very simple. Just use the Arrays.sort() method.

```
int[] array = {1, 2, 3, -1, -2, 4};

Arrays.sort(array);

System.out.println(Arrays.toString(array));
```

But, if you want to sort an array of Objects, then the object must implement Comparable interface. If you want to specify the sorting criteria, then you can pass the Comparator for the sorting logic.

How to Create a Deadlock Scenario Programatically?

Deadlock is a special scenario in the multi-threaded environment where two or more threads are blocked forever. The deadlock situation arises with at least two threads and two or more threads. Let’s write a simple program to create a deadlock.

```
public class ThreadDeadlock {
   public static void main(String[] args) throws InterruptedException {
     Object obj1 = new Object();
     Object obj2 = new Object();
     Object obj3 = new Object();

     Thread t1 = new Thread(new SyncThread(obj1, obj2), "t1");
      Thread t2 = new Thread(new SyncThread(obj2, obj3), "t2");
     Thread t3 = new Thread(new SyncThread(obj3, obj1), "t3");

     t1.start();
     Thread.sleep(5000);
     t2.start();
     Thread.sleep(5000);
     t3.start();

   }
}
```

```
class SyncThread implements Runnable{
    private Object obj1;
    private Object obj2;

    public SyncThread(Object o1, Object o2){
      this.obj1=o1;
      this.obj2=o2;
    }
    @Override
    public void run() {
      String name = Thread.currentThread().getName();
      System.out.println(name + " acquiring lock on "+obj1);

      synchronized (obj1) {
       System.out.println(name + " acquired lock on "+obj1);
       work();
       }

       System.out.println(name + " acquiring lock on "+obj2);
       synchronized (obj2) {
         System.out.println(name + " acquired lock on "+obj2);
         work();
        }

       System.out.println(name + " released lock on "+obj2);

       }

      System.out.println(name + " released lock on "+obj1);
      System.out.println(name + " finished execution.");
    }
```

```
   private void work() {
     try {
        Thread.sleep(30000);
     }

     catch (InterruptedException e) {
        e.printStackTrace(); }

}}
```

All the three threads will be able to acquire a lock on the first object. But, they are using the shared resources and started in such a way that they will keep on waiting indefinitely to acquire the lock on the second object. We can use the java thread dump to detect the deadlocks.

Find factorial of an integer?

The factorial of an integer is calculated by multiplying all the numbers from 1 to the

given number. F(n) = F(1)*F(2)…F(n-1)*F(n).

We can use recursion to find the factorial of an integer.

```
public static long factorial(long n) {
          if (n == 1)
                    return 1;
          else
                return (n * factorial(n - 1));
}
```

Revese a Linked List?

LinkedList descendingIterator() returns an iterator that iterates over the element in the reverse order. We can use this iterator to create a new Linked List with elements in the reverse order.

```
LinkedList<Integer> ll = new LinkedList<>();


ll.add(1);
ll.add(2);
ll.add(3);
System.out.println(ll);


LinkedList<Integer> ll1 = new LinkedList<>();


ll.descendingIterator().forEachRemaining(ll1::add);
System.out.println(ll1);
```

How to implement Binary Search?

The array elements must be sorted for implementing binary search. The binary search algorithm is based on the following conditions.

If the key is less than the middle element, then we now need to search only in the first half of the array. If the key is greater than the middle element, then we need to only search in the second half of the array.
And if the key is equal to the middle element in the array, then the search ends.
Finally, if the key is not found in the whole array, then it should return -1. This indicates that the element is not present.

```java
public static int binarySearch(int arr[], int low, int high, int key) {
        int mid = (low + high) / 2;

        while (low <= high) {
                if (arr[mid] < key)
                        low = mid + 1;

                else if (arr[mid] == key)
                              return mid;

                else
                    high = mid - 1;

                mid = (low + high) / 2;
         }

        if (low > high) {
                return -1;
        }
        return -1;
 }
```

Merge Sort in Java?

Merge sort is one of the most efficient sorting algorithms. It works on the principle of Divide and Conquers. It is based on the idea of breaking down a list into several sub-lists until each sublist consists of a single element. Then merging those sublists in a manner that results in a sorted list.

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

        int[] arr = { 70, 50, 30, 10, 20, 40, 60 };

        int[] merged = mergeSort(arr, 0, arr.length - 1);


        for (int val : merged) {
            System.out.print(val + " ");
        }
    }


    public static int[] mergeTwoSortedArrays(int[] one, int[] two) {


        int[] sorted = new int[one.length + two.length];
        int i = 0;
        int j = 0;
        int k = 0;


        while (i < one.length && j < two.length) {


            if (one[i] < two[j]) {
                sorted[k] = one[i];
                k++;
                i++;
            } else {
                sorted[k] = two[j];
                k++;
                j++;
            }
        }
        if (i == one.length) {
            while (j < two.length) {
                sorted[k] = two[j];
                k++;
                j++;
            }
        }

        if (j == two.length) {
            while (i < one.length) {
                sorted[k] = one[i];
                k++;
                i++;
            }
        }
        return sorted;
    }

    public static int[] mergeSort(int[] arr, int lo, int hi) {

        if (lo == hi) {
            int[] br = new int[1];
            br[0] = arr[lo];

            return br;
        }

        int mid = (lo + hi) / 2;
        int[] fh = mergeSort(arr, lo, mid);
        int[] sh = mergeSort(arr, mid + 1, hi);
        int[] merged = mergeTwoSortedArrays(fh, sh);

        return merged;
    }
}
```

Create a Pyramid of Characters in Java?

Pattern programs are used a lot in interviews to understand the logical thinking abilities of the interviewee. Pyramid patterns are very popular and once we get the logic on the way it’s created, writing code to achieve the same is an easy task.

Check if two arrays contains same elements?

We will first create a set of elements from both the arrays. Then compare the elements in these sets to find if there is an element that is not present in both the sets?

```
package com.ankit;
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

public class ArraySameElements {
        public static void main(String[] args) {
                Integer[] a1 = {1,2,3,2,1};
                Integer[] a2 = {1,2,3};

                Integer[] a3 = {1,2,3,4};
                System.out.println(sameElements(a1, a2));
                System.out.println(sameElements(a1, a3));


        }
        static boolean sameElements(Object[] array1, Object[] array2) {
                Set<Object> uniqueElements1 = new
HashSet<>(Arrays.asList(array1));
```

```
                    Set<Object> uniqueElements2 = new
HashSet<>(Arrays.asList(array2));


                    // if size is different, means there will be a mismatch
                    if(uniqueElements1.size() != uniqueElements2.size()) return false;


                    for(Object obj : uniqueElements1) {
                              // element not present in both?
                              if (!uniqueElements2.contains(obj)) return false;

                    }
                    return true;
          }
}
```

Sum of all elements in integer array?

It’s a very simple program. We can use for loop to iterate over the array elements and add them to get the final sum.

```
int[] array = { 1, 2, 3,
4, 5 }; int sum = 0;
for (int i : array)
          sum += i;

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

Find second largest number in an array?

There are many ways to solve this problem. We can sort the array in natural ascending order and take the second last value. But, sorting is an expensive operation.

We can also use two variables to find the second largest value in a single iteration.

```
private static int findSecondHighest(int[] array) {
        int highest = Integer.MIN_VALUE;
        int secondHighest = Integer.MIN_VALUE;

        for (int i : array) {
                if (i > highest) {
                        secondHighest = highest;
                        highest = i;
                }

                else if (i > secondHighest) {
                        secondHighest = i;
                }
        }
        return secondHighest;
}
```

How to Shuffle an Array in Java?

We can use Random class to generate random index numbers and shuffle the elements.

```
int[] array = { 1, 2, 3, 4, 5, 6, 7 };


Random rand = new

Random();
```

```
for (int i = 0; i < array.length;
i++)
{
        int randomIndexToSwap = rand.nextInt(array.length);
        int temp = array[randomIndexToSwap];
        array[randomIndexToSwap] = array[i];
        array[i] = temp;
}
System.out.println(Arrays.toString(array));
```

We can run the shuffling code inside another for loop to shuffle multiple rounds.

How to find if a string is present in a text file?

We can use Scanner class to read the file contents line by line. Then use String contains() method to check if the string is present in the file or not.

```
boolean findStringInFile(String filePath, String str) throws FileNotFoundException {
        File file = new File(filePath);
        Scanner scanner = new Scanner(file);

        // read the file line by line
        while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                if (line.contains(str)) {
                        scanner.close();
                        return true;
                }
        }
        scanner.close();
        return false;
}
```

The above code assumes that the string we are searching for in the file doesn’t contain newline characters.

How to print date in specific format?

We can use SimpleDateFormat class to get the date string into specific formatting.

```
String pattern = "MM-dd-yyyy";
SimpleDateFormat simpleDateFormat = new
SimpleDateFormat(pattern); String date =
simpleDateFormat.format(new Date());
System.out.println(date); // 06-23-2020
```

How to merge two lists in java?

We can use the addAll() method to merge multiple lists in Java.

```
List<String> list1 = new
ArrayList<>(); list1.add("1");
List<String> list2 = new
ArrayList<>(); list2.add("2");

List<String> mergedList = new
```

```
ArrayList<>(list1);

mergedList.addAll(list2);

System.out.println(mergedList); // [1, 2]
```

How to Sort HashMap by values?

HashMap is not an ordered collection. So, sorting its entries doesn’t make any sense. But, we can sort the entries based on value and store into LinkedHashMap. LinkedHashMap maintains the order of insertion.

```
package com.journaldev.programminginterviews;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Set;


public class SortHashMapByValue {
            public static void main(String[] args) {

                        Map<String, Integer> scores = new HashMap<>();
                        scores.put("David", 95);

                        scores.put("Jane", 80);

                        scores.put("Mary", 97);

                        scores.put("Lisa", 78);

                        scores.put("Dino", 65);

           System.out.println(scores);

           scores = sortByValue(scores);
```

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

 private static Map<String, Integer> sortByValue(Map<String, Integer> scores) {
        Map<String, Integer> sortedByValue = new LinkedHashMap<>();

        // get the entry set
        Set<Entry<String, Integer>> entrySet = scores.entrySet();
        System.out.println(entrySet);

        // create a list since the set is unordered
        List<Entry<String, Integer>> entryList = new ArrayList<>(entrySet);
        System.out.println(entryList);

        // sort the list by value
        entryList.sort((x, y) -> x.getValue().compareTo(y.getValue()));
        System.out.println(entryList);

        // populate the new hash map
        for (Entry<String, Integer> e : entryList)
            sortedByValue.put(e.getKey(), e.getValue());

        return sortedByValue;
    }
}
```

Remove all occurrences of a given character from input String?

String class doesn’t have any method to remove characters. We can use the replace() method to create a new string without the given character.

```
String str1 = "abcdABCDabcdABCD";

str1 = str1.replace("a", "");

System.out.println(str1); // bcdABCDbcdABCD
```

The string is immutable in Java. All the string manipulation methods return a new string. So, it’s necessary that we assign it to another variable.

How to get distinct characters and their count in a String?

We can create the character array from the string. Then iterate over it and create a HashMap with the character as key and their count as value.

```
String str1 =
"abcdABCDabcd"; char[]
chars = str1.toCharArray();
Map<Character, Integer> charsCount = new
HashMap<>();


for(char c : chars) {
          if(charsCount.containsKey(c)) {
                    charsCount.put(c, charsCount.get(c)+1);
          }
          else
               charsCount.put(c, 1);
}
```

```
System.out.println(charsCount); // {a=2, A=1, b=2, B=1, c=2, C=1, d=2, D=1}
```

How to prove String is immutable programatically?

```
String s1 = "Java"; // "Java" String created in pool and reference assigned to s1
String s2 = s1; //s2 is also having the same reference to "Java" in the pool
System.out.println(s1 == s2); // proof that s1 and s2 have same reference
s1 = "Python"; //s1 value got changed above, so how String is immutable?



//well, in the above case a new String "Python" got created in the pool//s1 is now
referring to the new String in the pool //BUT, the original String "Java" is still
unchanged and remains in the pool//s2 is still referring to the original String "Java" in
the pool

// proof that s1 and s2 have different

reference System.out.println(s1 ==

s2);


System.out.println(s2); // prints "Java" supporting the fact that original String
value is unchanged, hence String is immutable
```

Write a Program to showcase inheritance?

```
class Animal {
          String color;
}


class Cat extends Animal {
          void meuw(){
                    System.out.println("Meuw");
          }
}
```

Write a Program to Show Diamond Problem with Multiple Inheritance?

Java doesn’t allow extending multiple classes. It’s to keep it simple and avoid diamond problem.

```
interface I {
        void foo();
}

class A implements I{
        public void foo() {}
}

class B implements I{
        public void foo() {}
}

class C extends A, B { // won't compile
        public void bar() {
                super.foo();
        }
}
```

In above example, if Java would have allowed multiple class inheritance, then which super foo() method should get called? There could be a mechanism to fix this, but Java language developers thought it’s better to keep it simple by not allowing multiple inheritance.

Write a Program to show try catch example?

Let’s look at a simple try-catch block code.

```
try {
    FileInputStream fis = new FileInputStream("test.txt");
}
catch(FileNotFoundException e) {
    e.printStackTrace();
}
```

From Java 7 onwards, We can also catch multiple exceptions in a single catch block. It’s useful when we have the same code in all the catch blocks.

```
public static void foo(int x) throws IllegalArgumentException, NullPointerException {
    // some code
}

public static void main(String[] args) {
    try {
        foo(10);
    } catch (IllegalArgumentException | NullPointerException e) {
        System.out.println(e.getMessage());
    }
}
```

Write a code to show NullPointerException

If we are calling a function on the “null”, it will throw NullPointerException.

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

    printString(null, 3);}
static void printString(String s, int count){

    for (int i = 0 ; i < count; i++) {
```

```
                    System.out.println(s.toUpperCase()); // Exception in thread "main" java.lang.NullPointerException
          }}
```

That’s why it’s better to have null check in place for early validation.

```
static void printString(String s, int count){
          if(s == null) return;
          for (int i = 0 ; i < count; i++) {
                    System.out.println(s.toUpperCase());
          }
}
```

We can also throw IllegalArgumentException based on the project requirements.

How to Create a Record in Java?

Records is a preview feature introduced in Java 14. Records allow us to create a POJO class with minimal code. It automatically generates hashCode(), equals(), getter methods, and toString() method code for the class. Records are final and implicitly extends java.lang.Record class.

```
import java.util.Map;
 public record EmpRecord(int id, String name, long salary, Map<String, String> addresses) {
}
```

How to create Text Blocks in Java?

Java 13 added text blocks as a preview feature. We can create multiline strings

using text blocks. The multiline string has to be written inside a pair of tripledouble quotes.

```
String textBlock = """
                    Hi
                    Hello
                    Yes""";
```

It’s same as creating a string as “Hi\nHello\nYes”.

Show example of switch expressions and multi-label case statements

The switch expressions were added as a preview feature in Java 12. It became a standard feature in Java 14 release. The below examples show switch expressions as well as multi-label case statements.

```
int choice = 2;
int x = switch (choice) {case 1, 2, 3:
          yield choice;default:
          yield -1;
};

System.out.println("x = " + x); // x = 2
```

We can also use lambda expressions in switch expressions.

```
String day = "TH";String result = switch (day) {case "M", "W", "F" -> "MWF";case
"T", "TH", "S" -> "TTS";default -> {
          if (day.isEmpty())
                    yield "Please insert a valid day.";
          else
                    yield "Looks like a Sunday.";
}
};
System.out.println(result); // TTH
```

How to Compile and Run a Java Class from Command Line?

Let’s say we have a class as below.

```
public class Test {
public static void main(String args[]) {
                    System.out.println("Hi");
          }
}
```

We can compile it using the following code.

```
javac Test.java
```

For running the class, we can run the following command.

```
java Test
```

From the recent releases, java command will take care of compilation also if the class file is not present.

If the class is in a package com.journaldev, then it should be inside the folder com/journaldev. The command to compile and run will be as follows.

```
java com/journaldev/Test.java
```

If our class requires some additional JARs to compile and run, we can use the -cp java option.

```
java -cp .:~/.m2/repository/log4j/log4j/1.2.17/log4j-1.2.17.jar com/journaldev/Test.java
```

How to create Enum in Java?

Let’s look at a simple Enum.

```
public enum ThreadStates {
        START,
        RUNNING,
        WAITING,
        DEAD;
}
```

ThreadStates is the enum with fixed constants fields START, RUNNING, WAITING, and DEAD.

All Enum implicitly extends java.lang.Enum class and implements Serializable and Comparable interfaces.

How to use forEach() method?

The forEach() method provides a shortcut to perform an action on all the elements of an iterable. Let’s say we have to iterate over the list elements and print it.

```
List<String> list = new ArrayList<>();


Iterator<String> it =
list.iterator(); while
(it.hasNext()) {
        System.out.println(it.next());
}
```

We can use forEach() method with lambda expression to reduce the code size.

```
List<String> list = new
ArrayList<>();
list.forEach(System.out::print);
```

Write an interface with default and static method?

Java 8 introduced default and static methods in interfaces. This has bridged the gap between interfaces and abstract classes.

```

public interface Interface1 {
        // regular abstract method

        void method1(String str);
        default void log(String str){
                System.out.println("I1 logging::"+str);

        }
        static boolean isNull(String str) {
                System.out.println("Interface Null Check");

                return str == null ? true : "".equals(str) ? true : false;
        }
}
```

How do we create a Functional interface?

An interface with exactly one abstract method is called Functional Interface. @FunctionalInterface annotation is added so that we can mark an interface as functional interface.

The major benefit of java 8 functional interfaces is that we can use lambda expressions to instantiate them and avoid using bulky anonymous class implementation.

```
@FunctionalInterfaceinterface Foo {
        void test();
}
```

Show an example of using lambda expressions in Java

Runnable is an excellent example of a functional interface. We can use lambda expressions to create a runnable.

```
Runnable r1 = () -> System.out.println("My Runnable");
```

Show examples of overloading and overriding in Java

When a class have two or more methods with the same name, they are called overloaded methods.

```
class Foo {

        void print(String s) {
                System.out.println(s);
        }

        void print(String s, int count) {
                while (count > 0) {
                        System.out.println(s);
                        count--;
                }}
}
```

When a superclass method is also implemented in the child class, it’s a case of overriding.

```
class Base {
        void printName(){
                System.out.println("Base Class");
        }
}
class Child extends Base{
        @Override
        void printName(){
                System.out.println("Child Class");
        }
}
```

Guess the Output of Code Snippets

Let’s look at 8 code snippets and guess their output.

```
String s1 = "abc";String s2 = "abc";
System.out.println("s1 == s2 is:" + s1
== s2);
```

Output: false

Explanation: The given statements output will be “false” because in java + operator precedence is more than == operator. So the given expression will be evaluated to “s1 == s2 is:abc” == “abc” i.e false.

```
String s3 =
"JournalDev"; int
start = 1;
char end = 5;
System.out.println(start +
end);
System.out.println(s3.substring(start

, end));
```

Output: ourn

Explanation: The given statements output will be “ourn”. First character will be automatically type caste to int. After that since in java first character index is 0, so it will start from ‘o’ and print till ‘n’. Note that in String substring function it leaves the end index.

.

.

```
HashSet shortSet = new HashSet();for (short i = 0; i < 100; i++) {
            shortSet.add(i);
            shortSet.remove(i - 1);
}
System.out.println(shortSet.size());
```

.

Output: 100

Explanation: The size of the shortSet will be 100. Java Autoboxing feature has been introduced in JDK 5, so while adding the short to HashSet<Short> it will automatically convert it to Short object. Now “i-1” will be converted to an int while evaluation and after that it will autoboxed to Integer object but there is no Integer object in the HashSet, so it will not remove anything from the HashSet and finally its size will be 100.

.

What will be the boolean “flag” value to reach the finally block?

```
try {
          if (flag) {
                    while (true) {
                    }
          } else {
                    System.exit(1);
          }
} finally {
          System.out.println("In Finally");
}
```

Explanation: The finally block will never be reached here. If flag will be TRUE, it will go into an infinite loop and if it’s false it’s exiting the JVM. So finally block will never be reached here.

```
String str =
null; String
str1="abc";
System.out.println(str1.equals("abc") |

str.equals(null));
```

Output:

NullPointerException

Explanation: The given print statement will throw java.lang.NullPointerException because while evaluating the OR logical operator it will first evaluate both the literals and since str is null, .equals() method will throw exception. Its always advisable to use short circuit logical operators i.e “||” and “&&” which evaluates the literals values from left and since the first literal will return true, it will skip the second literal evaluation.

```
String x = "abc";String y =
"abc"; x.concat(y);
System.out.print
(x);
```

Output: abc

Explanation: The x.concat(y) will create a new string but it’s not assigned to x, so the value of x is not changed.

```
public class MathTest {

        public void main(String[] args) {

                int x = 10*10-10;
                System.out.println(x);
        }
}
```

Output: Runtime error

Explanation: This is a tricky question, it looks like the test is about the order of execution of the mathematical operators and syntax of main method will get overlooked. It will produce Runtime error because main method is not static, something like below.

.

```
pankaj:bin pankaj$ java MathTest
Error: Main method is not static in class MathTest, please define the main method as:
```

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

```
public class Test {

	public static void main(String[] args) {
		try {
			throw new IOException("Hello");
		}catch(IOException | Exception e) {
			System.out.println(e.getMessage());
		}
	}
}
```

.

Output: Compile-Time Error

Explanation: It will be a compile time error as The exception IOException is already caught by the alternative Exception.

Find 5 mistakes in the following code snippet

```
package com.journaldev.programming-
interviews; public class String Programs {

	static void main(String[10] args) {
		String s = "abc"
		System.out.println(s);
	}
}
```

. Package name can’t have hyphens.

. Class name can’t have whitespaces.
. The main method is not public, so it won’t run.
. The main method argument shouldn’t specify the size.
. The semicolon is missing in the string definition.

Write a Java Program to remove all white spaces from a string with using replace().

This is a simple program where we have our string variable str1.

Another string variable str2 is initialized with the replaceAll option which is an inbuilt method to remove n number of whitespaces. Ultimately, we have printed str2 which has no whitespaces.

```
class RemoveWhiteSpaces
{
   public static void main(String[] args)
   {
      String str1 = "Saket Saurav                    is a QualityAna    list";

      //1. Using replaceAll()

      Method String str2 =

      str1.replaceAll("\\s", "");

      System.out.println(str2);

        }
}
}
```

Output:

```
SaketSauravisaQualityAnalist
```

Write a Java Program to remove all white spaces from a string without using replace().

This is another approach to removing all the white spaces. Again, we have one string variable str1 with some value. Then, we have converted that string into a character array using toCharArray().

Then, we have one StringBuffer object sb which will be used to append the value stored at chars[i] index after we have included for loop and one if condition.

If the condition is set such that then the element at i index of the character array should not be equal to space or tab. Finally, we have printed our StringBuffer object sb.

```
class RemoveWhiteSpaces
{
  public static void main(String[] args){
    String str1 = "Saket Saurav          is an Autom ation Engi ne          er";

    char[] chars =

    str1.toCharArray();

    StringBuffer sb = new

    StringBuffer();

    for (int i = 0; i &lt; chars.length; i++)
    {
      if( (chars[i] != ' ') &amp;&amp; (chars[i] != '\t') )
      {
        sb.append(chars[i]);
      }
    }


    System.out.println(sb);   //Output : CoreJavajspservletsjdbcstrutshibernatespring

  }
}
```

Output:

```
SaketSauravisanAutomationEngineer
```

Write a Java Program to find the second–highest number in an array.

In this program, we have initialized an array with 10 random elements out of which we are going to find the second-highest number. Here, we have two integers- the largest and second-largest. Both set to the first index of the element. Then, we have printed all the elements using for loop.

Now, the logic is when the element at the 0th index is greater than the largest, then assign arr[0] to largest and secondLargest to largest. Again, if the element at the 0th index is greater than the secondLargest, then assign secondLargest to arr[0].

This will be repeated for each iteration and ultimately after comparing or completing iterations up to array length will give you the secondLargest element.

```
package codes;
public class SecondHighestNumberInArray {
public static void main(String[] args)
  {
    int arr[] = { 100,14, 46, 47, 94, 94, 52, 86, 36, 94, 89 };
    int largest = 0;
    int secondLargest = 0;
    System.out.println("The given
    array is:"); for (int i = 0; i <
    arr.length; i++)
    {
      System.out.print(arr[i] + "\t");
    }

    for (int i = 0; i < arr.length; i++)
    {
      if (arr[i] > largest)
      {
        secondLargest =
        largest; largest = arr[i];
      }
      else if (arr[i] > secondLargest)
      {
        secondLargest = arr[i];
      }
    }

    System.out.println("\nSecond largest number is:" +
    secondLargest);

    System.out.println("Largest Number is: " +largest);

  }
}
```

Output:

The given array is:

```
100 14 46 47 94 94 52 86 36 94 89
Second largest number
is:94 Largest Number
is: 100
```

Write a Java Program to check Armstrong number.

First of all we need to understand what Armstrong Number is. Armstrong number is the number which is the sum of the cubes of all its unit, tens and hundred digits for three-digit numbers.

153 = 1*1*1 + 5*5*5 + 3*3*3 = 1 + 125 + 27 = 153

If you have a four-digit number lets say

1634 = 1*1*1*1 + 6*6*6*6 + 3*3*3*3 + 4*4*4*4 = 1 + 1296 + 81 + 256 = 1634

Now, in this program, we have a temp and integers declared. We have initialized c with value 0. Then, we need to assign the integer value which we are going to check for Armstrong (in our case, let us say 153). Then we have assigned our temp variable with that number which we are going to check.

Thereafter, we have used while conditional check where the remainder is assigned to a and the number is divided by 10 and assigned to n. Now, our c variable which was set to zero initially is assigned with c+(a*a*a). Suppose we have to evaluate a four-digit number then c should be assigned with c + (a*a*a*a).

Lastly, we have put an if-else statement for conditional checking where we have compared the value contained in c against temp(which has the actual number stored at this point). If it matches, then the number is Armstrong otherwise not.

```
class Armstrong{
 public static void main(String[] args) {
  int c=0,a,temp;

  int n=153;//It is the number to check
  Armstrong temp=n;
  while(n&gt;0)
  {
  a=n%1
  0;
  n=n/10;
   c=c+(a*a*
   a);
   }
   if(temp==c)
   System.out.println("armstrong
   number"); else
      System.out.println("Not armstrong number");
  }
}
```

Output:

```
armstrong number
```

Write a Java Program to find the duplicate characters in a string.

In this program, we have created a string variable str and initialized an integer count with zero.

Then, we have created a character array to convert our string variable to the character. With the help of for loop, we are performing a comparison between different characters at different indexes.

If two characters of consecutive index match, then it will print that character and the counter will be incremented by 1 after each iteration.

```
public class DuplicateCharacters {

        public static void main(String[] args) {
                // TODO Auto-generated
               method stub String str = new
               String("Sakkett");
               int count = 0;
               char[] chars = str.toCharArray();
               System.out.println("Duplicate
               characters are:"); for (int i=0;
               i&lt;str.length();i++) {
                          for(int j=i+1; j&lt;str.length();j++) {
                                   if (chars[i] == chars[j]) {
                                           System.out.println(cha
                                           rs[j]); count++;
                                           break;
                                   }}}}}
```

Output:

```
Duplicate characters
are: k
```

t

Write a Java Program to iterate ArrayList using for–loop, while–loop, and advance for–loop.

In this program, we have inserted three elements and printed the size of the ArrayList.

Then, we have used While Loop with an iterator. Whenever the iterator has (next) element, it will display that element until we reach the end of the list. So it will iterate three times.

Likewise, we have done for Advanced For Loop where we have created an object called obj for the ArrayList called list. Then printed the object.

Thereafter, we have put the condition of For Loop where the iterator i is set to 0 index, then it is incremented by 1 until the ArrayList limit or size is reached. Finally, we have printed each element using a get(index) method for each iteration of For Loop.

```
import java.util.*;

public class

arrayList {
   public static void main(String[]
      args) { ArrayList list = new
      ArrayList(); list.add("20");
      list.add("30");
      list.add("40");
      System.out.println(list.size()
      );
       System.out.println("While
      Loop:");

      Iterator itr = list.iterator();

      while(itr.hasNext()) {
         System.out.println(itr.next());
      }

      System.out.println("Advanced For
      Loop:");

       for(Object obj : list) {
         System.out.println(obj);
        }
      System.out.println("For
      Loop:");
       for(int i=0; i&lt;list.size();
      i++) {
         System.out.println(list.get(i));
      }
}
}
```

Output:

```
3
While Loop:
20
30
40
Advanced For
Loop: 20

30
40
For Loop:
20
30
40
```

Write a Java Program to find whether a string or number is palindrome or not.

You can use any of the reverse string program explained above to check whether the number or string is palindrome or not.

What you need to do is to include one if-else statement. If the original string is equal to a reversed string then the number is a palindrome, otherwise not.

```
import java.util.Scanner;

public class Palindrome {
    public static void main (String[]
        args) { String original, reverse
        = "";
        Scanner in = new Scanner(System.in);
        int length;
        System.out.println("Enter the number or
        String"); original = in.nextLine();
        length = original.length();
        for (int i =length -1; i>;=0; i--) {
          reverse = reverse + original.charAt(i);
        }
        System.out.println("reverse is:" +reverse);

        if(original.equals(reverse))
          System.out.println("The number is
          palindrome");
        else
          System.out.println("The number is not a palindrome");

    }
}
```

Output:

For String-

```
Enter the number or
String vijay
reverse is:yajiv
The number is not a palindrome
```

For Number-

```
Enter the number or
String 99
reverse is:99
The number is palindrome
```

Write a Java Program to iterate HashMap using While and advance for loop.

Here we have inserted three elements in HashMap using put() function.

The size of the map can get using the size() method. Thereafter, we have used a While loop for iterating through the map which contains one key-value pair for each element. Keys and Values can be retrieved through getKey() and getValue().

Likewise, we have used advanced for loop where we have a “me2” object for the HashMap.

```
    import
    java.util.HashMap;
    import
    java.util.Iterator;
    import java.util.Map;

    public class HashMapIteration {

public static void main(String[] args)
                                        {
 // TODO Auto-generated method stub
        HashMap&lt;Integer,String&gt; map = new
        HashMap&lt;Integer,String&gt;(); map.put(2, "Saket");
        map.put(25, "Saurav");
        map.put(12, "HashMap");
        System.out.println(map.size());
        System.out.println("While
        Loop:"); Iterator itr =
        map.entrySet().iterator();

        while(itr.hasNext()) {
          Map.Entry me = (Map.Entry) itr.next();
          System.out.println("Key is " + me.getKey() + " Value is " + me.getValue());
        }

        System.out.println("For Loop:");

        for(Map.Entry me2: map.entrySet()) {
          System.out.println("Key is: " + me2.getKey() + " Value is: " + me2.getValue());
        }
      }
    }
```

Output:

```
3
While Loop:
Key is 2 Value is
Saket Key is 25
Value is Saurav
Key is 12 Value is
HashMap For Loop:
Key is: 2 Value is:
Saket Key is: 25
Value is: Saurav
Key is: 12 Value is: HashMap
```

Write a Java Program to count the number of words in a string using HashMap.

This is a collection class program where we have used HashMap for storing the string.

First of all, we have declared our string variable called str. Then we have used split() function delimited by single space so that we can split multiple words in a string.

Thereafter, we have declared HashMap and iterated using for loop. Inside for loop, we have an if-else statement in which wherever at a particular position, the map contains a key, we set the counter at that position and add the object to the map.

Each time, the counter is incremented by 1. Else, the

counter is set to 1. Finally, we are printing the HashMap.

Note: The same program can be used to count the number of characters in a string. All you need to do is to remove one space (remove space delimited in split method) in String[] split = str.split(“”);

```
    import java.util.HashMap;

    public class FinalCountWords {

public static void main(String[] args)
                                               {
 // TODO Auto-generated method stub
        String  str  =  "This  this  is  is  done  by
        Saket Saket"; String[] split = str.split(" ");

            HashMap<String,Integer> map = new HashMap<String,Integer>();
        for (int i=0; i<split.length; i++) {
          if
            (map.containsKey(split
            [i]))   {   int   count   =
            map.get(split[i]);
            map.put(split[i],
            count+1);
          }
          else {
            map.put(split[i], 1);
          }
        }
        System.out.println(map);
      }
}
```

Output:

```
{Saket=2, by=1, this=1, This=1, is=2, done=1}
```

Write a Java Program to swap two numbers without using the third variable.

Rest all things will be the same as the above program. Only the logic will change. Here, we are assigning x with the value x + y which means x will have a sum of both x and y.

Then, we are assigning y with the value x – y which means we are subtracting the value of y from the sum of (x + y). Till here, x still has the sum of both x and y. But y has the value of x.

Finally, in the third step, we are assigning x with the value x – y which means we are subtracting y (which has the value of x) from the total (x + y). This will assign x with the value of y and vice versa.

```
import java.util.Scanner;

class SwapTwoNumberWithoutThirdVariable
{
  public static void main(String args[])
  {
    int x, y;
    System.out.println("Enter x and
    y"); Scanner in = new
    Scanner(System.in);

    x =
    in.nextInt();
    y =
    in.nextInt();

    System.out.println("Before Swapping\nx =

    "+x+"\ny = "+y); x = x + y;
    y = x - y;
    x = x - y;


    System.out.println("After Swapping without third variable\nx = "+x+"\ny = "+y);
  }
}
```

Output:

```
Enter x
and y 45
98
Before
Swapping x =
45
y = 98
After Swapping without a third
variable x = 98
y = 45
```

Write a Java Program to reverse a string without using String inbuilt function reverse().

There are several ways with which you can reverse your string if you are allowed to use the other string inbuilt functions.

In this method, we are initializing a string variable called str with the value of your given string. Then, we are converting that string into a character array with the toCharArray() function. Thereafter, we are using for loop to iterate between each character in reverse order and printing each character.

```
public class FinalReverseWithoutUsingInbuiltFunction {
   public static void main(String[]
      args) { String str = "Saket
      Saurav";
      char chars[] = str.toCharArray(); // converted to character array and printed in
      reverse order
      for(int i= chars.length-1;
         i&gt;=0; i--) {
         System.out.print(chars[i]);
      }
   }
}
```

Output:

varuaS tekaS

Write a Java program by using BufferedReader class to prompt a user to input his/her name and then the output will be shown as an example below:

Hello Dara!

```
import java.io.*;
public class JavaExercises
{
public static void main(String[] args)
{
printName();
}

static void
printName(){ String
pname=null;

try{
  BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
  System.out.print("Enter your name:");
  pname=br.readLine();
  }

catch(IOException e){}

System.out.println("Hello "+pname);
}
}
```

Write Java program to allow the user to input two integer values and then the program prints the results of adding, subtracting, multiplying, and dividing among the two values. See the example below:

Enter value

a:30 Enter

```
value b:10

The result of adding is 40.

The result of subtracting is

20; The result of

multiplying is 300. The

result of dividing is 3.
```

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

static void

caculateValues(){ int

a,b;
int
resulta,results,resultm;
float resultd;
Scanner sc=new
Scanner(System.in);
System.out.print("Enter a:");
a=sc.nextInt();
System.out.print("Enter b:");
b=sc.nextInt();
resulta=a+b;
results=a-b;
resultm=a*b;
resultd=(float)a
/b;
System.out.println("The result of adding is "+resulta);
System.out.println("The result of subtracting is
"+results); System.out.println("The result of multiplying
is "+resultm); System.out.println("The result of dividing
is "+resultd);

 }

}
```

Write Java program to generate a random number between 1 to 6.
To generate a random number, you can use the Random class of java.util package. You may use the abs() method of Math class to make sure you can get only a positive number.

```
import java.util.*;
public class JavaExercises
{
public static void main(String[] args)
{
caculateValues();
}
static void caculateValues(){
int a;
Random rn=new Random();
a=1+Math.abs(rn.nextInt()%6);
System.out.println("The result: "+a);

}}
```

Write Java program to allow the user to input two float values and then the program adds the two values together. The result will be assigned to the first variable.

```
Enter value a:12.5

The value of a before adding is

12.5. Enter value b:34.9

The value of a after adding is 47.4.
```

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

```
public class JavaExercises
{
public static void main(String[] args)
{
 caculateValues();
}

static void caculateValues(){

 float
 a;
 float
 b;
 Scanner sc=new Scanner(System.in);
 System.out.print("Enter a:");
 a=sc.nextFloat();
 System.out.println("The value of a before
 adding:"+a); System.out.print("Enter b:");
 b=sc.nextFloat
 (); a+=b;
 System.out.println("The value of a after adding:"+a);
}}
```

Write Java program to allow the user to input the amount of deposit, yearly interest rate (percentage), and income tax(percentage). Then the program will calculate the amount of interest that the person earns in the year. See the example output below:

```
The amount of deposit:

1000 Yearly interest rate:

7.5% Income tax rate: 4%

The amount of interest earned in the year:71.0
```

```
import java.util.*;
public class JavaExercises
{
public static void main(String[] args)
{
 caculateInterest();
}

static void caculateInterest(){
 float amount_dep, rate, tax, interest_earned,
 tax_amount; Scanner sc=new
 Scanner(System.in); System.out.print("Enter
 the amount of deposit:");
 amount_dep=sc.nextFloat();
 System.out.print("Enter yearly interest
 rate:"); rate=sc.nextFloat();
 interest_earned=amount_dep*(rate/100); //amount of interest before tax
 calculation System.out.print("Enter income tax rate:");
 tax=sc.nextFloat();
 tax_amount=interest_earned*(tax/100);
 interest_earned-=tax; //the final interest
 earned
 System.out.println("The interest earned in the year:"+interest_earned);


}
}
```

Write Java program to allow the user to input his/her age. Then the program will show if the person is eligible to vote. A person who is eligible to vote must be older than or equal to 18 years old.

```
Enter your age: 18

You are eligible to vote.
```

```
import java.util.*;
public class JavaExercises
{
public static void main(String[] args)
{
checkEligibility();
}

static void
```

```
checkEligibility(){ int

age;
Scanner sc=new
Scanner(System.in);
System.out.print("What is your
age?"); age=sc.nextInt();
if(age>=18)
System.out.println("You are eligible to vote.");
else
System.out.println("You are not eligible to vote.");


}
}
```

Write a Java program that determines a student’s grade.

The program will read three types of scores(quiz, mid-term, and final scores) and determine the grade based on the following rules:
-if the average score >=90% =>grade=A
-if the average score >= 70% and <90% => grade=B
-if the average score>=50% and <70% =>grade=C
-if the average score<50%

=>grade=F See the example

output below:

Quiz score: 80

Mid-term score: 68

Final score:

90 Your

grade is B.

```
import java.util.*;
public class JavaExercises
{
public static void main(String[] args)
{
showGrade();
}

static void showGrade(){

float quiz_score,
mid_score,final_score,avg; Scanner
sc=new Scanner(System.in);
System.out.print("Quiz score:");
quiz_score=sc.nextFloat();
System.out.print("Mid-term
score:"); mid_score=sc.nextFloat();
System.out.print("Final score:");
final_score=sc.nextFloat();
avg=(quiz_score+mid_score+final_sc
ore)/3;

if(avg>=90) System.out.println("Your grade A.");
else if((avg>=70) && (avg<90))
System.out.println("Your grade B."); else if((avg>=50)
&& (avg<70)) System.out.println("Your grade C."); else
if(avg<50) System.out.println("Your grade F.");
else System.out.println("Invalid");

 }
}
```

Write a Java program to calculate the revenue from a sale based on the unit price and quantity of a product input by the user.

The discount rate is 10% for the quantity purchased between 100 and 120 units, and 15% for the quantity purchased greater than 120 units. If the quantity purchased is less than 100 units, the discount rate is 0%. See the example output as shown below:

```
Enter unit price: 25

Enter quantity: 110

The revenue from sale:

2475.0$ After discount:

275.0$(10.0%)
```

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

```
public class JavaExercises
{
public static void main(String[] args)
{
calculateSale();
}

static void

calculateSale(){ float

unitprice=0f;
int quantity=0;
float
revenue=0f;
float discount_rate=0f, discount_amount=0f;

Scanner sc=new Scanner(System.in);
System.out.print("Enter unit price:");
unitprice=sc.nextFloat();
System.out.print("Enter quantity:");
quantity=sc.nextInt();

if(quantity<100)
revenue=unitprice*quantit
y;
else if(quantity>=100 && quantity<=120)
{
discount_rate=(float)10/100;
revenue=unitprice*quantity;
discount_amount=revenue*discoun
t_rate; revenue-
=discount_amount;
}

else if(quantity>120)
{
discount_rate=(float)15/100;
revenue=unitprice*quantity;
discount_amount=revenue*discoun
t_rate; revenue-
=discount_amount;
}


System.out.println("The revenue from sale:"+revenue+"$");
System.out.println("After
discount:"+discount_amount+"$("+discount_rate*100+"%)");

 }
}
```

Write a Java program to detect key presses.

If the user pressed number keys(from 0 to 9), the program will tell the number that is pressed, otherwise, program will show "Not allowed".

```
import java.io.*;
public class JavaExercises
{
public static void main(String[] args)
{
  detectKey();
}

static void detectKey(){

char key=' ';
System.out.print("Press a number
key:"); try{
key = (char)System.in.read();
}catch(IOException
e){}; switch (key)
{
case '0': System.out.println("You pressed 0."); break;
case '1': System.out.println("You pressed 1."); break;
case '2': System.out.println("You pressed 2."); break;
case '3': System.out.println("You pressed 3."); break;
case '4': System.out.println("You pressed 4."); break;
case '5': System.out.println("You pressed 5."); break;
case '6': System.out.println("You pressed 6."); break;
case '7': System.out.println("You pressed 7."); break;
case '8': System.out.println("You pressed 8."); break;
case '9': System.out.println("You pressed
9."); break; default: System.out.println("Not
allowed!"); break;

}
}
}
```

Write a Java program that allows the user to choose the correct answer of a question. See the example below:

What is the correct way to declare a variable to store an integer value in Java?
a. int 1x=10;
b. int x=10;
c. float x=10.0f;
d. string x="10";
Enter your
choice: c

```
import java.io.*;

public class JavaExercises
{
public static void main(String[] args)
{
  selectChoice();
}

static void
selectChoice(){ char
ans=' ';
System.out.println("What is the correct way to declare a variable to store an
integer value in Java?");
System.out.println("a. int 1x=10");
System.out.println("b. int x=10");
System.out.println("c. float
x=10.0f"); System.out.println("d.
string x=\"10\"");
System.out.print("Enter your
choice:"); try{
ans = (char)System.in.read();
}

catch(IOException
e){}; switch (ans)
{
case 'a': System.out.println("Invalid
choice!"); break; case 'b':
System.out.println("Congratulation!"); break;
case 'c': System.out.println("Invalid
choice!"); break; case 'd':
System.out.println("Invalid choice!"); break;


default: System.out.println("Bad choice!");break;

}}}
```

Write a Java program by using two for loops to produce the output shown below:

```
*******

******

*****

****

***

**

*
```

```
public class JavaExercises
{
public static void main(String[] args)
{
printStars();
}

static void printStars(){

int i,j;
for(i=0;i<=6;i+
+){
for(j=1;j<=7-i;j++)
System.out.print("*");
System.out.println("");

}
  }
}
```

Write a Java program by using three for loops to print the following pattern:

```
1******

12*****
123****
1234***
12345**
123456*
1234567
```

```
public class JavaExercises
{
public static void main(String[] args)
{
  printPattern();
}

static void
printPattern(){ int
i,j,k;

for (i = 1; i <= 7; i++)
{
  for (j = 1; j <= i;
     ++j)
     System.out.print(
     j);

 for (k = 7 - i; k >= 1;
              k--)
 System.out.print("*");

  System.out.println("");


}
 }


}
```

Write Java program to prompt the user to choose the correct answer from a list of answer choices of a question.

The user can choose to continue answering the question or stop answering it. See the example below:

What is the command keyword to exit a loop in Java?

a. int

b. continue

c. break

d. exit

Enter your

choice: b

Incorrect!

Again? press y to continue:

```
import java.io.*;

public class JavaExercises
{
public static void main(String[] args)
{
selectChoice();
}

static void selectChoice(){

String choice;
String
con="y"; try{
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in)); System.out.println("What is the
command keyword to exit a loop in Java?");
System.out.println("a.quit");
System.out.println("b.continu
e");
System.out.println("c.break")
; System.out.println("d.exit");


while (con.compareTo("y")==0)
{
System.out.print("Enter your
choice:"); choice =br.readLine();
if (choice.compareTo("c")==0)
{
System.out.println("Congratulation!");
}
else if (choice.compareTo("q")==0 || choice.compareTo("e")==0)
{ System.out.println("Exiting...!");
break; } else
System.out.println("Incorrect!");

System.out.print("Again? press y to
continue:"); con =br.readLine();
}
}

catch(IOException e){}
}
}
```

Write Java program to print the table of characters that are equivalent to the Ascii codes from 1 to 122.

The program will print the 10 characters per line.

```
public class JavaExercises
{
public static void main(String[] args)
{
selectCharacters();
}

static void

printCharacters(){ int i

=1;
while (i <=122)
{
System.out.print((char)i+
"\t"); if (i % 10 == 0)
System.out.print(
""); i++;


}
 }
}
```

By using do while loop, write Java program to prompt the user to choose the correct answer from a list of answer choices of a question.

The user can choose to continue answering the question or stop answering it. See the example below:

```
What is the command keyword to exit a loop in Java?

a. int

b. continue

c. break

d. exit

Enter your

choice: b

Incorrect!

Again? press y to continue:
```

```
import java.io.*;

public class JavaExercises
{
public static void main(String[] args)
{
selectChoice();
}

static void selectChoice(){
```

```
String choice;
String con;
try{
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in)); System.out.println("What is the
command keyword to exit a loop in Java?");
System.out.println("a.quit");
System.out.println("b.continu
e");
System.out.println("c.break")
; System.out.println("d.exit");

do
{

System.out.print("Enter your
choice:"); choice =br.readLine();

if (choice.compareTo("c")==0)
{
System.out.println("Congratulation!");
}
else if (choice.compareTo("q")==0 || choice.compareTo("e")==0)
{ System.out.println("Exiting...!");
break; } else
System.out.println("Incorrect!");

System.out.print("Again? press y to
continue:"); con =br.readLine();
} while (con.compareTo("y")==0);

}catch(IOException e){}


 }
}
```

By using do while loop, write Java program to print the table of characters that are

equivalent to the Ascii codes from 1 to 122.

The program will print the 10 characters per line

```
public class JavaExercises
{
public static void main(String[] args)
{
 printCharacters();
}
static void printCharacters(){

int i
=1; do
{
   System.out.print((char)i+"\t");

   if (i % 10 == 0)
     System.out.println
   (""); i++;
 }while(i<=122);

 }
}
```

By using the bubble sort algorithm, write a Java program to sort an integer array of 10 elements in ascending.

```
public class BubbleSort{

public static void main(String[] args){
//unsorted array
int[] arr={12,34,23,2,4,56,80,34,45,90};
//sorted array using
bubble sort
bubblesort(arr,arr.length);
//display the content of sorted
array int i;
for(i=0;i<arr.length;i++) System.out.println(arr[i]);

}
public static void bubblesort(int[]
dataset, int n){ int i,j;
for(i=0;i<n;i
++) for(j=n-
1;j>i;j--)
if(dataset[j]<dataset[j-1])
{
int
temp=dataset[j];
dataset[j]=dataset[
j-1]; dataset[j-
1]=temp;

}
}
}
```

Modify the Java code in exercise 1 to sort the array in descending order.

```
public class BubbleSort {

public static void main(String[] args){
//unsorted array
int[] arr={12,34,23,2,4,56,80,34,45,90};
//sorted array using
bubble sort
bubblesort(arr,arr.length);
//display the content of sorted
array int i;
for(i=0;i<arr.length;i++) System.out.println(arr[i]);

}

public static void bubblesort(int[]
dataset, int n){ int i,j;
for(i=0;i<n;i
++) for(j=n-
1;j>i;j--)
if(dataset[j]>dataset[j-1])
{
int
temp=dataset[j];
dataset[j]=dataset[
j-1]; dataset[j-
1]=temp;


}
}
}
```

By using the sequential search algorithm, write a Java program to search for an element of an integer array of 10 elements.

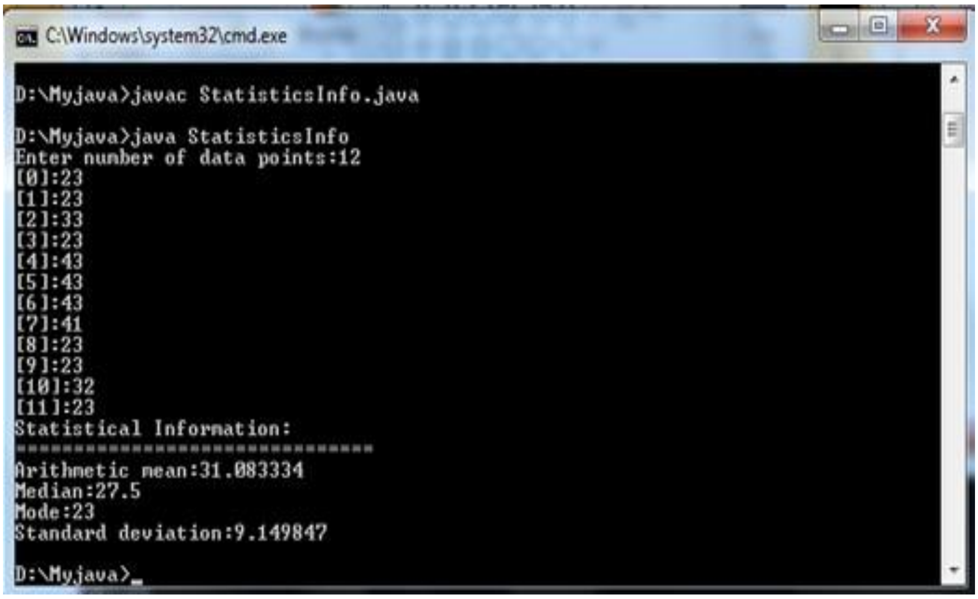
```
public class SequentialSearch{

public static void main(String[]
args){ int[]
arr={23,2,4,56,80,23,4,5,6,10};
//searching by using the sequential search
technique int pos=seqsearch(arr,56,
arr.length);
if(pos!=-1) System.out.println(" The values is found at the position of "+ pos);

}

public static int seqsearch(int[] dataset,int
target,int n){ int found=0;
int i;

int pos=-1;
for(i=0;i<n && found!=1;i++)
 if(target==dataset[i]){pos=i;found=1;
 }

return pos;
 }}
```

Modify the Java code in exercise 1 to search for an element of the array by using binary search algorithm.

```
public class BinarySearch{


public static void main(String[]
args){ int[]
arr={23,2,4,56,80,23,12,34,5,23}
;
//searching by using the binary search
technique int pos=binsearch(arr,56,
arr.length);
if(pos!=-1) System.out.println(" The values is found at the position of "+ pos);

}

public static int binsearch(int[] dataset,int
target, int n){ insertsort(dataset,n);//make sure
the list sorted
int i;
int pos=-1;
int
found=0;

if(target<dataset[n/2]){//look in the
first half for(i=0;i<n/2 &&
found!=1;i++)
if(target==dataset[i]){
found=1;pos=i;}
}
else{ //look in the second haft
for(i=n/2;i<n && found!=1;i++)
if(target==dataset[i]){
found=1;pos=i;}
}


return pos;
}

public static void insertsort(int[] dataset, int n)
{

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


int pick_item =
dataset[i]; int
inserted = 0;
for (j = i - 1; j >= 0 && inserted != 1; )
{
if (pick_item < dataset[j])
{
dataset[j + 1] =
dataset[j]; j--;
dataset[j + 1] = pick_item;
}
else inserted = 1;


}
}
```

```
}
}
```

Write a Java program to answer about the statistical information such as arithmetic mean, median, mode, and standard deviation of an integer data set. The data points are input by the user from keyboard. This program will display the output similar to the one shown below:



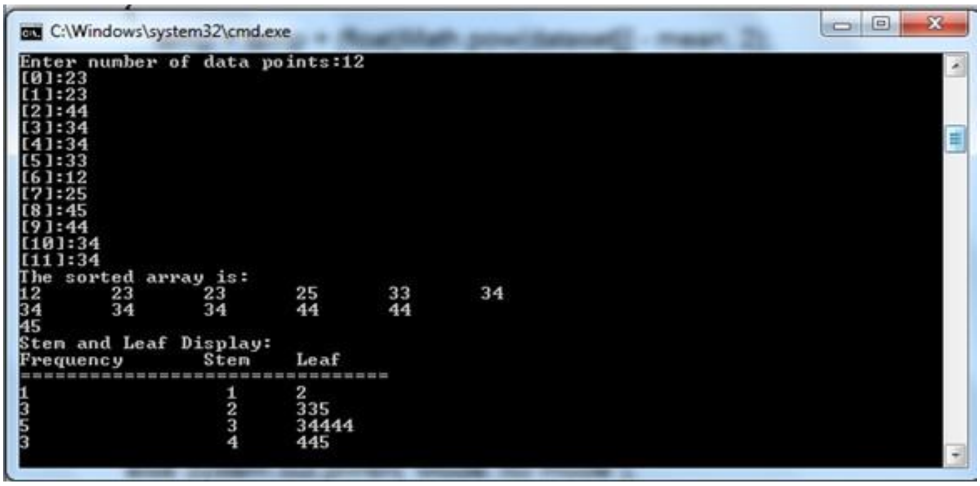
```
D:\Myjava>javac StatisticsInfo.java

D:\Myjava>java StatisticsInfo
Enter number of data points:12
[0]:23
[1]:23
[2]:33
[3]:23
[4]:43
[5]:43
[6]:43
[7]:41
[8]:23
[9]:23
[10]:32
[11]:23
Statistical Information:
===============================
Arithmetic mean:31.083334
Median:27.5
Mode:23
Standard deviation:9.149847

D:\Myjava>_
```

```
import

java.util.Scanner;

public class

StatisticsInfo
```

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

static void showStatistics(){
//

int n;
float mean,median,std;
Scanner sc=new Scanner(System.in);
System.out.print("Enter number of data
points:"); n=sc.nextInt();
if (n < 3)
{
System.out.println("The number of data points should be greater than 2.");

}
else
{

//declare an array of n size to store integral
data points int[] dataset = new int[n];
//allow user
inputs int i = 0;
for (i = 0; i < n; i++)
{
System.out.print("["+i+"]:"
); dataset[i] =
sc.nextInt();
}

//sort the data set
bubblesort(dataset,
n);

//calculate the
mean int sum =
0;
int j = 0;
while (j <
n)
{
sum = sum +
dataset[j]; j++;
}

mean = (float)sum / n;

//calculate median

//If n is odd, median=dataset[n/2]
//If n is even, median=(dataset[n/2]+dataset[1+n/2])/2
//The index of array starts from 0, so you need to subtract 1 from the indices
used in calculating the median
if (n % 2 != 0) median = dataset[n / 2];
else median = (dataset[(n / 2) - 1] + dataset[n / 2]) / (float)2;

//calculate the mode
int[][] mode = new int[n][2];
//initialize 2D array storing numbers of occurences,
and values for (i = 0; i < 2; i++)
for (j = 0; j < n; j++)
mode[j][i] = 0; mode[0][0] =
1;

for (i = 0; i < n;
i++) for (j = 0; j < n
- 1; j++)
if (dataset[i] == dataset[j + 1]) { ++mode[i][0]; mode[i][1] = dataset[i]; }

int max;
int k = 0;
max =
mode[0][0]; for
(j = 0; j < n; j++)
if (max < mode[j][0]) { max = mode[j][0]; k = j; }


//calculate standard
deviation,std float temp =
0.0f;
```

```
for (j = 0; j < n; j++)
{
temp = temp + (float)Math.pow(dataset[j] - mean, 2);
}

std = (float)Math.sqrt(temp / (n - 1));

//Show results

System.out.println("Statistical Information:");
System.out.println("================================");
System.out.println("Arithmetic
mean:"+mean);
System.out.println("Median:"+median)
;
if (mode[k][1] != 0)
System.out.println("Mode:"+
mode[k][1]); else
System.out.println("Mode: no
mode");
System.out.println("Standard deviation:"+std);

}
}}
```

Write a Java program to display an integer data set in the form of stem and leaf. The data points are input by the user from keyboard. This program will display the output similar to the one shown below:



```
Enter number of data points:12
[0]:23
[1]:23
[2]:44
[3]:34
[4]:34
[5]:33
[6]:12
[7]:25
[8]:45
[9]:44
[10]:34
[11]:34
The sorted array is:
12      23      23      25      33      34
34      34      34      44      44
45
Stem and Leaf Display:
Frequency       Stem    Leaf
================================
1                 1     2
3                 2     335
5                 3     34444
3                 4     445
```

```
import

java.util.Scanner;

public class

StemLeaf
{
public static void main(String[] args)
{
showStemLeaf();
}

static void showStemLeaf(){
//

int

n;

Scanner sc=new Scanner(System.in);
System.out.print("Enter number of data
points:"); n=sc.nextInt();
if (n < 3)
{
System.out.println("The number of data points should be greater than 2.");

}
else

{

//declare an array of n size to store integral
data points int[] dataset = new int[n];
//allow user
inputs int i = 0;
for (i = 0; i < n; i++)
{
System.out.print("["+i+"]:"
); dataset[i] =
sc.nextInt();
}

//sort the data set
bubblesort(dataset,
n);
stemleaf(dataset,n);

}
}

//method to sort data set
static void bubblesort(int[] dataset, int n)
{
int i, j;
for (i = 0; i < n;
i++) for (j = n - 1;
j > i; j--)
if (dataset[j] < dataset[j - 1])
{
int temp =
dataset[j]; dataset[j]
= dataset[j - 1];
dataset[j - 1] =
temp;
}

}

//show stem and leaf

static void stemleaf(int[] data_points,int n)
{


//Display sorted array
System.out.println("The sorted
array is:"); for(int i=0;i<n;i++){
System.out.print(data_points[i]+"\t
"); if(i%5==0 && i!=0)
System.out.println();
}
```

```
//store stem and leaf in a 2D
array int[][] stem_leaf=new
int[n][2]; for(int i=0;i<n;i++){
stem_leaf[i][0]=data_points[i
]/10;
stem_leaf[i][1]=data_points[i
]%10;
}


//initialize 2D array storing numbers of occurences,
and values int[][] mode=new int[n][2];
for(int i=0;i<n;i++)
for(int j=0;j<2;j++)mode[i][j]=0;


//find mode
mode[0][0]=1
; int
count=1;
for(int i=count-
1;i<n;i++){ for(int
j=count-1;j<n-1;j++){
if(stem_leaf[i][0]==stem_leaf[j+1][0])
{count++;mode[i][0]++;mode[i][1]=stem_leaf[i][0];} else if(i==0)
mode[i][1]=stem_leaf[i][0];}
}
System.out.println();
System.out.println("Stem and Leaf
Display:");
System.out.println("Frequency\tStem\
tLeaf");
System.out.println("=================================");
int c=0,leaf=0;
for(int
i=0;i<n;i++){
if(mode[i][1]!=0)
{
leaf+=mode[i][0]
;
```

```
System.out.format("%-18d",mode[i][0]);
System.out.format("%-
6d",mode[i][1]); for(int
j=c;j<leaf;j++){
System.out.format("%s",stem_leaf[j][
1]);}

c=leaf;
System.out.printl
n();


}
}
}
}
```

How to check if String has all unique characters in JAVA

In this post, we will see if String has all unique characters or not.

There are multiple ways to find if String has all unique characters or not.

By Using HashSet:

. You can add each character to HashSet.

. If HashSet’s add method returns false then it does not have all unique characters.

How to check if one String is rotation of Another String in JAVA

. Create a new String with str3= str1 + str1

- Check if str3 contains str2 or not.

- if str3 contains str2 then str2 is rotation of str1 else it is not

Java Program to check if one String is rotation of another.

```
package org.arpit.java2blog;

public class StringRotationMain {

  public static void main(String[] args) {

     System.out.println(
          "java2blog and blogjava2 are rotation of each other : " +
isRotation("java2blog", "blogjava
2")
);     System.out.println(
          "java2blog and avablog2j are rotation of each other : " +
          isRotation("java2blog", "avablog
2j")
);

  }

  public static boolean isRotation(String str, String rotation) {
     String str2 = str + str;

     if
     (str2.contains(rotation)
     ) {
        return true;
     }
     return false;

  }

  }
```

Find All substrings of A String in JAVA

In this post, we will see java program to find all substrings of a String. For example: If input is “abb” then output should be “a”, “b”,”b”,

“ab”, “bb”, “abb” We will use String class’s subString method to find all subString

Program:

```
class SubstringsOfStringMain
{
public static void main(String args[])
{
 String str="abbc";
 System.out.println("All substring of abbc are:");
  for (int i = 0; i < str.length(); i++) {
  for (int j = i+1; j <= str.length(); j++) {
       System.out.println(str.substring(i,j));

  }
 }
}
}
```

When you run above program, you will get following output:

```

All substring of abbc are:
a
ab
abb
abbc
b
bb
bbc
b
bc
c

```

Find All PermutAtions of A String in JAVA

To find all permutations of String in java. We will use a very simple approach to do it.

> Take out first character of String and insert into different places of permutations of remaining String recursively.

Lets say you have String as **ABC**. So we take out A from ABC

First character =A and RemainingString = BC

As we are applying recursion here, we will find permutations of BC. Take out B from BC.

First character= B and RemainingString = C

As we are applying recursion here, we will find permutations of C.

When we take out C, our String size becomes 0 and that is our base case here.

First character = C and RemainingString = “”

We insert C to differences indexes of Permutations of RemainingString(“”), so we get permutation of C as C.

We insert B to different indexes of Permutations of remainingString(C), so we get BC and CB.

C: BC, CB

Now we insert A to different indexes in BC and CB.

BC : ABC , BAC ,

BCA CB : ACB,

CAB, CBA

so thats how we got all permutations of ABC.

It may look tricky but once you practice the solution, you will be able to understand it better.

Java program to find all Permutations of String in java:

```java

package org.arpit.java2blog;

import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

public class PermutationOfStringJava {

public static void main(String[]
args) {
  Set set=permutationOfString("ABC");
 System.out.println("Permutations of String ABC
 are:"); for (Iterator iterator = set.iterator();
 iterator.hasNext();) { String string = (String)
 iterator.next(); System.out.println(string);
 }

}

public static Set
permutationOfString(String str) {
 Set permutationSet=new
HashSet();

if(str.length()==0)
 {
  permutationSet.add("");
          return
permutationSet; }

 // take out first character of String
 char c=str.charAt(0);
 // Remaining String
 String
rem=str.substring(1);
 Set permutatedSetForRemainingString=permutationOfString(rem);


  for (String permutedString: permutatedSetForRemainingString) {
   for (int j = 0; j <= permutedString.length(); j++) {
    String
    permutation=insertFirstCharAtDiffPlaces(permutedString
    ,c,j); permutationSet.add(permutation);
   }

  }
  return permutationSet;

 }
 public static String insertFirstCharAtDiffPlaces(String perm,char
 firstChar,int index)
 {
  // Inserting firstCharacter of orig String at difference places based on
  index
  return perm.substring(0,index)+firstChar+perm.substring(index);
 }

 }
```

When you run above program, you will get following out:

```
Permutations of String ABC are:
ACB
ABC
BCA
CBA
CAB
BAC
```

Longest Common Prefix in an Array of Strings in Java

```java
String[] strArr={"java2blog","javaworld","javabean","javatemp"};
```

So Longest common prefix in above String array will be “java” as all above string starts with “java”. Lets take one more example:

```java
String[] strArr={"sqlblog","sql2world","sqlquery","sqlproc"};
```

So Longest common prefix in above String array will be “sql” as all above string starts with “sql”.

Algorithm:

- Find minimum length String.
- Iterate over array of String and if we find any mismatch with minimum length String, we break the loop and that index will give us longest common prefix of this array of String,
-

Java Program to find Longest Common Prefix:

Create a main Class called LongestCommonPrefixMain.java.

```

package org.arpit.java2blog;

public class LongestCommonPrefixMain {

public static void main(String[] args)
{
String[] strArr={"java2blog","javaworld","javabean","javatemp"};
String longestPrefix=getLongestCommonPrefix(strArr);
System.out.println("Longest Prefix : "+longestPrefix);
}
public static String getLongestCommonPrefix(String[] strArr) {
if(strArr.length==0) return "";
// Find minimum length String
String minStr=getMinString(strArr);

int minPrefixStrLength=minStr.length();
for(int i=0;i<strArr.length;i++){
int j;
for( j=0;j<minPrefixStrLength;j++){
if(minStr.charAt(j)!=strArr[i].charAt(j))
break;
}
if(j<minPrefixStrLe
ngth)
minPrefixStrLength
=j; }
return
minStr.substring(0,minPrefixStrLength);
}
public static String
getMinString(String[] strArr) {
String minStr=strArr[0];
for(int
i=1;i<strArr.length;i++){
if(strArr[i].length()<minStr.le
ngth()) minStr=strArr[i];
}
return
minStr; }
}
```

When you run above program, you will get below output:

```

Longest Prefix : java
```