Java Hackathon Question and Answers

Q1. Consider there is a 3 Boolean variable called a, b, c. Check if at least two out of three Booleans are true

```
package hackathon;
import java.util.Scanner;
public class Q_1 {
public static void main(String[] args) {
Scanner scan=new Scanner(System.in);
System.out.println("enter boolean value of a ");
boolean a=scan.nextBoolean();
System.out.println("enter boolean value of b");
boolean b=scan.nextBoolean();
System.out.println("enter boolean value of c");
boolean c=scan.nextBoolean();

if((a && b) || (a&&c) || (b&&c) ){
System.out.println("2 values are true");
}
else{
System.out.println("two values are not ture");
}
}
}
```

Q2. write a program to find factorial (Non Recursive)

```
package hackathon;
import java.util.Scanner;
public class Q_2 {
//find factorial of a number(non recursive)
public static int factorial(int n){
int fact=1;
for(int i=1;i<=n;i++)
{
fact=fact*i;
}
return fact;
}
public static void main(String[] args) {
Scanner scan = new Scanner(System.in);
System.out.println("enter value of n");
int n=scan.nextInt();
int result=factorial(n);
System.out.println("Result: "+ result);
```

```
}
}
```

Q3. Given an array of integers, sort the integer values.

```
package hackathon;
import java.util.Arrays;
import java.util.Scanner;
public class Q_3 { //sort array of integers
public static void displayArray(int[ ] arr){
for(int i=0;i<arr.length;i++){
System.out.print(arr[i]+ " ");
}
}
public static void quickSort(int[] arr, int low, int high) {
if (arr == null || arr.length == 0)
return;
if (low >= high)
return;
int middle = low + (high - low) / 2;
int pivot = arr[middle];
int i = low, j = high;
while (i <= j) {
while (arr[i] < pivot) {
i++;
}
while (arr[j] > pivot) {
j--;
}
if (i <= j) {
int temp = arr[i];
arr[i] = arr[j];
arr[j] = temp;
i++;
j--;
}
}
if (low < j)
quickSort(arr, low, j);
if (high > i)
quickSort(arr, i, high);
}
public static void main(String[] args){
Scanner scan = new Scanner(System.in);
```

```
System.out.println("enter the size of the array");
int size=scan.nextInt();
System.out.println("enter the array values");
int [] arr=new int[size];
for(int i=0;i<size;i++){
arr[i]=scan.nextInt();
}
int low = 0;
int high = arr.length - 1;
quickSort(arr, low, high);
displayArray(arr);
}
}
```

Q4. Given an array of integers check the Palindrome of the series.

```
package hackathon;
import java.util.Scanner;
public class Q_4 {
public static int isPalindrome(int num){
int number=num,remainder=0,reverse=0, flag=0;
while(number!=0){
remainder=number%10;
number=number/10;
reverse=reverse*10+remainder;
}
if(reverse==num){
flag=1;
}
else{
flag=0;
}
return flag;
}
public static void main(String[] args) {
Scanner scan = new Scanner(System.in);
System.out.println("enter the size of the array");
int size=scan.nextInt();
if (size == 0){
System.out.println("array should have at least 1 value");
return;
}
System.out.println("enter the array values");
int [] arr=new int[size];
for(int i=0;i<size;i++){
```

```
arr[i]=scan.nextInt();
}
int flag=0;
for(int i=0;i<arr.length;i++){
flag=isPalindrome(arr[i]);
if(flag==1){
System.out.println(arr[i]+" - Is Palindrome");
}
}
}
}
```

Q5. Given an array prints the unique numbers and also print the number of occurrences of duplicate numbers.

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

public class Q_5 {
public static void main(String[] args) {
        int l;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the length of
the array : ");
        l=sc.nextInt();
        int a[]=new int [l];
        System.out.println("Enter the "+l+ "
integer values: ");
        for(int i=0;i<l;i++)
        {
            a[i]=sc.nextInt();
        }
        sc.close();

        Q_5 q=new Q_5();
         q.findout(l,a);
    }

    public void findout(int s, int b[])
    {
        Arrays.sort(b);
```

```
        for(int i=0;i<s;i++)
        {
            int count=1;
            for(int j=i+1;j<s;j++)
            {
                if (b[j]==b[j-1])
                {
                    count ++;
                    i++;
                }
                else
                    break;
            }
            if(count==1)
            {
                System.out.println("the "+
b[i]+" is unique no");
            }
            else
            {
                System.out.println("The
occurence of "+b[i]+" is "+count);
            }
        }
        System.out.println("the array is "+
Arrays.toString(b));
    }
}
```

Q6. WJP to perform ascending order Selection sort

```
package hackathon;
import java.util.Scanner;
//Asecding order selection sort
public class Q_6 {
public static void main(String[] args) {
Scanner scan = new Scanner(System.in);
System.out.println("enter the size of the array");
int size=scan.nextInt();
System.out.println("enter the array values");
```

```
int [] arr=new int[size];
for( int i = 0; i < arr.length; i++ ){
arr[i]=scan.nextInt();
}
for ( int i = 0; i < arr.length; i++ ) {
//find minimum, starting from index i
int minIndex = i;
int min = arr[i];
for ( int j = i + 1; j < arr.length; j++ ) {
if ( arr[ j ] < min ) {
minIndex = j;
min = arr[j];
}
}
// now move the smallest element to the front, and the element at index i to the
index of the minimal element
int temp = arr[ i ];
arr[ i ] = min;
arr[ minIndex ] = temp;
}
for ( int i = 0; i < arr.length; i++ ) {
System.out.print(arr[i]+" ");
}
}
}
```

Q7. what are the different ways to create String object?

There are two methods to create String object

1.Using String literal

example: String str=”Hello”;

2.Using new keyword

example: String str1=new String(“Hello World”);

Q8.how can we make string uppercase or lowercase?

Using String methods toUpperCase() we can convert a string to UpperCase

```
String  s1="hello";
String upper=s1.toUpperCase();
System.out.println(upper);
```

OUTPUT: HELLO

Q9. how can we make string uppercase or lowercase?

Using String methods toLowerCase() we can convert a string to LowerCase

```
String  s1="HELLO";
```

```
String lower=s1.toLowerCase();
        System.out.println(lower);
```

OUTPUT: hello

Q10. what is string subsequence method?

This method returns a new character sequence that is a subsequence of this sequence.

```
String Str = new String("Hello Beautiful World");

System.out.print("Value :" );
System.out.println(Str.subSequence(0, 15) );
System.out.print("Value :" );
System.out.println(Str.subSequence(16, 21) );
```

OUTPUT:

Value :Hello Beautiful

Value :World

Q11. how to split a string in java?

We can split a string using java split() method. This method splits the string against given regular expression and returns a char array.

```
String[] Str = new String("Hello Beautiful World").split(" ");
System.out.println(Arrays.toString(Str));
```

OUTPUT: [Hello, Beautiful, World]

Q12. Write a program to check palindrome (MalayalaM) for both numbers and string?

```
package hackathon;
import java.util.Scanner;
//To check Palindrome for numbers and string
public class Q_12 {
public static void main(String[] args) {
// TODO Auto-generated method stub
Scanner scan=new Scanner(System.in);
System.out.println("enter your input");
String input=scan.nextLine();
//System.out.println("you have entered: " + input);
int i=0,flag=1;
int j=input.length()-1;
while(i<=j){
if(input.charAt(i)==input.charAt(j)){    //Works for both characters and numbers
i++;
j--;
```

```
}
else{
break;
}
}
if(i>j){
System.out.println(input+" is a palindrome");
}
else{
System.out.println(input+ " is not a palindrome");
}
}
}
```

Q13. Given a string print the reverse of the string.(Input: Java Code Output: edoC avaJ)

```
package hackathon;
import java.util.Scanner;
//REverse a given string
public class Q_13 {
public static void main(String[] args) {
Scanner scan = new Scanner(System.in);
System.out.println("enter input string");
String str=scan.nextLine();
for(int i=str.length()-1;i>=0;i--){
System.out.print(str.charAt(i));
}
}
}
```

Q14. Given a string print the reverse of the words string.(Input: Java Code Output: Code Java)

```
package hackathon;
import java.util.Scanner;
public class Q14 {
public static void main(String[] args) {
Scanner scan = new Scanner(System.in);
System.out.println("enter input string");
String[] str=scan.nextLine().split(" ");
for(int i=str.length-1;i>=0;i--){
System.out.print(str[i]+ " ");
}
}
```

```
}
```
Q15. Given a string print the unique words of the string.
```
package hackathon;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;
//Print unique words of the given string
public class Q15 {
public static void main(String[] args) {
// TODO Auto-generated method stub
Scanner scan = new Scanner(System.in);
System.out.println("enter input string");
String[] str=scan.nextLine().split(" ");
ArrayList<String> al=new ArrayList<String>();
for(String s: str){
if(!al.contains(s)){
al.add(s);
}
}
Iterator<String> it= al.iterator();
for(String s:al){
System.out.print(it.next()+ " ");
}
}
}
```
Q16. Write a method that will remove given character from the String?
```
package hackathon;
import java.util.Scanner;
public class Q16 {
//REmove given character from string
public static void main(String[] args) {
Scanner scan = new Scanner(System.in);
System.out.println("enter input string");
String str=scan.nextLine();
System.out.println("enter the character you want to remove");
String c=scan.nextLine();
if(str.contains(c)){
String newStr=str.replace(c,"");
System.out.println(newStr);
}
else{
System.out.println(c+" is not found in the given string");
}
```

```
}
}
```

Q17. WJP to find total number of integers, uppercase and lowercase character in the give string.

```
package hackathon;
import java.util.Scanner;
public class Q17 {
public static void main(String[] args) {
Scanner scan =new Scanner(System.in);
System.out.println("enter the input");
String str=scan.nextLine();
int upperCaseCount=0, lowerCaseCount=0, integerCount=0;
for(int i=0;i<str.length();i++){
if(Character.isUpperCase(str.charAt(i))){
upperCaseCount++;
}
else if(Character.isLowerCase(str.charAt(i))){
lowerCaseCount++;
}
else if(Character.isDigit(str.charAt(i))){
integerCount++;
}
}
System.out.println("upperCaseCount: "+ upperCaseCount);
System.out.println("lowerCaseCount: "+ lowerCaseCount);
System.out.println("integerCount: "+integerCount);
}
}
```

Q18. WJP to display duplicate character in string.

```
package hackathon;
import java.util.Scanner;
public class Q18 {
//Display duplicate characters in String
public static void main(String[] args) {
// TODO Auto-generated method stub
Scanner input= new Scanner(System.in);
System.out.println(" enter ur string \n");
String inputString=input.nextLine();
int[] occured=new int[128];
for(int i=0;i<inputString.length();i++){
if(inputString.charAt(i)!=' ')
```

```
occured[inputString.charAt(i)]++;
}
for(int j=0;j<occured.length;j++){
if(occured[j]>1){
System.out.println((char)(j) + " is duplicate");
}
}
}
}
```

Q19. WJP to display number of occurrence of all character

```
package hackathon;
import java.util.Scanner;
//Occurance of each character
public class Q19 {
public static void main(String[] args) {
// TODO Auto-generated method stub
Scanner input= new Scanner(System.in);
System.out.println(" enter ur input \n");
String inputString=input.nextLine();
int[] occured=new int[128];
for(int i=0;i<inputString.length();i++){
if(inputString.charAt(i)!=' ')
occured[inputString.charAt(i)]++;
}
for(int j=0;j<occured.length;j++){
if(occured[j]!=0){
System.out.println((char)(j) + " occured "+occured[j]+ " times" );
}
}
}}
```

Q20. WJP to find total number of repeated integers, uppercase and lowercase character in the give string

```
package hackathon;
import java.util.*;
//number of repeated integers, lowercase and uppercase characters in string
public class Q20 {
public static void main(String[] args) {
// TODO Auto-generated method stub
Scanner input= new Scanner(System.in);
System.out.println(" enter ur string \n");
String inputString=input.nextLine();
int[] occured=new int[128];
for(int i=0;i<inputString.length();i++){
```

```
if(inputString.charAt(i)!=' ')
occured[inputString.charAt(i)]++;
}
for(int j=0;j<occured.length;j++){
if(occured[j]>1 && ( j >=48 &&j<= 57)){
System.out.println((char)(j) + " integer and is repeated "+occured[j]+ " times"
);
}
if(occured[j]>1 && ( j >=65 &&j<= 90)){
System.out.println((char)(j) + " Upper Case character and is repeated
"+occured[j]+ " times" );
}
if(occured[j]>1 && ( j >=97 &&j<= 122)){
System.out.println((char)(j) + " Lower Case character and is repeated
"+occured[j]+ " times" );
}
}
}
}
```

Q21. WJP to convert string to int.

```
package hackathon;
import java.util.Scanner;
//Convert string to int
public class Q21 {
public static void main(String[] args) {
// TODO Auto-generated method stub
Scanner scan= new Scanner(System.in);
System.out.println("enter your input");
String str=scan.nextLine();
try{
int number=Integer.parseInt(str);
System.out.println("Value: "+number);
}
catch(Exception e){
System.out.println("Exception" + e);
System.out.println("please enter valid input");
}
}
}
```

Q22. WJP to convert int to string

```
package hackathon;
import java.util.Scanner;
//Convert int to string
```

```
public class Q22 {
public static void main(String[] args) {
// TODO Auto-generated method stub
Scanner scan= new Scanner(System.in);
System.out.println("enter your input");
try{
int number=scan.nextInt();
String str=String.valueOf(number);
System.out.println("Value: "+str);
}
catch(Exception e){
System.out.println("Exception" + e);
System.out.println("please enter valid input");
}
}
}
```

Q23. WJP to differentiate input as string, int or bool

```
package hackathon;
import java.util.Scanner;
//Check if input is string, int or boolean
public class Q23 {
public static void main(String[] args) {
String input;
int ch1;
Scanner one = new Scanner(System.in);
System.out.println("enter your input");
input = one.nextLine();
try {
ch1 = Integer.parseInt(input);
System.out.println("integer");
return;
}
catch (NumberFormatException e) {
}
try {
if(input.equalsIgnoreCase("true")||input.equalsIgnoreCase("false")){
System.out.println("boolean");
return;
}
else{
System.out.println("String");
return;
}
```

```
} catch (NumberFormatException e) {
}
}
}
```

Q24. Write a program which inputs a positive natural number N and prints the possible consecutive number combinations, which when added give N. INPUT: N = 9 OUTPUT: 4 + 5 2 + 3+ 4

```
package hackathon;
import java.io.*;
import java.util.Scanner;
/*
* prints the possible consecutive number combinations,
* which when added give N.
*  INPUT: N = 9 OUTPUT: 4 + 5 2 + 3+ 4
*/
public class Q24 {
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
System.out.print("Enter a number : ");
int n=in.nextInt();
int sum=0,c=0,j=0;
for(int i=1;i<n;i++)
{
sum=i;
j=i+1;
while(sum<n)
{
sum=sum+j;
j++;
}
if(sum==n)
{
for(int k=i;k<j;k++)
{
if(k==i)
System.out.print(k);
else
System.out.print(" + "+k);
}
System.out.println();
}
}
```

}

}

Q25. Write a program for binary search. And 5 i/p has to take from user as binary elements.

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

public class Q_25 {

    public static void main(String[] args) {

        int c, first, last, middle, n, search;
        Scanner in = new Scanner(System.in);
        int[] array = new int[5];
        System.out.println("Enter " + 5 + "
integers");
        if(array.length!=5){
            System.out.println("Please enter 5
values");
        }
        for (c = 0; c < 5; c++)
            array[c] = in.nextInt();
        System.out.println("Entered array: " +
Arrays.toString(array));
        Arrays.sort(array);
        System.out.println("Sorted array: " +
Arrays.toString(array));

        System.out.println("Enter the element
to search: ");
        int key = in.nextInt();
        int res = Arrays.binarySearch(array,
key);
        if (res >= 0)
            System.out.println(key + " found at
index = "
```

```
                    + res);
        else
            System.out.println(key + " Not
found");

    }

}
```

Q26. WJP to merge two sorted array.(Do not use third array) array1[10] = 1,2,4,6,9,10 array2[4] = 3, 5,7,8 After merge : array1[10] = 1,2,3,4,5,6,7,8,9,10

```
package hackathon;
import java.util.Arrays;

public class Q_26 {
    public static void merge(int array1[], int
array2[]) {

        int m  = array1.length-1;
        for(int i=array1.length-1; i>=0; i--) {

            if(array1[i] != 0) {
                array1[m] = array1[i];
                m = m-1;
            }
        }

        int i = m+1;
        int j=0, k=0;
        while((i<array1.length) &&
(j<array2.length)) {
            if(array1[i] < array2[j]) {
                array1[k++] = array1[i++];
            }
            else
                array1[k++] = array2[j++];
        }
        while(j < array2.length) {
```

```
            array1[k++] = array2[j++];
        }
    }


    public static void main(String[] args) {
        int array1[] = {1,2,4,6,9,10,0,0,0,0};
        int array2[] = {3,5,7,8};


        merge(array1, array2);
        System.out.println("Merged array : " +
Arrays.toString(array1));
    }
}
```

Q27. WJP to perform ascending order Quick sort

```
package hackathon;
import java.util.Scanner;
//Merge sort
public class Q27 {
private int[] array;
private int[] tempMergArr;
private int length;
public static void main(String[] args){
Scanner input=new Scanner(System.in);
System.out.println("enter size of your array");
int size1=input.nextInt();
int[] inputArr=new int[size1];
System.out.println("please enter"+size1+" values ");
for(int i=0;i<inputArr.length;i++){
inputArr[i]=input.nextInt();
}
Q27 prg = new Q27();
prg.sort(inputArr);
for(int i:inputArr){
System.out.print(i);
System.out.print(" ");
}
}
public void sort(int[] inputArr) {
```

```
this.array = inputArr;
this.length = inputArr.length;
this.tempMergArr = new int[length];
doMergeSort(0, length - 1);
}
public void doMergeSort(int lowerIndex, int higherIndex) {
if (lowerIndex < higherIndex) {
int middle = lowerIndex + (higherIndex - lowerIndex) / 2;
// Below step sorts the left side of the array
doMergeSort(lowerIndex, middle);
// Below step sorts the right side of the array
doMergeSort(middle + 1, higherIndex);
// Now merge both sides
mergeParts(lowerIndex, middle, higherIndex);
}
}
public void mergeParts(int lowerIndex, int middle, int higherIndex) {
for (int i = lowerIndex; i <= higherIndex; i++) {
tempMergArr[i] = array[i];
}
int i = lowerIndex;
int j = middle + 1;
int k = lowerIndex;
while (i <= middle && j <= higherIndex) {
if (tempMergArr[i] <= tempMergArr[j]) {
array[k] = tempMergArr[i];
i++;
} else {
array[k] = tempMergArr[j];
j++;
}
k++;
}
while (i <= middle) {
array[k] = tempMergArr[i];
k++;
i++;
}
}
}
```

Q28. WJP to find factorial of a number using recursion.

```
package hackathon;
import java.util.Scanner;
```

```
public class Q28{
public static void main(String args[]){
//Scanner object for capturing the user input
Scanner scanner = new Scanner(System.in);
System.out.println("Enter the number:");
//Stored the entered value in variable
int num = scanner.nextInt();
//Called the user defined function fact
int factorial = fact(num);
System.out.println("Factorial of entered number is: "+factorial);
}
static int fact(int n)
{
int output;
if(n==1){
return 1;
}
//Recursion: Function calling itself!!
output = fact(n-1)* n;
return output;
}
}
```

Q29. WJP to perform Merge sort using recursion

```
package hackathon;
import java.util.Scanner;
public class Q29 {
public static void merge(int data[], int left, int right, int middle) {
int first_data = middle - left + 1;
int second_data = right - middle;
int a = 0, b = 0, c = left;
int Left[] = new int[first_data];
int Right[] = new int[second_data];
for (int x = 0; x < first_data; x++) {
Left[x] = data[left + x];
}
for (int y = 0; y < second_data; y++) {
Right[y] = data[middle + 1 + y];
}
while (a < first_data && b < second_data) {
if (Left[a] < Right[b]) {
data[c] = Left[a];
```

```
a++;
c++;
} else {
data[c] = Right[b];
b++;
c++;
}
}
while (a < first_data) {
data[c] = Left[a];
a++;
c++;
}
while (b < second_data) {
data[c] = Right[b];
b++;
c++;
}
}
public static void MergeSort(int data[], int start, int end) {
if (start < end) {
int middle = (end + start) / 2;
MergeSort(data, start, middle);
MergeSort(data, middle + 1, end);
merge(data, start, end, middle);
}
}
public static void Non_recursive_mergeSort(int data[], int start, int end) {
int i = 2, j = 0;
while (i < data.length) {
j = 0;
while (j < data.length - 1) {
int right = (j + i) - 1;
int left = j;
if (right > data.length) {
right = data.length - 1;
}
int middle = (left + right) / 2;
merge(data, left, right, middle);
j = j + i;
}
i = i * 2;
if (i >= data.length) {
```

```
i = i / 2;
merge(data, 0, data.length - 1, (i - 1));
i = data.length;
}
}
}
public static void main(String[] args) {
// TODO Auto-generated method stub
Scanner sc=new Scanner (System.in);
System.out.println("Enter N ( how many numbers to be sorted)");
int n = sc.nextInt();
int [] data=new int[n];
System.out.println("Enter "+n+ " numbers 1 by 1");
for (int i=0 ; i<n; i++)
{
int  number = sc.nextInt();
data[i]=number;
}
System.out.print("List before  sorting n");
System.out.println();
for (int i = 0; i < n; i++) {
System.out.print(data[i] + " ");
}
System.out.println();
System.out.print("After sorting, numbers are ");
System.out.println();
MergeSort(data, 0, data.length - 1);
for (int i = 0; i < n; i++) {
System.out.print(data[i] + " ");
}
}
}
```

Q30. Write a function to find out longest palindrome in a given string?

```
package hackathon;
import java.util.Scanner;
public class Q30 {
/**
* @param input is a String input
* @return The longest palindrome found in the given input.
*/
public static String getLongestPalindrome(final String input) {
int rightIndex = 0, leftIndex = 0;
String currentPalindrome = "", longestPalindrome = "";
```

```
for (int centerIndex = 1; centerIndex < input.length() - 1; centerIndex++) {
leftIndex = centerIndex - 1;  rightIndex = centerIndex + 1;
while (leftIndex >= 0 && rightIndex < input.length()) {
if (input.charAt(leftIndex) != input.charAt(rightIndex)) {
break;
}
currentPalindrome = input.substring(leftIndex, rightIndex + 1);
longestPalindrome = currentPalindrome.length() > longestPalindrome.length() ?
currentPalindrome : longestPalindrome;
leftIndex--;  rightIndex++;
}
}
return longestPalindrome;
}
public static void main(String ... args) {
Scanner scn = new Scanner(System.in);
System.out.println("Enter string");
String str = scn.nextLine();
String longestPali = getLongestPalindrome(str);
System.out.println("String: " + str);
System.out.println("Longest Palindrome: " + longestPali);
}
}
```

Q31. Read a file content and write it to a new file in reverse order.(reverse line 1-10 to line 10-1)

Only Read Activity:

```
package hackathon;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.io.UnsupportedEncodingException;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
public class Q31 {
private static final int BUFFER_SIZE = 8192;
private final FileChannel channel;
```

```
private final String encoding;
private long filePos;
private ByteBuffer buf;
private int bufPos;
private ByteArrayOutputStream baos = new ByteArrayOutputStream();
private RandomAccessFile raf;
private byte lastLineBreak = '\n';
public Q31(File file) throws IOException {
this(file, null);
}
public Q31(File file, String encoding) throws IOException {
raf = new RandomAccessFile(file, "r");
channel = raf.getChannel();
filePos = raf.length();
this.encoding = encoding;
}
public void close() throws IOException {
raf.close();
}
public String readLine() throws IOException {
byte c;
while (true) {
if (bufPos < 0) {
if (filePos == 0) {
if (baos == null) {
return null;
}
String line = bufToString();
baos = null;
return line;
}
long start = Math.max(filePos - BUFFER_SIZE, 0);
long end = filePos;
long len = end - start;
buf = channel.map(FileChannel.MapMode.READ_ONLY, start, len);
bufPos = (int) len;
filePos = start;
// Ignore Empty New Lines
c = buf.get(--bufPos);
if (c == '\r' || c == '\n')
while (bufPos > 0 && (c == '\r' || c == '\n')) {
bufPos--;
c = buf.get(bufPos);
```

```
}
if (!(c == '\r' || c == '\n'))
bufPos++;// IS THE NEW LENE
}

/*
* This will ignore all blank new lines.
*/
while (bufPos-- > 0) {
c = buf.get(bufPos);
if (c == '\r' || c == '\n') {
// skip \r\n
while (bufPos > 0 && (c == '\r' || c == '\n')) {
c = buf.get(--bufPos);
}
// restore cursor
if (!(c == '\r' || c == '\n'))
bufPos++;// IS THE NEW Line
return bufToString();
}
baos.write(c);
}
}
}
private String bufToString() throws UnsupportedEncodingException {
if (baos.size() == 0) {
return "";
}
byte[] bytes = baos.toByteArray();
for (int i = 0; i < bytes.length / 2; i++) {
byte t = bytes[i];
bytes[i] = bytes[bytes.length - i - 1];
bytes[bytes.length - i - 1] = t;
}
baos.reset();
if (encoding != null)
return new String(bytes, encoding);
else
return new String(bytes);
}
@SuppressWarnings("resource")
public static void main(String[] args) throws IOException {
```

```
FileReader logReader = new FileReader("C:\\Users\\Tekarch\\Desktop\\Assignments\\README.TXT");
BufferedReader buffer = new BufferedReader(logReader);
System.out.println("~~~~~~~~~~~~~~~~~~~~Simple way to read file in Java without Reversing ~~~~~~~~~~~~~~~~~~~\n");
for (String line1 = buffer.readLine(); line1 != null; line1 = buffer.readLine()) {
System.out.println(line1);
}
File file = new File("C:\\Users\\Tekarch\\Desktop\\Assignments\\README.TXT");
Q31 reader = new Q31(file, "UTF-8");
String line;
System.out.print("\n~~~~~~~~~~~~~~~~~~~~ Reading a file in Reverse Order ~~~~~~~~~~~~~~~~~~~~ \n\n");
while ((line = reader.readLine()) != null) {
System.out.println(line);

}
}
}
```

Read&Write activity:

```
import java.io.BufferedReader;
//import java.io.File;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;


public class Question31 {

	public static void main(String[] args) throws
FileNotFoundException {
		String
path1="/Users/kalpeshpatel/Documents/WebEx/File1.txt";
		String
path2="/Users/kalpeshpatel/Documents/WebEx/File2.txt";
		try {
			BufferedReader input = new BufferedReader(new
FileReader(path1));
			ArrayList list = new ArrayList();
			String line;
```

```
                    while ((line = input.readLine()) != null) {
                      list.add(line);
                    }
                    input.close();

                    Collections.reverse(list);

                    PrintWriter output = new PrintWriter(new
BufferedWriter(new FileWriter(path2)));
                    for (Iterator i = list.iterator(); i.hasNext();)
                    {
                      output.println((String) i.next());
                    }
                    output.close();
                  }
                catch (IOException e) {
                    System.err.println(e);
                  }

                /*try{
                    File sourceFile=new File(path1);
                    Scanner content=new Scanner(sourceFile);
                    PrintWriter pwriter =new PrintWriter(path2);

                    while(content.hasNextLine())
                    {
                       String s=content.nextLine();
                       StringBuffer buffer = new StringBuffer(s);
                       buffer=buffer.reverse();
                       String rs=buffer.toString();

                       pwriter.println(rs);
                    }
                    content.close();
                    pwriter.close();
                    System.out.println("File is copied successful!");
                    }

                    catch(Exception e){
                        System.out.println("Something went wrong");
                    }
                 }*/



        }
}
```

Q32. You are given two sorted arrays, A and B, and A has a large enough buffer at the end to hold B. Write a method to merge B into A in sorted order.

```
package hackathon;
public class Q32
```

```
{
public void merge( int[] A, int[] B, int size_A, int size_B )
{
int total_size = size_A + size_B;
int point_A = size_A - 1;
int point_B = size_B - 1;
for( int i = total_size - 1; i >= 0; i-- )
{
if( point_A >= 0 && point_B >=0 )
{
if( A[point_A] >= B[point_B] )
{
A[i] = A[point_A--];
}
else
{
A[i] = B[point_B--];
}
}
else if( point_B >= 0 )
{
A[i] = B[point_B--];
}
else
{
break;
}
}
}
public static void main( String[] args )
{
int[] a = {1,3,5,7,0,0,0,0,0,0,0,0,0};
int[] b = {2,4,6,8};
Q32 m = new Q32();
m.merge( a, b, 4, 4 );
for( int i = 0; i < 8; i++ )
{
System.out.print( a[i] );
}
System.out.println();
}
}
```

Q33. A circus is designing a tower routine consisting of people standing atop one another's shoulders. For practical and aesthetic reasons, each person must be both shorter and lighter than the person below him or her. Given the heights and weights of each person in the circus, write a method to compute the largest possible number of people in such a tower. EXAMPLE: Input (ht, wt): (65, 100) (70, 150) (56, 90) (75, 190) (60, 95) (68, 110) Output: The longest tower is length 6 and includes from top to bottom: (56, 90) (60,95) (65,100) (68,110) (70,150) (75,190)

```
package hackathon;
import java.util.*;
public class Q33{
public static ArrayList<Person> findHighestTower(List<Person> persons) {
if (persons == null) return null;
return findHighestTower(persons, null);
}
private static ArrayList<Person> findHighestTower(List<Person> persons,
Person bottom) {
int maxNum = 0;
ArrayList<Person> maxTower = null;
for (Person p : persons) {
if (p.canStandAbove(bottom)) {
ArrayList<Person> tower = findHighestTower(persons, p);
int num = tower.size();
if (num > maxNum) {
maxNum = num;
maxTower = tower;
}
}
}
if (maxTower == null) maxTower = new ArrayList<Person>();
if (bottom != null) maxTower.add(bottom);
return new ArrayList<Person>(maxTower);
}

//treat it as a longest-increasing-subsequence (LIS) problem
public static ArrayList<Person> findHighestTower2(List<Person> persons) {
if (persons == null) return null;
Collections.sort(persons);
ArrayList<Person>[] solutions = new ArrayList[persons.size()];
getLIS(persons, solutions, 0);
ArrayList<Person> bestSolution = new ArrayList<Person>();
for (ArrayList<Person> solution : solutions) {
if (solution.size() > bestSolution.size())
```

```
bestSolution = solution;
}
return bestSolution;
}
private static void getLIS(List<Person> persons, ArrayList<Person>[]
solutions, int index) {
if (index > persons.size() - 1) return;
ArrayList<Person> bestSolution = new ArrayList<Person>();
Person current = persons.get(index);
for (int i = 0; i < index; ++i) {
if (persons.get(i).weight < current.weight) {
if (solutions[i] != null & solutions[i].size() > bestSolution.size()) {
bestSolution = solutions[i];
}
}
}
ArrayList<Person> newSolution = new ArrayList<Person>(bestSolution);
newSolution.add(current);
solutions[index] = newSolution;
getLIS(persons, solutions, index + 1);
}
private static class Person implements Comparable<Object> {
int height, weight;
public Person(int h, int w) {
height = h;
weight = w;
}
private boolean canStandAbove(Person that) {
return that == null ||
(height < that.height &&
weight < that.weight);
}
public String toString() {
return "(" + height + ", " + weight + ")";
}
public int compareTo(Object o) {
Person that = (Person) o;
return height != that.height ?
((Integer) height).compareTo(that.height) :
((Integer) weight).compareTo(that.weight);
}
}
//TEST-------------------------------
```

```
public static void main(String[] args) {
Person[] persons = {
new Person(56,94),
new Person(60,95),
new Person(65,100),
new Person(68,93),
new Person(70,150),
new Person(75,200),
new Person(75,100),
new Person(76,190),
new Person(76,220),
};
List<Person> list = Arrays.asList(persons);
Collections.shuffle(list);
System.out.println(findHighestTower(list));
// System.out.println(findHighestTower2(list));
}
}
```

Q34. Write a method to implement *, - , / operations. You should use only the + operator

```
package hackathon;
import java.util.Scanner;
public class Q34 {
public static int negate(int number) {
return ~number + 1;
}
public static int abs(int number) {
return number < 0 ? negate(number) : number;
}
public static int multiply(int a, int b) {
int multiplier = Math.min(abs(a), abs(b));
int multiplicant = (multiplier == abs(a) ? abs(b) : abs(a));
int result = 0;
for (int i = 0; i < multiplier; i = i + 1) {
result = result + multiplicant;
}
if (abs(a) == a) { // a >= 0
if (abs(b) == b) // b >= 0
return result;
else
// b < 0;
return negate(result);
```

```
} else { // a < 0
if (abs(b) == b) // b >= 0
return negate(result);
else
// b < 0;
return result;
}
}
public static int substract(int a, int b) {
return a + negate(b);
}
public static int divide(int a, int b) {
if (b == 0) {
throw new java.lang.ArithmeticException("Divide by 0.");
}
int divident = abs(a);
int divisor = abs(b);
int quotient = 0;

while (divident >= divisor) {
divident = substract(divident, divisor);
quotient = quotient + 1;
}
if (abs(a) == a) { // a >= 0
if (abs(b) == b) // b >= 0
return quotient;
else
// b < 0;
return negate(quotient);
} else { // a < 0
if (abs(b) == b) // b >= 0
return negate(quotient);
else
// b < 0;
return quotient;
}
}
public static void main(String[] args) {

Scanner input=new Scanner(System.in);
System.out.println("enter value of a");
int a=input.nextInt();
System.out.println("enter value of b");
```

```
int b=input.nextInt();
int mul=multiply(a,b);
System.out.println("Multiply="+mul);
int sub=substract(a,b);
System.out.println("Sub="+sub);
int div=divide(a,b);
System.out.println("Div="+ div);
}
}
```

Q35. Write test cases for how to test just the withdrawing functionality from ATM

1. Machine does not accept card and PIN.
2. Machine finds wrong PIN.
3. Machine finds card insertion in wrong way.
4. Machine takes 3 invalid PIN attempt.
5. Invalid account type selected in the menu.
6. Lack of money in the savings account.
7. Expired card inserted in the machine.
8. Machine accepts both Visa and Master card credit and debit cards.
9. Machine accepts card and PIN detail.
10. Machine successfully takes out the money.
11. Machine takes out the balance printout after the withdrawal.
12. Machine logs out of the client session immediately after withdrawal successfully.

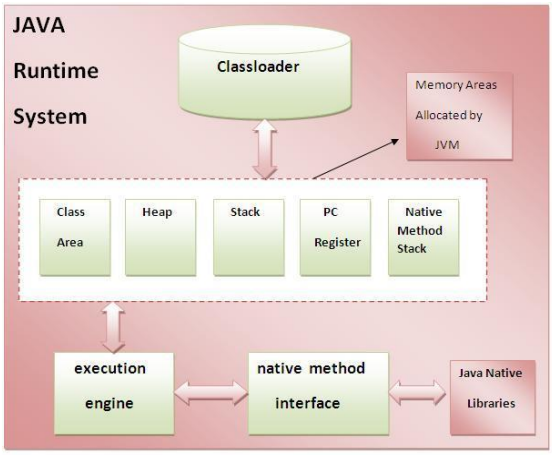
Q36. Write to test scenarios to test Pencil

1.The basic functionality : It should write
2.Boundary conditions: The lead should be present inside the wood
3.Stress conditions: The pencil doesn't break on holding or dropping
4.Usability: The pencil is easy to hold
5.security/safety: Is pencil paint harmful to health

Q37. What is JVM and explain me the Java memory allocation

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

M contains classloader, memory area, execution engine etc.



1) Classloader
Classloader is a subsystem of JVM that is used to load class files.
2) Class(Method) Area
Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.
3) Heap
It is the runtime data area in which objects are allocated.
4) Stack

va Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

ch thread has a private JVM stack, created at the same time as thread.

new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register
PC (program counter) register. It contains the address of the Java virtual machine instruction currently being executed.
6) Native Method Stack
It contains all the native methods used in the application.
7) Execution Engine

contains:

A virtual processor

Interpreter: Read bytecode stream then execute the instructions.

Just-In-Time(JIT) compiler: It is used to improve the performance.JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.Here the term ?compiler? refers to a translator from the instruction set of a Java virtual machine (JVM) to

the instruction set of a specific CPU.

Q38. What is Polymorphism and encapsulation?

Polymorphism :- Polymorphism is an important object-oriented programming concept. Polymorphism means the ability to take more than one form. Polymorphism refers to a programming language's ability to process objects differently depending on their data type or class. More specifically, it is the ability to redefine methods for derived classes.

Encapsulation :- The process of binding data members and functions in a class is known as, encapsulation. Encapsulation is the powerful feature (concept) of object-oriented programming. With the help of this concept, data is not accessible to the outside world and only those functions which are declared in the class, can access it.

Q39. What is method overloading and Method over riding?

If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in java.

Q40. Why string is Immutable?

String is immutable for several reasons, some of them are,

- Security: parameters are typically represented as String in network connections, database connection urls, usernames/passwords etc. If it were mutable, these parameters could be easily changed.
- Synchronization and concurrency: making String immutable automatically makes them thread safe thereby solving the synchronization issues.
- Caching: when compiler optimizes your String objects, it sees that if two objects have same value (a="test", and b="test") and thus you need only one string object (for both a and b, these two will point to the same object).
- Class loading: String is used as arguments for class loading. If mutable, it could result in wrong class being loaded (because mutable objects change their state).

Q41. What is the difference between String and String buffer

1) Mutability: String is immutable (Once created, cannot be modified) while StringBuffer is mutable (can be modified).

2)Performance:While performing concatenations you should prefer StringBuffer over Stringbecause it is faster.
The reason is: When you concatenate strings using String, you are actually creating new object every time since String is immutable.

Q42. What is the difference between array and array list

1. Implementation of array is simple fixed sized array but Implementation of ArrayList is dynamic sized array.
2. Array can contain both primitives and objects but ArrayList can contain only object elements
3. You can't use generics along with array but ArrayList allows us to use generics to ensure type safety.
4. You can use *length* variable to calculate length of an array but *size()* method to calculate size of ArrayList.
5. Array use assignment operator to store elements but ArrayList use *add()* to insert elements.

Q43. What is the difference between hash map and Hash table

HashMap	Hashtable
) HashMap is non synchronized. It is not-thread safe and can't be shared between many threads without proper synchronization code.	ashtable is synchronized. It is thread-safe and can be shared with many threads.
) HashMap allows one null key and multiple null values.	ashtable doesn't allow any null key or value.
) HashMap is a new class introduced in JDK 1.2.	ashtable is a legacy class.
) HashMap is fast.	ashtable is slow.
) We can make the HashMap as synchronized by calling this code	ashtable is internally synchronized and can't be

Map m = Collections.synchronizedMap(hashMap);	unsynchronized.
) HashMap is traversed by Iterator.	ashtable is traversed by Enumerator and Iterator.
) Iterator in HashMap is fail-fast.	numerator in Hashtable is not fail-fast.
) HashMap inherits AbstractMap class.	ashtable inherits Dictionary class.

Q44. What is a vector in Java?

Vector implements a dynamic array. It is similar to ArrayList, but with two differences −

- Vector is synchronized.
- Vector contains many legacy methods that are not part of the collections framework.

Vector proves to be very useful if you don't know the size of the array in advance or you just need one that can change sizes over the lifetime of a program.

Q45. What is set in java?

- Set is an interface which extends Collection. It is an unordered collection of objects in which duplicate values cannot be stored.
- Basically, Set is implemented by HashSet, LinkedSet or TreeSet (sorted representation).
- Set has various methods to add, remove clear, size, etc to enhance the usage of this interface

Q46. What is an abstract class?

A class that is declared with abstract keyword, is known as abstract class in java. It can have abstract and non-abstract methods (method with body).

```
abstract class Bike{
                abstract void run();
}
```

```
class Honda4 extends Bike{
		void run(){
			System.out.println("running safely..");
		}
		public static void main(String args[]){
			Bike obj = new Honda4();
			obj.run();
		}
}
```

Q47. What is an interface?

An interface in java is a blueprint of a class. It has static constants and abstract methods.

The interface in java is a mechanism to achieve abstraction. There can be only abstract methods in the java interface not method body. It is used to achieve abstraction and multiple inheritance in Java.

Java Interface also represents IS-A relationship.

It cannot be instantiated just like abstract class.

```
interface printable{
void print();
}
class A6 implements printable{
public void print(){System.out.println("Hello");}

public static void main(String args[]){
A6 obj = new A6();
obj.print();
}
}
```

Q48. Why Java is Platform independent?

Java is a platform independent programming language, Because when you install jdk software on your system then automatically JVM are installed on your system. For every operating system separate JVM is available which is capable to read the .class file or byte code. When we compile your Java code then .class file is generated by javac compiler these codes are readable by JVM and every operating system have its own JVM so JVM is platform dependent but due to JVM java language is become platform independent.

Q49. What are access modifiers? Give me an example?

There are two types of modifiers in java: access modifiersand non-access modifiers.

The access modifiers in java specifies accessibility (scope) of a data member, method, constructor or class.
There are 4 types of java access modifiers:

1. private
2. default
3. protected
4. public

There are many non-access modifiers such as static, abstract, synchronized, native, volatile, transient etc.

```
class A{
private int data=40;
private void msg(){System.out.println("Hello java");}
}
```

```
public class Simple{
public static void main(String args[]){
A obj=new A();
System.out.println(obj.data);//Compile Time Error
obj.msg();//Compile Time Error
}
}
```

Q50. What are java exceptions? Give me an example

- An exception is an event, which occurs during the execution of a program, that interrupts the normal flow of the program. It is an error thrown by a class or method reporting an error in code.
- The *'Throwable'* class is the superclass of all errors and exceptions in the Java language
- Exceptions are broadly classified as *'checked exceptions'* and *'unchecked exceptions'*. All RuntimeExceptions and Errors are unchecked exceptions. Rest of the exceptions are called checked exceptions. Checked exceptions should be handled in the code to avoid compile time errors.
- Exceptions can be handled by using *'try-catch'* block. Try block contains the code which is under observation for exceptions. The catch block contains the remedy for the exception. If any exception occurs in the try block then the control jumps to catch block.
- If a method doesn't handle the exception, then it is mandatory to specify the exception type in the method signature using *'throws'* clause.

- We can explicitly throw an exception using *'throw'* clause.

```
public class MyExceptionHandle {
public static void main(String a[]){
try{
for(int i=5;i>=0;i--){
System.out.println(10/i);
}
} catch(Exception ex){
System.out.println("Exception Message: "+ex.getMessage());
ex.printStackTrace();
}
System.out.println("After for loop...");
}
}
```

Example Output

```
2
2
3
5
10
Exception Message: / by zero
java.lang.ArithmeticException: / by zero
at
com.myjava.exceptions.MyExceptionHandle.main(MyExceptionHandle.java:12
)
After for loop...
```

Q51. What is the difference between throws and throwable?

throws is also a keyword in java which is used in the method signature to indicate that this method may throw mentioned exceptions. The caller to such methods must handle the mentioned exceptions either using try-catch blocks or using throws keyword. Below is the syntax for using throws keyword.

```
class ThrowsExample
{
void methodOne() throws SQLException
{
//This method may throw SQLException
}

void methodTwo() throws IOException
{
```

```
//This method may throw IOException
}

void methodThree() throws ClassNotFoundException
{
//This method may throw ClassNotFoundException
}
}
```

Throwable is a super class for all types of errors and exceptions in java. This class is a member of java.lang package. Only instances of this class or it's sub classes are thrown by the java virtual machine or by the throw statement. The only argument of catch block must be of this type or it's sub classes. If you want to create your own customized exceptions, then your class must extend this class.

Below example shows how to create customized exceptions by extending java.lang.Throwable class.

```
class MyException extends Throwable
{
//Customized Exception class
}

class ThrowAndThrowsExample
{
void method() throws MyException
{
MyException e = new MyException();

throw e;
}
}
```

Q52. What is the difference between Error and exception?

An error is an irrecoverable condition occurring at runtime like out of memory error. These kind of jvm errors cannot be handled at runtime.

Exceptions are because of condition failures, which can be handled easily at run time.

1) Recovering from Error is not possible. The only solution to errors is to terminate the execution. Where as you can recover from Exception by using either try-catch blocks or throwing exception back to caller.

2) You will not be able to handle the Errors using try-catch blocks. Even if you handle them using try-catch blocks, your application will not recover if they happen. On the other hand, Exceptions can be handled using try-catch blocks and can make program flow normal if they happen.

3) Exceptions in java are divided into two categories – checked and unchecked. Where as all Errors belongs to only one category i.e unchecked.
4) Compiler will not have any knowledge about unchecked exceptions which include Errors and sub classes of RunTimeException because they happen at run time. Where as compiler will have knowledge about checked Exceptions. Compiler will force you to keep try-catch blocks if it sees any statements which may throw checked exceptions.
5) Exceptions are related to application where as Errors are related to environment in which application is running.

Q53. What is the difference between Error, throwable and exception?
The *'Throwable'* class is the superclass of all errors and exceptions in the Java language.
An error is an irrecoverable condition occurring at runtime like out of memory error. These kind of jvm errors cannot be handled at runtime.
An exception is an event, which occurs during the execution of a program, that interrupts the normal flow of the program

Q54. What are collection APIs, give me an example

Collections in java is a framework that provides an architecture to store and manipulate the group of objects.
All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc. can be performed by Java Collections.
Java Collection simply means a single unit of objects.

Examples include
interfaces (Set, List, Queue, Deque etc.) and
classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet etc).

Q55. What is the difference between final and finally?
Final is a keyword.
Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed.

Java finally is a block that is used *to execute important code* such as closing connection, stream etc.
Java finally block is always executed whether exception is handled or not.
Java finally block follows try or catch block.

Q56. Will java supports multiple inheritance?
Java supports multiple inheritance through interfaces only. A class can implement any number of interfaces but can extend only one

class. Multiple inheritance is notsupported because it leads to deadly diamond problem.

Q57. What are the different types of interface?

Different types of interfaces supported by java are Set, List, Queue, Deque.

Q58. What are wrapper class? Give me an example

Each of Java's eight primitive data types has a class dedicated to it. These are known as wrapper classes because they "wrap" the primitive data type into an object of that class. The wrapper classes are part of the java.lang package, which is imported by default into all Java programs.

Wrapper class in java provides the mechanism *to convert primitive into object and object into primitive*.

Example:

```
public class WrapperExample1{
public static void main(String args[]){
    //Converting int into Integer
    int a=20;
    Integer i=Integer.valueOf(a);//converting int into Integer
Integer j=a;//autoboxing, now compiler will write Integer.valueOf(a) internally
System.out.println(a+" "+i+" "+j);
}}
```

Q59. What is boxing and unboxing in Java? Explain with an example

The automatic conversion of primitive data types into its equivalent Wrapper type is known as boxing and opposite operation is known as unboxing.

No need of conversion between primitives and Wrappers manually so less coding is required.

```
class BoxingExample1{
public static void main(String args[]){
int a=50;
Integer a2=new Integer(a);//Boxing

Integer a3=5;//Boxing

System.out.println(a2+" "+a3);
}
}
```

The automatic conversion of wrapper class type into corresponding primitive type, is known as Unboxing.

```
class UnboxingExample1{
```

```
public static void main(String args[]){
Integer i=new Integer(50);
int a=i;

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

Q60. Explain for each loop

The for-each loop introduced in Java5. It is mainly used to traverse array or collection elements. The advantage of for-each loop is that it eliminates the possibility of bugs and makes the code more readable.

Syntax: for(data_type variable : array | collection){
}
Example class ForEachExample1{

```
public static void main(String args[]){
int arr[]={12,13,14,44};
for(int i:arr){
System.out.println(i);
}
}
}
```

Q61. What are iterators, explain with an example

Iterator is used for iterating (looping) various collection classes such as HashMap, ArrayList, LinkedList etc.

```
import java.util.ArrayList;
import java.util.Iterator;

public class IteratorDemo1 {

public static void main(String args[]){
ArrayList names = new ArrayList();
names.add("Chaitanya");
names.add("Steve");
names.add("Jack");

Iterator it = names.iterator();

while(it.hasNext()) {
String obj = (String)it.next();
System.out.println(obj);
}
}
```

}

Q62. write an algorithm to reverse first 3 numbers, then next 3 numbers, then next 3 numbers, the number will be based on var k. Array = [3,2,4,7,0,3,1,5,8, 4] k=3 OutPut = [4,2,3,3,0,7,8,5,1,4]

```
package hackathon;

import java.util.Scanner;

public class Q62 {
private static int[] traverseArray(int[] a, int k) {
int i;
int rem = a.length % k; //to fig out if any elements will be left at the end
for(i = 0; i < a.length ; i = i + k) {
a = reverseArray(a, i, i+k-1);
}
if(rem != 0) {
reverseArray(a, a.length - rem, a.length - 1);
}
return a;
}
private static int[] reverseArray(int[] a, int start, int end) {
int temp, i ,j;
for(i = start, j = end; i < j && j < a.length ; i++, j--) {
temp = a[i];
a[i] = a[j];
a[j] = temp;
}
return a;
}
public static void main(String[] args) {
// TODO Auto-generated method stub
Scanner input=new Scanner(System.in);
System.out.println("enter your array size");
int size=input.nextInt();
int[] array=new int[size];
int[] resArray=new int[size];
System.out.println("please enter "+size+" values");
for(int i=0;i<array.length;i++){
array[i]=input.nextInt();
}
System.out.println("enter your value for k");
```

```
int k=input.nextInt();
int[] resultArray=traverseArray(array,k);
for(int i=0;i<array.length;i++){
System.out.print(resultArray[i]+ " ");
}
}
}
```