JAVA PROGRAMMING

- ✓ **Java** is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language.
- ✓ **Applications:**
  1. Desktop Applications such as acrobat reader, media player, antivirus, etc.
  2. Web Applications such as irctc.co.in, javatpoint.com, etc.
  3. Enterprise Applications such as banking applications.
  4. Mobile
  5. Embedded System
  6. Smart Card
  7. Robotics
  8. Games, etc.

  1)Standalone Application
  2)Web Application
  3) Enterprise Application
  4) Mobile Application
- ✓ **JAVA Versions**

1) Java SE (Java Standard Edition

2) Java EE (Java Enterprise Edition)

3) Java ME (Java Micro Edition)

4) JavaFX

- ✓ **Features :**
  1. Simple
  2. Object-Oriented
  3. Portable
  4. Platform independent
  5. Secured
  6. Robust
  7. Architecture neutral
  8. Interpreted
  9. High Performance
  10. Multithreaded
  11. Distributed
  12. Dynamic
- ✓ **Components of Java Architecture**

The Java architecture includes the three main components:

1. Java Virtual Machine (JVM)
2. Java Runtime Environment (JRE)
3. Java Development Kit (JDK)

- ✓ **Java Virtual Machine (JVM)**

JVM is an abstract machine that provides the environment in which Java bytecode executed. JVM converts the byte code into machine code.

- ✓ **Java Runtime Environment (JRE)**

  Java Run-time Environment (JRE) is the part of the Java Development Kit (JDK). It is a freely available software distribution which has Java Class Library, specific tools, and a stand-alone JVM. It is the most common environment available on devices to run java programs. The source Java code gets compiled and converted to Java bytecode.

- ✓ **Java Development Kit (JDK)**

  It is a software development environment used in the development of Java applications and applets. Java Development Kit holds JRE, a compiler, an interpreter or loader, and several development tools in it.

1) HELLOWORLD PROGRAM:

```
public class helloworld {    // class name should be as the filename
  public static void main(String[] args) {
     /*public means it can be access by anyone
     like an open source and void means it will
     not going to return any values
      */
     System.out.println("HELLO WORLD!");// ln means it will go to next line if we are not using the ln means it will go to the next by next
  }
}
```

2) **INPUT PROGRAM**:

```
import java.util.Scanner; // or else we can use the * instead of Scanner because in * only all will present
public class inputprogram {
   public static void main(String[] args) {
      int a=10;
      /* compile time input-means the input that was given  directly
      compile time error=when we made any mistakes in the syntax while writing the code
       */
      System.out.println(a);
      /*Run time input-means the input was given by the user
      as per their requirement also called as user input
      Run time error-when we made any mistakes in the logic of the code while writing the program
       */
      Scanner input = new Scanner(System.in);
```

```
        System.out.print("enter the value of the n :");
        int n = input.nextInt();// for next int we are using the camel case rule
        System.out.print("the value of n is : "+n);// just print instead of using prinln it will
    give the result one after another
    //      int a='A';
    //      System.out.println(a); // output as 65 because the ASCII value of the A is 65.
    //      char a='s';
    //      System.out.println(a);
    //      java follows the unicode& ascii system while the c language follows the ascii code
    system
        System.out.println("नमस्ते");
      }
    }
```

3) DATATYPES IN JAVA

```
import java.util.*;

public class datatypesinjava{
   public static void main(String[] args) {
      Scanner obj=new Scanner(System.in);//new keyword will allocate the memory space
    for the variables
    //      primitive datatypes
    //      int a=obj.nextInt();              //integer datatype-size is 4 bytes
    //      byte b=obj.nextByte();             // byte size-8 bits
    //      float c=obj.nextFloat();          // c=23.45f //float occupies 4 bytes
    //      double d=obj.nextDouble();         // d=34.376537367 // double occupies 8 bytes
    //      long e=obj.nextLong();            //e=344567785L // long occupies 8 bytes
    //      boolean d=obj.nextBoolean();      // boolean values are true/false  // boolean size is
    1 bit
    //      char c =obj.next().charAt(0);     // char size is 2 in java but 1 in c language.
    //      System.out.println(c);
    //        non primitive data types(Also called as derived data types that are derived from
    primitive)
        String name=obj.nextLine();
        System.out.println(name);
      }
    }
```

4) TYPECASTING:

```
import java.util.*;

public class typecasting{

   public static void main(String[] args) {
      Scanner input=new Scanner(System.in);
    //      byte a=input.nextByte();
    //      byte b=input.nextByte();
    //      byte c=(byte)(a+b); //EXPLICT TYPE CASTING-we have to do the typecasting
        /*type casting is done here because when we add a and b
```

```
        it gives integer but the c is in byte so it will throw an error.
        to overcome it we are doing the type casting  */
        /*IMPLICIT TYPE CASTING-compiler itself will do the type casting process
        for example if we use the Shorthand operators then there is no need of doing the type
    casting process */
        float v=input.nextFloat();
        float d=(int)(v);
        System.out.println(d);
    }
}
```

5) VARIBALES:

```
//import java.util.Scanner;  // variables in the java are the entities that can store the data
//public class variables{
//    public static void main(String[] args) {
//        Scanner input = new Scanner(System.in);
//         int a=input.nextInt();  // local variable declaration (variables are declared inside the
block)
//         System.out.println(a);  // int a=100;
//    }
//}
public class variables{
   int b=120; // declaration of instance variable(variables are declared outside the block or
method
   public static void main(String[] args) {
      System.out.println();
   }
}
//public class variables{
//    static int f=100; // static keyword is used to access the data instead of creating any object
//    /* declaration of static variable(variables are declared same
//    as instance variable but while declaration we have to use static keyword over there
//    */
//    public static void main(String[] args) {
```

```
//    System.out.println(f);
//   }
//}
```

6) OPERATORS:

```
//import java.util.Scanner;
public class operators {
   public static void main(String[] args) {
//       Scanner sc = new Scanner(System.in);
//       int a = sc.nextInt();
//       int b = sc.nextInt();
//    operator types-Arithematic operator
//       System.out.println(a + b);// addition
//       System.out.println(a - b); // subraction
//       System.out.println(a * b); // multiplication
//       System.out.println(a / b);
//       /* division rule-divide int/int=int as result,int/float=float,
//       int/double=double as result-for example 2/9=0.23=but it gives
//       only 0 as ouput as per the division rule/*
//        */
//       System.out.println(a % b);//modulus operation=a is less than b then the ouput is 'a'
value as per definition it gives remainder value
//   }
//}
//       assignment operator
//       int c = 100; // assign the 100 value to the variable name c,so = is the assignment
operator
//       System.out.println(c);
//   }
//}
       /*Relational operators-this will return a boolean values if condition
        is true it will return true,otherwise false/*
        */
//       System.out.println(a < b); //2<3=true
//       System.out.println(a > b); //2>3=false
//       System.out.println(a <=b); //2<=3 -->true
//       System.out.println(a >=b); //2>=3 -->false
//       System.out.println(a!= b); //2!=3= true
//         System.out.println(a==b); // it will compare the values and returns the boolean
value
//   }
//}
       //logical operators(syntax)=condition1 && condition2
       // -&&(and),||(or),!=negation means the values will get inverted
       // logical operators are used for multiple condition checking and also it will not take
time to check all
```

```
//        System.out.println(a>b && a<b &&a!=b&& a%b==0);//if both conditions are true only it returns true otherwise all are false-logical AND
//        System.out.println(a>b || a<b);// if any one condition is true it will return true otherwise false-logical OR
//        System.out.println(!(a>b&& a<b));//it will invert the result-logical NOT
//    }
//}
    /*Bitwise operators
    &(AND),
    /(OR/inclusive OR),
    ^(ex-or/exclusive OR),
    >>(Right shift),
    <<(Left shift),
    >>>(unsigned right shift with zero fill),
    ~(tilt operator)
    */
//        System.out.println(a & b); // compiler itself convert the decimal into binary and give the result in the decimal format.(.) as per my trick
//        System.out.println(a | b); // or is nothing but (+) as per my trick(0 0=0 remaining all are 1's)
//        System.out.println(a ^ b); // (^) denotes if 1 1=0,0 0=0,1 0=1,0 1=1(same inputs=o,different inputs=1)
//        System.out.println(a >> b); // 8>>4=1000 we have to remove the 4 bits from right side from the 1000 it becomes 0,method 2:8/2^4=8/16=0 as per division rule
//        System.out.println(a << b); // 8<<4=1000 we have to add the 4 bits of 0's to the right side so it becomes 10000000=128,method 2:8*2^4=128
//        System.out.println(a >>> b);// 8>>>4=00001000 here the left most bits are filled with zeros=0
//        System.out.println(a&(~b));//0 becomes 1 and 1 becomes 0=1000=0111 is the negation operation
//    }
//}
    /*unary operators are also called as increment and decrement operators
    increment opeartors and decrement
    1)pre increment and post increment(++a,a++)
    2)pre decrement and post decrement(--a,a--)
     */
//        int a = 12;//13//14
//        int b = ++a;//pre increment it will first increment the a value after the incremental value will be stored in the b//output=13
//        int c = a++;//post increment it will first write the updated a value and store it in the c then it will update the a value//o/p=13
//        System.out.println(a + " " + b + " " + " " + c);// 14 13 13
//    }
//}
//        int s = 123;//122//121
```

```
//        int t = --s;//pre decrement it will decrement the s value and updated s stored in the
t//output=122
//        int r = s--;//post decremnt it will write the updated s value and stored in the r and
then it will update the s value//op=122
//        System.out.println(s + " " + t + " " + r);//121 122 122
//    }
//}
//        int s = 90;
//        int m = ++s + s-- + --s + s++;
//        System.out.println(s + " " + m);
//    }
//}
    /*ternary operator/conditional operator
    syntax=(condition)? true part:false part;
     */
//        int number = 100;
//        String c = (number % 2 == 0) ? "even" : "odd";// After question mark is true part
and after the colon is the false part
//        System.out.println(c);
        int a=10,b=12,c=10;
        String max=(a>b&&a>c)?"a is the largest":(b>a&&b>c)?"b is the largest":"c is the
largest";// we can give the multiple conditions inside the ternary operator
        System.out.println(max);
    }
}
```

7) CONTROL STATEMENTS:

```
import java.util.*;
public class Conditionalstatements { // control statements try to control the flow of
program
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
//    /*conditional/branching statements
//    -->if condition
//    -->else if condition
//    -->nested if condition
//    -->else if ladder condition
//    -->switch/selection statement
//    */
//  if condition:
//        if (a >= 2000) { //a=2000 o/p:2100
//            a += 100;//a=a+100(shorthand operators mainly used for assign the operation
and automatic type casting
//        }
//        System.out.println(a);
//        System.out.println("welcome");
```

```
//    }
//}
//    if(a!=0) {// if we have one statement we dont want to use curly braces and if it has multiple conditions we have to use the curly braces
//        System.out.println(true);//if the condition is true it will execute this or else it will excute the remaining code
//    }
//    System.out.println("virat kohli-The father of pakisthan");
//    }
//}
//  else if condition:example:positive number
//        if (a > 0) {
//            System.out.println("positive number");
//        } else {
//            System.out.println("negative number");
//        }
//    }
//    }
//  example:even or odd number
//        if(s%2==0){
//        System.out.println("even number");
//        }
//        else {
//        System.out.println("odd number");
//        }
//    }
//}
//  example:leap year or not a leap year
//        if(s%400==0||s%4==0&&s%100!=0){
//            System.out.println("leap year");
//        }
//        else{
//            System.out.println("not a leap year");
//        }
//    }
//}
//    Nested if condition=if condition inside an another if condition
//        if (s != 10) {
//            if (s > 10) {
//                if (s % 2 == 0) {
//                    if (s < 0) {
//                        System.out.println("welcome to the match");
//                    }
//                    System.out.println("welcome to the india vs pakisthan match ");
//                }
//            }
```

```
//        }
//    }
//}
//  else if ladder condition:grading system
//        if (s >= 90) {
//            System.out.println("s grade");
//        } else if (s >= 80 && s < 90) {
//            System.out.println("A grade");
//        } else if (s >= 70 && s < 80) {
//            System.out.println("B grade");
//        } else {
//            System.out.println("fail");
//        }
//    }
//}
//example:largest of three numbers
//        int b = input.nextInt();
//        int c = input.nextInt();
//        if (a > b && a > c) {
//            System.out.println("a is the largest");
//        } else if (b > a && b > c) {
//            System.out.println("b is the largest");
//        } else {
//            System.out.println("c is the largest");
//        }
//    }
//}
/*switch statement or selection statement
    synatx:switch(expression/value){
    case casevalue:
       //statements;
       break; //optional
    .........
    default:
      //statements that none of the above one is true
      break;
      }
      */
//  example:Basic atm
//        switch (a) { //switch statement will only allow the char and int datatypes remaining
like float,string,double not allowed in switch statement
//            case 1:
//                System.out.println("withdrawal");
//                break; // break will help us to terminate from the block
//            case 2:
```

```
//              System.out.println("deposit"); // here no break is there so it will go to the next
condition and print the statement
//          case 3:
//              System.out.println("balance checking");
//              break;
//          case 4:
//              System.out.println("pin chnage");
//              break;
//          default:
//              System.out.println("enter the correct option");
//              break;
//      }
//  }
//}
//  example:months program
//      switch (month) {
//          case 1:
//              System.out.println("january");
//              break;
//          case 2:
//              System.out.println("february");
//              break;
//          case 3:
//              System.out.println("march");
//          case 4:
//              System.out.println("april");
//              break;
//          case 5:
//              System.out.println("may");
//              break;
//          default:
//              System.out.println("enter the valid value");
//              break;
//      }
//  }
//}
//  example for switch
//      char A = input.next().charAt(0);
//      switch (A) {
//          case 89:
//              System.out.println("virat");
//              break;
//          case 65:
//              System.out.println("A");
//              break;
//          case 65+1:
```

```
//                System.out.println("hello everyone");
//                break;
//            default:
//                System.out.println("enter the valid expression");
//                break;
//        }
//    }
//}
//example programs-calculator program
//        double b = input.nextDouble();
//        char S = input.next().charAt(0);
//        double result=0;
//        switch (S) {
//            case '+':
//                result = a + b;
//                break;
//            case '-':
//                result = a - b;
//                break;
//            case '*':
//                result = a * b;
//                break;
//            case '/':
//                if (b != 0) {
//                    result = a / b;
//                } else{
//                    System.out.println("error");
//                }
//                break;
//            default:
//                System.out.println("enter the valid operation");
//                return;
//        }
//        System.out.println(result);
//    }
//}
//  basic program
//        if (a < 2000 && a % 2 != 0 || a < 1000) {
//            System.out.println("yes");
//        }
//    }
//}
//head or tail program
//        char A = input.next().charAt(0);
//        char B=input.next().charAt(0);
//        if (A == 'h' || B == 'H') {
```

```
//            System.out.println("head");
//        }
//        else if (A == 't' || B == 'T') {
//            System.out.println("Tail");
//        }
//    }
//}
//    using switch also we can do the same process
//        switch (A) {
//            case 'h':
//                System.out.println("head");
//                break;
//            case 'H':
//                System.out.println("Head");
//                break;
//            case 't':
//                System.out.println("tail");
//                break;
//            case 'T':
//                System.out.println("Tail");
//                break;
//        }
//    }
//}
//example program=prime number or not
        int flag=0;
        if(n<=1){
            System.out.println(n + " is not a prime number");
        }
        else{
            for(int i=2;i<n;i++){
                if(n%i==0){
                    flag=1;
                    break;
                }
            }
            if(flag==0){
                System.out.println(n+ " is a prime number");
            }
            else{
                System.out.println(n+ " is not a prime number");
            }
        }
    }
}

import java.util.Scanner;
```

```
public class loopingstatements {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        /*also called as iterative statements
        -->for loop=if we know the range of values we can use for loop
        -->while loop=it will checks the condition after then only it will prints the statements
        -->do while loop=it prints the statements first after that only it checks the condition
         */
    /*for loop syntax
    for(intilaization;condition;increment/decrement operators){// number of iterations is known to us
        statements
        }
    /*
//        for (int i = 0; i <= 10; i++) { //printing the 0 to 10 numbers
//            System.out.println(i);
//        }
//    }
//}
//        int sum = 0;
//        for (int i = 1; i < 11; i++) {
//            sum+=i;//sum of the elements from 1 to 11
//        }
//        System.out.println(sum);
//    }
//}
//  mutliplication of a number
//        for (int i = 1; i <= 10; i++) {
//            System.out.println(n + "*" + i + "=" + n * i);
//        }
//    }
```

```
//}
//    declaration of for loop in different ways
//        int i = 0;
//        for (; i <= 10; ) {
//            System.out.println(i);
//            i++;
//        }
//    }
//}
//    example programs
//        for (int i = 0; i < 11; i++) {
//            if (i % 2 == 0) {
//                System.out.println(i); //even numbers between the 0 and 11
//            }
//        }
//    }
//}
//        int start = 20, end = 90;
//        for (start = 20; start <= end; start++) {
//            if (start % 2 != 0) {
//                System.out.println(start);//odd numbers between 20 to 90
//            }
//        }
//    }
//}
//        for (int i = 1; i <=n; i++) {
//            if (n % i == 0) {
//                System.out.print(i+" ");//factors of a number
//            }
//        }
```

```
//    }
//}
    /*-->while loop syntax or also called as entry control loop
    intilization;
    while(condition){ //number of iterations is unknown
        //statements; //it prints the statements repeatedly until it is proved as false
        updation;
     */
//        int i = 0;
//        while (i <= 10) {
//            System.out.println(i);//printing the 0 to 10 numbers using while loop
//            i++;
//        }
//    }
//}
////example programs for while loop
//        int n = s.nextInt();
//        int count = 0;
//        while (n != 0) { //n>0 both are correct only
//            n = n / 10;
//            count++;
//        }
//        System.out.println(count); //count of digits
//    }
//}
//        int sum = 0;
//        while (n!=0) {
//            int r = n % 10;
//            sum += r;
//            n = n / 10;
```

```
//        }
//        System.out.println(sum);//sum of digits of the given number
//    }
//}
//-->do while loop syntax
//    intilization;
//    do {
//        //statements;
//        updation;
//    }
//    while(condition);
//        int i = 0;
//        do {
//            System.out.println(i);//it will print the i=0 first and then based on the condition it will print the statements
//            i++;
//        }
//        while (i<=n&&i!=0);//even though here i gave the condition like i not equal to 0 but it will print because that is the working of the do while loop
//    }
//}
//    nested for loop is also called as enhanced for loop =for loop inside the for loop
//        for (int i = 1; i <= n; i++) { //outer loop
//            for (int j = 1; j <= n; j++) { //inner loop
//                if (i == j) {
//                    break;// break inner loop
//                }
//                if (i == 2 && j == 3) {
//                    break;//break outer loop
//                }
//                System.out.println(i + " " + j);
```

```
//            }
//            System.out.println("j loop break");
//        }
//    }
//}
    /*-->Jumping statements=mainly used to skip and break/terminate the statements from one to another
    -->break=Mainly used to terminate from the loop or switch/selection statements
    -->continue=mainly used to skip the current loop and proceed to next loop
    -->return=returns the values
    */
//    program for break
//        for (int i = 0; i <= n; i++) {
//            if (i == 1) {
//                break;
//            }
//            System.out.println(i);
//        }
//    }
//}
//    program for continue loop
//        for (int i = 0; i <= n; i++) {
//            if (i % 2 == 0) { //here i gave the even numbers condition
//                continue; //but i want the odd numbers to be printed so i used continue statement to skip the even numbers condition
//            }
//            System.out.println(i);
//        }
//    }
//}
//  return statement it will return some values
```

```
//     return the values
//example program=for a given number calculate the sum of even digits and sum of odd digits
//        int odd = 0, even = 0;
//        while (n!=0) {
//            int r = n % 10;
//            if (r % 2 == 0) {
//                even = even + r;
//            }
//            else {
//                odd = odd + r;
//            }
//            n = n / 10;
//        }
//        System.out.println(even);
//        System.out.println(odd);
//    }
//}
//        int n = s.nextInt();
//        int evensum = 0;
//        while (n != 0) {
//            int r = n % 10;
//            if (r % 2 == 0) {
//                evensum += r;
//            }
//            n=n/10;
//        }
//        System.out.println(evensum);//sum of the even numbers in the number
//    }
//}
//        int n = s.nextInt();
```

```
//        int oddsum = 0;
//        while (n != 0) {
//            int r = n % 10;
//            if (r %2 != 0) {
//               oddsum+= r;
//            }
//            n=n/10;
//        }
//        System.out.println(oddsum);//sum of the odd elements
//    }
//}
// Factorial of a number
        int n = s.nextInt();
//        int fact=1;
//        for(int i=1;i<=n;i++){
//            fact=fact*i;
//        }
//        System.out.println(fact);
//    }
//}
//Fibonacci of a number
        int n1 = 0, n2 = 1, n3;
        System.out.println(n1);
        System.out.println(n2);
        for (int i = 2; i < n; i++) {
            n3 = n1 + n2;
            System.out.println(n3+" ");
            n1 = n2;
            n2 = n3;
        }
```

```
    }
}
```

8) ARRAYS:

```
import java.util.*; //if we want to use the inbulit functions of array then use the Arrays instead of *
public class arrays {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        /*Arrays concept and syntax
        Arrays=Collection of the similar datatypes where we can store 'n' number of data inside the single variable
        array syntax: array index is dtarting from the 0
        //a[0]=0 is the index(where the data is located),a[0] is the element inside the position 0
        -->1dimensional array syntax
        datatype arrayname=new datatype[size];
        -->2dimensional array(collection of 1D)
        datatype arrayname[][]=new dataype[row][column];
        -->3D array syntax
        datatype arrayname[][][]=new datatype[n][row][column];
         */
    //  compile time input for array
    //      int a[]=new int[6];
    //      int a[]={1,3,4,5,6};
    //  Run time input for array
    //      int a[]=new int[n];
    //      for(int i=0;i<n;i++){
    //          a[i]=sc.nextInt();
    //      }
    //   }
    //}
    //      int a[]=new int[n];
    //      for(int i=0;i<n;i++){
    //          a[i]=sc.nextInt();
    //      }
    //      for(int i=0;i<n;i++){ // instead of using i++ we can use i=i+2
    //          if(a[i]%2==0){    //To print the even numbers in the array
    //              System.out.println(a[i]);
    //          }
    //      }
    //   }
    //}
    //      int a[]=new int[n];
    //      for(int i=0;i<n;i++){
    //          a[i]=sc.nextInt();
```

```
//        }
//        for(int i=1;i<n;i++){
//            if(a[i]%2!=0){    //To print the odd numbers in the array
//                System.out.println(a[i]);
//            }
//        }
//    }
//}
//  Compile time input for 2D array
//        int row = 3, column = 3;
//        int a[][]= {   {1,2,3}, // first method
//                       {4,5,6},
//                       {7,8,9}    };
//        int e[][] = new int[row][column];//second method
//        e[0][0] = 12;
//        e[0][1] = 23;
//        e[0][2] = 34;
//        e[1][0] = 45;
//        e[1][1] = 56;
//        e[1][2] = 67;
//        e[2][0] = 78;
//        e[2][1] = 89;
//        e[2][2] = 90;
//        for (int i = 0; i < row; i++) {
//            for (int j = 0; j <column; j++) {
//                System.out.print(e[i][j] + " ");
//            }
//            System.out.println();
//        }
//    }
//}
//  Run time input for 2D array
//        int r = sc.nextInt(), c = sc.nextInt();//user input
//        int a[][] = new int[r][c];
//        for (int i = 0; i < r; i++) {
//            for (int j = 0; j < c; j++) {
//                a[i][j] = sc.nextInt();
//            }
//        }
//        for (int row[] : a) {
//            for (int col : row) {
//                System.out.print(col + " ");
//            }
//            System.out.println();
//        }
//    }
```

```
//}
//  Example programs for array
//        int a[] = new int[n];
//        for (int i = 0; i < n; i++) {
//            a[i] = sc.nextInt();
//        }
//        int evensum = 0,oddsum=0;
//        for (int i = 0; i < n; i++) {
//            if (a[i] % 2 == 0) {
//                evensum = evensum + a[i];//sum of the even numbers in the array
//            }
//            else{
//                oddsum=oddsum+a[i];//sum of the odd elements in the array
//            }
//        }
//        System.out.println(evensum);
//        System.out.println(oddsum);
//
//    }
///}
//        example program=even index odd value sum and odd index even value sum
//        int a[] = new int[n];
//        for (int i = 0; i < n; i++) {
//            a[i] = sc.nextInt();
//        }
//        int even = 0, odd = 0;
//        for (int i = 0; i < n; i++) {
//            if (i % 2 == 0) {  //even index
//                if (a[i] % 2 != 0) { //odd value
//                    odd = odd + a[i];
//                }
//            } else {
//                if (i % 2 != 0) { //odd index
//                    if (a[i] % 2 == 0) { //even value
//                        even = even + a[i];
//                    }
//                }
//            }
//        }
//        System.out.println(odd);
//        System.out.println(even);
//    }
//}
/*  inbuilt functions for the arrays in java  //Always use the camel case rule for any inbuilt functions in the java
```

--> length=Mainly used to find the number of the elements in the array//in arrays for length don't use() for remaining we have to use() for length
--> toString=Mainly used to convert the arrays into string list
--> sort=To sort the array in the ascending order
--> fill=To fill the any values inside the array
--> binarySearch=It will return the index values as output(Binary search is possible if and only the array is in sorted format)
   syntax for binarySearch in arrays
   Arrays.binarySearch(a,4)//a=array is to search and 4=key value
   Arrays.binarySearch(a,2,9,3)//2=fromIndex,9=toIndex,3=key value
--> equals=it will check whether both are equal or not
 ** DISADVANTAGES OF THE ARRAYS
--> Collection of the similar datatypes
--> we cannot change the size of array once we fixed
--> If you want to grow the size of the array then go with ArrayList
 */
```
//        int[] a = {1, 2, 9, 7, 3, 6};
//        Arrays.sort(a);//sort method
//        System.out.println(Arrays.toString(a));//convert the array into string list
//    }
//}
//        System.out.println(a.length);//it won't pass the parameter
//        Arrays.fill(a,10);
////         System.out.println(Arrays.toString(a));
//         Arrays.sort(a);
////         System.out.println(Arrays.binarySearch(a, 6));
//        System.out.println(Arrays.binarySearch(a,2,5,3));//it will return the index value
//    }
////}
//        int[] arr = {1, 2, 9, 5, 6, 3};
//        System.out.println(Arrays.equals(a, arr));//it will return the boolean value like true,false
//    }
//}
//        int sum = 0;
//        for (int i = 0; i < n; i++) {
//            sum = sum + a[i]; //sum of the elements in the array
//        }
//        System.out.println(sum);
//    }
//}
//        int M = sc.nextInt();
//        int a[] = new int[M];
//        for (int i = 0; i < n; i++) {
//            a[i] = sc.nextInt();
//        }
```

```
//        for (int i = 0; i < n; i++) {
//            if (n% a[i] == 0) {
//                System.out.print(a[i] + " "); //factors of the given number
//            }
//        }
//    }
//}
//        int a[] = new int[n];
//        for (int i = 0; i < n; i++) {
//            a[i] = sc.nextInt();
//        }
//        for (int i = 0; i < n; i++) {
//            int sum = 0;
//            while (a[i] != 0) {
//                int r = a[i] % 10;
//                sum += r;
//                a[i] = a[i] / 10;
//            }
//            System.out.print(sum+" ");//sum of the elements in the individual element in the
array
//        }
//    }
//}
//        int a[] = new int[n];
//        for (int i = 0; i < n; i++) {
//            a[i] = sc.nextInt();
//        }
//        Arrays.sort(a); //sort the array then it is very easy to find the maximum element in
the array
//        System.out.println(Arrays.toString(a));
//        int maxelement = a[n - 1]; // maximum element in the array
//        System.out.println(maxelement);
//    }
//}
//        int a[] = new int[n];
//        for (int i = 0; i < n; i++) {
//            a[i] = sc.nextInt();
//        }
//        Arrays.sort(a);
//        System.out.println(Arrays.toString(a));
//        int maxelement = a[n-2]; // second maximum element in the array program
//        System.out.println(maxelement);
//    }
//}
//        int a[] = new int[n];
//        for (int i = 0; i < n; i++) {
```

```
//            a[i] = sc.nextInt();
//        }
//        Arrays.sort(a);
//        System.out.println(Arrays.toString(a));
//        int i=0;
//        if (i == 0) {
//            int minelement = a[i]; //smallest elemnent in the array
//            System.out.println(minelement);
//        }
//    }
//}
// maximum element in the array without using any inbuilt functions
        int a[] = new int[n];
        for (int i = 0; i < n; i++) {
            a[i] = sc.nextInt();
        }
        int maxElement = a[0];
        for (int i = 0; i < n; i++) {
            if (a[i] > maxElement) {
                maxElement = a[i];
            }
        }
        System.out.println(maxElement);
    }
}
```

9) STRINGS:

```
import java.sql.SQLOutput;
import java.util.*;
public class strings {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
 /* Strings-collection of the characters(non primitive data type)=immutable(changes is not
possible)
    string index is starting from 0 only like arrays to n-1
    Strings Intialising methods:
 -->Literal method //It will create the memory space for variables/strings in the string
constant pool
 -->New keyword method //new method will create the memory space for the
variables/strings
  ** First compiler will check the string constant pool if any string is there it will not
create the memory space
    for others directly it will return the strings.(always scp=string constant pool is there in
objects(heap)
  ** frames is also called as the stack in the memory
    String s="12.345f" //here we gave,even the float value inside the double codes it will
consider it as
```

```
   the string only  like 1,2,.,3,4,5,f as characters.
   */
//  Compile input for the strings
//        String s = "VIRAT KOHLI";
//        System.out.println(s);
//    }
//}
//  Run time input for the strings
//        int cgpa = sc.nextInt();
//        String name = sc.next();
//        System.out.println(name+" is eligible for placements with "+cgpa+ " cgpa ");
//    }
//}
//    9
//    is eligible for placements with 9 cgpa  //output for the above statement
/*  Here there is the problem with nextLine()=when we are giving the nextInt() or
anyother datatype
before the string it will read integer and skip the nextLine() method.To avoid that problem
we have
 to create the dummy nextLine to enter the next input.
   */ /* we can use the next() and nextLine() for string but next will read only the first
words in
       sentence but nextLine() will read the entire sentence */
//        int cgpa = sc.nextInt();
//        sc.nextLine(); //Dummy variable here to overcome the problem of skipping the line
//        String name = sc.nextLine();
//        System.out.println(name + " is eligible for placements with " + cgpa + " cgpa ");
//    }
//}
//  ** To make the strings mutable use the string builder and string buffer **
//        String a = "virat";
//        StringBuilder b = new StringBuilder(a);
//        StringBuffer c = new StringBuffer("hello");//we cannot assign the builder value to
the buffer value
//        System.out.print(a+" "+b+" "+c);
//    }
//}
/*Inbuilt functions for the string bulider and the string buffer-not applicable for the strings
-->reverse()=Used to reverse the string
-->append()=used to insert the data at the end of the string
-->delete(start index,end index) where value is the elemnent that you wish to delete
-->insert(index,value) where value is the elemnent that you wish to insert
 */
//        String a = sc.nextLine();
//        StringBuilder a1 = new StringBuilder(a);
//        StringBuffer a2 = new StringBuffer("virat");
```

```
//        System.out.println(a2.reverse()); //tariv
//        System.out.println(a1.append(18));//virat kohli18
//        System.out.println(a1.delete(0,3));//delete only allows the integers like index position,here 0 to 3 means o to 2
//        System.out.println(a1.insert(3,"k"));
//    }
//}
 /* Differences between string builder and string buffer(**imp interview question)
 -->STRING BUFFER
  1)Synchronized  2)Thread Safe(we can access the one thread at a time) 3)Slower operation
 -->STRING BUILDER
  1)Non-Synchronized
  2)Non-thread safe(multithreading-we can access the mutliple threads simuntaneously)
  3)Fast operation/execution
  */
//  String Methods=Lot of string methods are there but here i am using the some of important methods.
//        String[] a = {"virat", "kohli", "chiku"}; //o/p: virat?kohli?chiku
//        System.out.println(String.join("-", a));//for join we have to create the string like above one
//        String a=sc.nextLine();
//        System.out.println(a.length());//here in strings length having this open and close brackets()
//        System.out.println(a.trim());//it will remove the white spaces at the starting or ending of the string
//        System.out.println(a.charAt(2));//it will fetch the element at the index position
//        System.out.println(a);
//        String a1=sc.nextLine();
//        System.out.println(a.compareTo(a1));
        //if both strings are equal it will return the integer value as 0
        // if the calling string is lexicographically(alphabetical order) less than the another string=Negative value
        // if the calling string is lexicographically(alphabetical order) greater than the another string=positive value
//        System.out.println(a.compareToIgnoreCase("k"));
        //Negative Integer: If the calling string is lexicographically less than the specified string.
        //Zero: If the calling string is lexicographically equal to the specified string.
        //Positive Integer: If the calling string is lexicographically greater than the specified string,
//        String a = sc.nextLine();
//        String a1=sc.nextLine();
//        System.out.println(a.equals(a1));//it will return the boolean values
//        string+string=here '+' is the concadenation and int+int=here '+' is the addition
//        System.out.println(a+a1); //both ways we can add the 2 strings
```

```
//        System.out.println(a.concat(a1));//concadenation
//        System.out.println(a.isEmpty());//it also returns the boolean value if string is empty
true,otherwise false
//        System.out.println(a.replace("i","s"));//replace(char oldchar,char newchar)
//        System.out.println(a.replaceAll("k","5"));
//        System.out.println(a.split(" ",2));//mainly used to give the spaces between the
strings
        //whenever we use split method we cannot understand the output
clearly=[Ljava.lang.String;@6d6f6e28 like this
//        System.out.println(a.substring(1, 4)); //1 to 3 elements because index is from 0 to
n-1
//        System.out.println(a.toCharArray());//convert strings into character arrays
//        System.out.println(a.toLowerCase()); //convert the string into lowercase letters
//        System.out.println(a.toUpperCase()); //converts the string into uppercase letters
//        System.out.println(a.indexOf("t"));//it will return the index value where the
element you want to search
//    }
//}
//  Reverse of a string program
//        String A = s.nextLine();
//        for (int i = A.length(); i > 0; i--) {
//            System.out.print(A.charAt(i - 1));
//        }
//    }
//}
//  count of characters
        String A = sc.nextLine();
        System.out.println(A.length());//number of characters in the string
    }
}
```

10) MATH FUNCTIONS:

```
import java.util.Scanner;
public class math{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
//        int b=sc.nextInt();
//        System.out.println(Math.max(a,b)); //Finds the maximum value among the 2 values
//        System.out.println(Math.min(a,b));//Finds the minimum value among the 2 values
//        //absolute values always give the result in the  positive values even if you give the
negative values
////        System.out.println(Math.abs(-2));
//        System.out.println(Math.ceil(a));//ceiling will give the above values eg:35.79=36.0
//        System.out.println(Math.floor(a)); //floor will give the values below value
eg:35.79=35.0
```

```
//        System.out.println(Math.round(a)); //It will remove the digits after the decimal
point and round up the values
//        System.out.println(Math.pow(a,b));
        //here it is giving in the decimal i want it in the integer so i am doing typecasting
here
//        System.out.println((int)Math.pow(a,b));
//        System.out.println(Math.sqrt(a));//find the square root of the number
//        System.out.println(Math.cbrt(a));//find the cube root of the number
//        System.out.println(Math.floorDiv(a,b));//it performs the division and dicards the
fractional part
        //division will give the result as 1 if we give 1/2(division rule) if a is smaller it give
that as o/p
//        System.out.println(Math.random()); //it will give the random values as output
//        System.out.println(Math.random()*100+1);//it gives the random values in between
1 to 100
          System.out.println(Math.log10(a));//it will gives the log values
//        System.out.println(Math.log10(45464357)+1);
    }
}
```

11) METHODS IN JAVA

```
import java.util.Scanner;
public class methods {
    /*Methods in java are defined as the block of the code in which it run when it is called.
    we can pass the parameters(data) into a method.Method performs the certain actions
known as the functions
    Syntax for the declaration of the method:
    returntype methodName(camelcase)(optional:parameters/formal parameters)
    void=it will not going to return anything at the end //return type
    main=it will going to return anything at the end //return type
    example:void isPrime(int n)
    System.out.println("prime);
   2.String isPrime(int n)//int n is optional
        return "yes" //when you gave the return type as string it will going to return the
string only
        return "no"
Types of the methods:
-->Methods without arguments/parameters without return
-->Methods without arguments with return
-->Methods with arguments without return
-->Methods with arguments with return
1) void isPrime() //no arguments,it will not return anything
2) String isPrime() //methods with return and without arguments
3) void isPrime(int n)
4) String isPrime(int n)
 */
// Prime number program using methods
```

```
//    static void isPrime(int n) {
//        if (n <= 1) {
//            System.out.println(n + " is not a prime number");
//        } else {
//            int flag = 0;
//            for (int i = 2; i < n; i++) {
//                if (n % i == 0) {
//                    flag = 1;
//                    break;
//                }
//            }
//            if (flag == 0) {
//                System.out.println(n + " is a prime number");
//            } else {
//                System.out.println(n + " is not a prime number");
//            }
//        }
//    }
//            public static void main (String[] args){
//               Scanner sc=new Scanner(System.in);
//               int n=sc.nextInt();
//                isPrime(n);
//    }
//}
////example:even or odd using the methods
//static void oddEven(int n) {
//    if (n % 2 == 0) {
//        System.out.println(n + " is a even number");
//    } else {
//        System.out.println(n + " is a odd number");
//    }
//}
//public static void main (String[] args){
//    Scanner sc = new Scanner(System.in);
//    int n = sc.nextInt();
//    oddEven(n);
//    }
//}
//Recursion=a function calls itself is called as recursion
//program for factorial of a nubmber using the recursion
//    static int fact(int n) {
//        if (n == 0) {
//            return 1;
//        } else {
//            return n * fact(n - 1);
//        }
```

```
//    }
//
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        int n = sc.nextInt();
//        int result=fact(n);
//        System.out.println(result);
//    }
//}
//    sum of n natural numbers using the recursion method without using the loops
    static void natural_numbers(int n) {
        if (n == 0) {
            return;
        }//it will give the result in the reverse order to make the output in normal ascending
order we hae to reverse the both statements
        natural_numbers(n-1);
        System.out.print(n+" ");
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        natural_numbers(n);
    }
}
```

12) PATTERN PROGRAMS IN JAVA

```
import java.util.*;
public class patterns {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        pattern1(n);
    }

    ////    Right half pyramid-1
//    static void pattern1(int n){
//        for(int row=1;row<=n;row++){
//            for(int col=1;col<=row;col++) {
//                System.out.print("* ");
//            }
//            System.out.println();
//        }
//    }
//}
////    square fill pattern-2
//    static void pattern2(int n) {
//        for (int row = 1; row <= n; row++) {
```

```
//            for (int col = 1; col <= n; col++) {
//                System.out.print("*");
//            }
//            System.out.println();
//        }
//    }
//}
//Reverse right half pyramid-3
//static void pattern1(int n){
//        for(int row=1;row<=n;row++){
//            for(int col=1;col<=n-row+1;col++) {
//                System.out.print("* ");
//            }
//            System.out.println();
//        }
//    }
//}
//    number increaing pyramid-4
//static void pattern1(int n){
//        for(int row=1;row<=n;row++){
//            for(int col=1;col<=row;col++) {
//                System.out.print(col+" ");
//            }
//            System.out.println();
//        }
//    }
//}
//    number incresing reverse pyramid-5
//    static void pattern1(int n) {
//        for (int row = 1; row <= n; row++) {
//            for (int col = 1; col <= n - row + 1; col++) {
//                System.out.print(col + " ");
//            }
//            System.out.println();
//        }
//    }
//}
//   Half  Diamond pattern-6
//    static void pattern1(int n) {
//        for (int row = 1; row < 2 * n; row++) {
//            int c = (row > n ? 2 * n - row : row);
//            for (int col = 1; col <= c; col++) {
//                System.out.print("*");
//            }
//            System.out.println();
//        }
```

```
//    }
//}
//    Triangle star pattern-7
//static void pattern1(int n){
//        for(int row=1;row<=n;row++){ //outer loop for rows
//            for(int j=row;j<n;j++) {
//                System.out.print(" ");
//            }
//            for(int j=1;j<=(2*row-1);j++){
//                System.out.print("*");
//            }
//            System.out.println();
//        }
//    }
//}
//    Reverse left half pyramid-8
//    static void pattern1(int n) {
//        for (int row = 1; row <= n; row++) {
//            for (int col =1; col<row; col++) {
//                System.out.print(" ");
//            }
//            for (int col= 1; col <=n-row+1; col++) {
//                System.out.print("*");
//            }
//            System.out.println();
//        }
//    }
//}
//    Left half pyramid-9
//    static void pattern1(int n) {
//        for (int row = 1; row <= n; row++) {
//            for (int col =row; col<=n; col++) {
//                System.out.print(" ");
//            }
//            for (int col= 1; col <=row; col++) {
//                System.out.print("*");
//            }
//            System.out.println();
//        }
//    }
//}
//Triangle star pattern-10
//    static void pattern1(int n) {
//        for (int row = 1; row <= n; row++) {
//            for (int col =n; col>row; col--) {
//                System.out.print(" ");
```

```
//            }
//            for (int k = 1; k <= (2 * row - 1); k++) {
//                System.out.print("*");
//            }
//            System.out.println();
//        }
//    }
//}
//Reverse triangle star pattern-11
//    static void pattern1(int n) {
//        for (int row = n; row >=1; row--) {
//            for (int col =n; col>row; col--) {
//                System.out.print(" ");
//            }
//            for (int k = 1; k <=(2*row-1); k++) {
//                System.out.print("*");
//            }
//            System.out.println();
//        }
//    }
//}
//    Pyramid pattern-12
//    static void pattern1(int n) {
//        for (int row = 1; row <= n; row++) {
//            for (int col =n; col>row; col--) {
//                System.out.print(" ");
//            }
//            for (int k = 1; k <=row; k++) {
//                System.out.print("* ");
//            }
//            System.out.println();
//        }
//    }
//}
//    Reverse pyramid pattern-13
//static void pattern1(int n) {
//    for (int row = 1; row <= n; row++) {
//        for (int col =1; col<row; col++) {
//            System.out.print(" ");
//        }
//        for (int k = 1; k <=n-row+1; k++) {
//            System.out.print("* ");
//        }
//        System.out.println();
//    }
//}
```

```
//}
//    Number triangular pattern-14
//    static void pattern1(int n) {
//        for (int row = 1; row <= n; row++) {
//            for (int col =n; col>row; col--) {
//                System.out.print(" ");
//            }
//            for (int k = 1; k <=row; k++) {
//                System.out.print(row+" ");
//            }
//            System.out.println();
//        }
//    }
//}
// Number changing pyramid-15
//static void pattern1(int n){
//        int nums=1;
//        for(int row=1;row<=n;row++){
//            for(int col=1;col<=row;col++){
//                System.out.print(nums+" ");
//                nums++;
//            }
//            System.out.println();
//        }
//    }
//}
// Reverse triangle pattern with rows numbers-16
//    static void pattern1(int n) {
//        for (int row = 1; row <= n; row++) {
//            for (int col = 1; col <=n - row + 1; col++) {
//                System.out.print(row+" ");
//            }
//            System.out.println();
//        }
//    }
//}
//  zero Triangle pattern-17
//    static void pattern1(int n) {
//        for (int row = 0; row < n; row++) {
//            for (int col = 0; col <=row; col++) {
//                System.out.print((n-row+col)%2+" ");
//            }
//            System.out.println();
//        }
//    }
//}
```

```
// Reverse number triangular pattern-18
//    static void pattern1(int n) {
//        for (int row = 0; row <n; row++) {
//            for (int col =0; col<row; col++) {
//                System.out.print(" ");
//            }
//            for (int k = 1; k <n-row+1; k++) {
//                System.out.print(k+row+" ");
//            }
//            System.out.println();
//        }
//    }
//}
//    half diamond pattern with numbers as elements-19
//    static void pattern1(int n) {
//        for (int row = 1; row <= 2*n; row++) {
//            int val = (row>n? (2 * n - row+1) : row);
//            for (int col = 1; col <= val; col++) {
//                System.out.print(col + " ,");
//            }
//            System.out.println();
//        }
//    }
//}
// half diamond from left side-20
//   static void pattern1(int n){
//        for(int row=1;row<=2*n-1;row++){
//            int val=(row>n?(2*n-row):row);
//            for(int col=1;col<=n-val;col++){
//                System.out.print(" ");
//            }
//            for(int col=1;col<=val;col++) {
//                System.out.print("*");
//            }
//            System.out.println();
//        }
//   }
//}
//    Alphabet square pattern-21
//    static void pattern1(int n) {
//        char value = 'A';
//        for (int row = 1; row <= n; row++) {
//            for (int col = 1; col <= n; col++) {
//                System.out.print(value+" ");
//                value++;
//            }
```

```
//            System.out.println();
//        }
//    }
//}
//alphabet left right trinagle pattern-21
//    static void pattern1(int n) {
//        char value = 'A';
//        for (int row = 1; row <= n; row++) {
//            for (int col = 1; col <= row; col++) {
//                System.out.print(value+" ");
//                value++;
//            }
//            System.out.println();
//        }
//    }
//}
// Alphabet Right triangle pattern-22
//    static void pattern1(int n) {
//        char value = 'A';
//        for (int row = 1; row <= n; row++) {
//            for (int col = 1; col <= n-row+1; col++) {
//                System.out.print(value+" ");
//                value++;
//            }
//            System.out.println();
//        }
//    }
//}
//    A
//   BC
//  DEF
// GHIJ
//KLMNO           -23
//    static void pattern1(int n) {
//        char value = 'A';
//        for (int row = 1; row <= n; row++) {
//            for (int col = row; col <n; col++){
//                System.out.print(" ");
//        }
//            for (int col = 1; col <=row; col++) {
//                System.out.print(value );
//                value++;
//            }
//            System.out.println();
//        }
//    }
```

```
//}
//ABCDE
//  FGHI
//   JKL
//    MN
//     O                -24
//    static void pattern1(int n) {
//        char value = 'A';
//        for (int row = 1; row <= n; row++) {
//            for (int col = 1; col <=row; col++){
//                System.out.print(" ");
//        }
//            for (int col = 1; col <=n-row+1; col++) {
//                System.out.print(value);
//                value++;
//            }
//            System.out.println();
//        }
//    }
//}
//     A
//    B C
//   D E F
//  G H I J
// K L M N O              -25
//    static void pattern1(int n) {
//        char value = 'A';
//        for (int row = 1; row <= n; row++) {
//            for (int col = row; col <=n; col++){
//                System.out.print(" ");
//        }
//            for (int col = 1; col <=row; col++) {
//                System.out.print(value+" ");
//                value++;
//            }
//            System.out.println();
//        }
//    }
//}
//    A B C D E
//     F G H I
//     J K L
//      M N
//       O                       -26
//    static void pattern1(int n) {
//        char value = 'A';
```

```
//        for (int row = 1; row <= n; row++) {
//            for (int col = 1; col <=row; col++){
//                System.out.print(" ");
//            }
//            for (int col = 1; col <=n-row+1; col++) {
//                System.out.print(value+" ");
//                value++;
//            }
//            System.out.println();
//        }
//    }
//}
//program-27
//   static void pattern1(int n){
//    for(int row=1;row<=2*n;row++){
//       int val=(row>n?row-n:n-row+1);
//       for(int col=1;col <=n-val;col++) {
//          System.out.print(" ");
//       }
//       for(int col=1;col<val;col++){
//          System.out.print("* ");
//       }
//        System.out.println();
//       }
//   }
//}
// * * * * *
//  * * * *
//   * * *
//    * *
//    *
//    *
//   * *
//  * * *
// * * * *        -28
//* * * * *
//    static void pattern1(int n) {
//        for (int row = 1; row <= n; row++) {
//            for (int col = 1; col < row; col++) {
//                System.out.print(" ");
//            }
//            for (int k = 1; k <= n - row + 1; k++) {
//                System.out.print("* ");
//            }
//            System.out.println();
//        }
```

```
//        for(int row=1;row<=n;row++){
//            for(int j=row;j<n;j++) {
//                System.out.print(" ");
//            }
//            for(int j=1;j<=row;j++){
//                System.out.print("* ");
//            }
//            System.out.println();
//        }
//    }
//}
// Diamond pattern-29
//              *
//             * *
//            * * *
//           * * * *
//          * * * * *
//           * * * *
//            * * *
//             * *
//              *

    //    static void pattern1(int n) {
//        for (int row = 1; row <= 2 * n - 1; row++) {
//            int val = (row > n ? 2 * n - row : row);
//            for (int col = 1; col <=n-val; col++) {
//                System.out.print(" ");
//            }
//            for (int col = 1; col <=val; col++) {
//                System.out.print("* ");
//            }
//            System.out.println();
//        }
//    }
//}
//            1
//          2 1 2
//        3 2 1 2 3
//      4 3 2 1 2 3 4
//    5 4 3 2 1 2 3 4 5      -30
//    static void pattern1(int n) {
//        for (int row = 1; row <= n; row++) {
//            for (int col = 1; col <=n-row ; col++) {
//                System.out.print("  ");
//            }
//            for (int col = row; col >= 1; col--) {
```

```
//             System.out.print(col + " ");
//         }
//         for (int col = 2; col <= row; col++) {
//             System.out.print(col + " ");
//         }
//         System.out.println();
//     }
// }
//}
//           1
//         2 1 2
//       3 2 1 2 3
//     4 3 2 1 2 3 4
//   5 4 3 2 1 2 3 4 5
//     4 3 2 1 2 3 4
//       3 2 1 2 3
//         2 1 2            -31
//           1
//   static void pattern1(int n) {
//     for (int row = 1; row <=2*n; row++) {
//         int val=(row>n?2*n-row:row);
//         for (int col = 1; col <=n-val ; col++) {
//             System.out.print("  ");
//         }
//         for (int col = val; col >= 1; col--) {
//             System.out.print(col + " ");
//         }
//         for (int col = 2; col <=val; col++) {
//             System.out.print(col + " ");
//         }
//         System.out.println();
//     }
// }
//}
//         4 4 4 4 4 4 4 4 4
//         4 3 3 3 3 3 3 3 4
//         4 3 2 2 2 2 2 3 4
//         4 3 2 1 1 1 2 3 4
//         4 3 2 1 0 1 2 3 4
//         4 3 2 1 1 1 2 3 4
//         4 3 2 2 2 2 2 3 4
//         4 3 3 3 3 3 3 3 4    -32
//         4 4 4 4 4 4 4 4 4
//   static void pattern1(int n) {
//     int originalN=n;
//     n=n*2;
```

```
//        for (int row = 0; row <=n; row++) {
//            for (int col = 0; col <=n; col++) {
//                int val =originalN-Math.min(Math.min(row, col), Math.min(n - row, n - col));
//                System.out.print(val + " ");
//            }
//            System.out.println();
//        }
//    }
//}
//            *
//          *   *
//         *     *
//        *******          -33
//    static void pattern1(int n) {
//        for (int row = 1; row <=n; row++) {
//            for (int col = 1; col <= 7; col++) {
//                if ((row == 4)|| (row + col == 5 )|| (col - row == 3)) {
//                    System.out.print("*");
//                } else {
//                    System.out.print(" ");
//                }
//            }
//            System.out.println();
//        }
//    }
//}
////         *******
//           *     *
//            *   *
//              *                -34
//    static void pattern1(int n) {
//        for (int row = 1; row <n; row++) {
//            for (int col = 1; col <= 7; col++) {
//                if ((row == 1)|| (row+col==8)||(row-col==0)){
//                    System.out.print("*");
//                }
//                else {
//                    System.out.print(" ");
//                }
//            }
//            System.out.println();
//        }
//    }
//}
//
```

```
//           *
//          * *
//         *   *
//         *    *
//        *      *
//         *    *
//          *  *          -35
//           * *
//            *
//    static void pattern1(int n) {
//        for (int row = 1; row <2*n; row++) {
//            for (int col = 1; col <= 9; col++) {
//                if ((row + col == 6) || (col - row == 4)||(row + col == 14) || (row - col == 4)) {
//                    System.out.print("*");
//                } else {
//                    System.out.print(" ");
//                }
//            }
//            System.out.println();
//        }
//    }
//}
//          ****
//          *   *
//          *   *
//          *   *
//          ****                -36
//    static void pattern1(int n) {
//        for (int row = 1; row <=n; row++) {
//            for (int col = 1; col <=n-1; col++) {
//                if ((row == 1)||(row==5)||(col==4)||(col==1)){
//                    System.out.print("*");
//                }
//                else {
//                    System.out.print(" ");
//                }
//            }
//            System.out.println();
//        }
//    }
//}
//         *****
//        *    *
//       *    *
//      *    *
//     *****   -37
```

```
//    static void pattern1(int n) {
//        for (int row = 1; row <=n; row++) {
//            for(int col=1;col<=n-row;col++){
//                System.out.print(" ");
//            }
//            for (int col = 1; col <=n; col++) {
//                if (row == 1 || row == n || col == 1 || col== n){
//                    System.out.print("*");
//                }
//                else {
//                    System.out.print(" ");
//                }
//            }
//            System.out.println();
//        }
//    }
//}
//    E
//    DE
//    CDE
//    BCDE
//    ABCDE           -38
//    static void pattern1(int n) {
//        for (int row =1; row <= n; row++) {
//            for (int col = 1; col <= row; col++) {
//                if (row == col) {
//                    System.out.print("E");
//                } else if (row - col == 1) {
//                    System.out.print("D");
//                } else if (row - col == 2) {
//                    System.out.print("C");
//                } else if (row - col == 3) {
//                    System.out.print("B");
//                } else {
//                    System.out.print("A");
//                }
//            }
//            System.out.println();
//        }
//    }
//}
//  program-39
//    static void pattern1(int n) {
//        for (int row =1; row <= n; row++) {
//            for (int col = 1; col <= n-row+1; col++) {
//                if (row+col==6) {
```

```
//                 System.out.print("A");
//             } else if (row + col == 5) {
//                 System.out.print("B");
//             } else if (row + col == 4) {
//                 System.out.print("C");
//             } else if (row +col == 3) {
//                 System.out.print("D");
//             } else {
//                 System.out.print("E");
//             }
//         }
//         System.out.println();
//     }
// }
//}
//     1       1
//     12      21
//     123     321
//     1234  4321
//     1234554321          =40
//  static void pattern1(int n) {
//     for (int row = 1; row <= n; row++) {
//         for (int col = 1; col <=row; col++) {
//             System.out.print(col);
//         }
//         for (int col = 1; col <=2*(n-row); col++) {
//             System.out.print(" ");
//         }
//         for(int col=row;col>=1;col--){
//             System.out.print(col);
//         }
//         System.out.println();
//     }
// }
//}
```

13) OOPS CONCEPT:

```
/*Object-oriented programming language-oops
There are mainly 6 types are there in the oops:
-->Class
-->Object
-->Abstraction
-->Polymorphism
-->Encapusulation
-->Inheritance
 */
```

```
/*Constructor:Constructor is the special method in class mainly used to intialize the
object
->Always in constructor the method name should be same as class name
eg:class Mobile{ //class name is also Mobile
public static void Mobile(){ //method name is Mobile
3 types of constructors:
-->Default constructor or Non-parametrized constructor
-->Parameterized constructor
-->Copy constructor
-->Default constructor example:
public static void Mobile()=here you are not passing any value inside the open and close
brackets so it is default
if you are not giving any constructor the compiler itself create the default constructor.
-->Parameterized constructor Example:
public static void Mobile(int a,int b)=here we are passing some values that is
parameterized constructor
if you are creating the parameterized constructor then compiler won't create the default
constructor
-->Copy constructor=creates an object using another object of the same Java class.
class Mobile[
public static void Mobile(String name,int price){
public static void Mobile(Mobile color){ //
        this.name=color.name; //this is the keyword
        this.price=color.price;
 */
import java.util.*;
public class oops {  //object creation and accessing the behaviours and properties from
class
   public static void main(String[] args) {
      Mobile realme=new Mobile(); //creation of object
      realme.music();
      System.out.println(realme.frontcamera);
   }
}
/*ClASS=It is blueprint/structure which contains properties and behaviours(fields and
methods)
OBJECT=Instance of class-to access the properties and behaviours of the class or
instance of a class is object
 */
class Mobile{  //class creation example
   int frontcamera,rearcamera,volumeup,volumedown,powerbutton;//properties=here we
are not giving any values so defaulty it take 0 as the answer
   void call() {
      System.out.println("u can make a call");
   }
   void music(){  //music and call are methods
```

```
        System.out.println("spotify");
    }
}
```
/*Abstraction=The process of hiding the unnecessary information and exposing the necessary data to the users
Real world examples are like car and ATM Machine etc.
Abstract class=The class that is declared as Abstract is called as Abstract class.It should be instantiated.it can have
both the abstract and non abstract methods.we must have to use only abstract.
There are two ways to achieve abstraction in java:
1) Abstract class (0 to 100%)
2) Interface (100%)
example:abstract class classname{ //Abstract class
2) abstract void run(); //Abstract Method
 */
/*Encapusulation:The process of combining the data and methods in a single unit is called encapusulation
example:Capsule is best example for encapsulation because all the different medicines are wrapped inside the capsule
java class is also an example of the encapsulation because all the methods/behaviour and fields are wrapped inside the class
public class encapsulation{
**Difference between encapuslation and abstraction

Encapsulation	Abstraction
1)Combining the data and methods in a sinle unit called "encapsulation"	1)Hiding the unneccessary information and exposing the relevant information to the users
2)Encapsulation is implemented by using private,protected,private pacakged access modifiers.	2) It can be implemented using interface and abstract class.
3)It soves the problem in the implementation level	3)It solves the problem in the design level

 */
/*Inheritance:The process of deriving the properties from one class to another class
Inheritance mainly has the two classes:
1) Parent class or super class or base class
2) Child class or sub class or derived class
Example:All the properties of father is inherited by his son.
The keyword used for inheritance is the "extends"
Types of inheritance in java:
1)Single inheritance
2)Mutlilevel inheritance
3)Hirerachical inheritance
4)Multiple inheritance
5)Hybrid Inheritance
-->Single inheritance:
 Class A---->extends----->class B
```
```

class B(child class) will inherit the features from class A(Parent class)
-->Multilevel inheritance
class A---->extends---->class B--->extends--->class C
here class C(child class) will inherit the features from class B(Parent class) and Class B(Child class) will inherit
features from the class A(Parent class).
--->Hierarchical inheritance

```
        class A
       /       \
  class B     class C
```

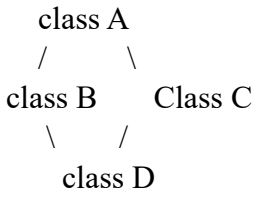
Class B and class C(Child class) will inherit the features from class A(Parent class)
In hierarchical inheritance one parent class and more than one child class will br there
--->Mutliple inheritance

```
  Class A   Class B
   \        /
     class C
```

Here class c(child class)will inherit the features from Class A and class B(Parent class)
In multiple inheritance one child class and more than the one parent class will be there.
** Java does not support the multiple inheritance **
we can only achieve the mutliple inheritance by using the interface concept.
Why java does not support multiple inheritance?
Diamond problem definition:
If both the parent classes have the methods with same name and parameters,then child class will get confused about
which method to call.
example:

```
import java.io.*;
class Parent1 {
    void fun() { //same method for parent class
        System.out.println("Parent1");
    }
}
class Parent2 {
    void fun() { //same method for the parent class
        System.out.println("Parent2");
    }
}
class Test extends Parent1, Parent2 { // Error: Multiple inheritance not allowed
    public static void main(String[] args) {
        Test t = new Test();
        t.fun();
    }
}
```

--->Hybrid Inheritance(multiple+hierarcial)



It is mix of two or more types of inheritance.since java does not support the multiple inheritance and hybrid inheritance
is also not supported by java.It can also achieve using interface concept.
*/
/*polymorphism:
Poly means many and morphism forms
The process of representing the one form into mutliple forms
Polymorphism perform a single action in different ways.
Real time example:Suppose if we were in classroom we are behaving like student,at home like son or daughter and at market
like customer and in bus we are like passenger.
Types of polymorphism in java:
1) Compile time polymorphism or Method overloading or static binding
2) Run time Polymorphism or Method overriding or dynamic binding
1)Method overloading: If the class contains two or more methods having the same name and different arguments
Simply,Changing no.of arguments is called method overloading
2)method overriding:Method must have same parameter as in the parent class.
If child class has same method as declared in the parent class.
Upcasting:When reference variable of parent class refers to the object of child class
*/

14) KEYWORDS IN JAVA

1)this-Used to refer/specify the current class.(if property name and constructor variables same then use this keyword.
2)super-calls immediate parent class constructor
3)final-To make a property as constant.
4)static-Instead of creating the object we can make the variable as the static one.

15) ACCESS MODIFIERS

There are 4 types of access modifiers are there:

1) Private-The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
   Syntax: private void msg(){
2) Default-The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
   Syntax: void msg(){ //it will consider as as default because we didn't mention default
3) Public-The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.
   Syntax: public class A{
   Public void msg(){

4) protected-The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
   Synatx: protected void msg(){ \\ here msg is the method name

16) Java related some Questions

1) Jagged array: It is similar to the array but only difference is it can contain different row size and columns size.

Eg: If we are declaring the rows size as 3 means columns can have the size 2.

Syntax: int[][] JaggedArray=new int[2][];//here row size is 2.

```
// First row has 3 columns
  arr[0] = new int[3];
// Second row has 2 columns
  arr[1] = new int[2];
```

Advantages of jagged array:

1) Dynamic Allocation
2) Space utilization
3) Flexibility
4) Improved performance.

2) Is java 100% pure object oriented programming language?

A) The answer is NO,Because of primitive datatypes like int,char,float,double,long etc java is not 100% pure object oriented programming.If we want to make it as pure object oriented programming make use of wrapper class like Integer,Float,Boolean etc instead of the primitive datatypes.

Autoboxing: Refers to the conversion of a primitive value into an object of the corresponding wrapper class is called autoboxing. For eg: converting int to Integer class.

Primitive datatype→Wrapper class(Autoboxing)

Unboxing: Refers to the conversion of object of wrapper class to the primitive datatype is called unboxing.for example,Integer class is converted into int.

Wrapper class→Primitive datatype

17) MULTI THREADING

"Multithreading" is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

Threads can be created by using two mechanisms :

1. Extending the Thread class=We create a class that extends the "java.lang.Thread" class.
2. Implementing the Runnable Interface=We create a new class which implements java.lang.Runnable interface.

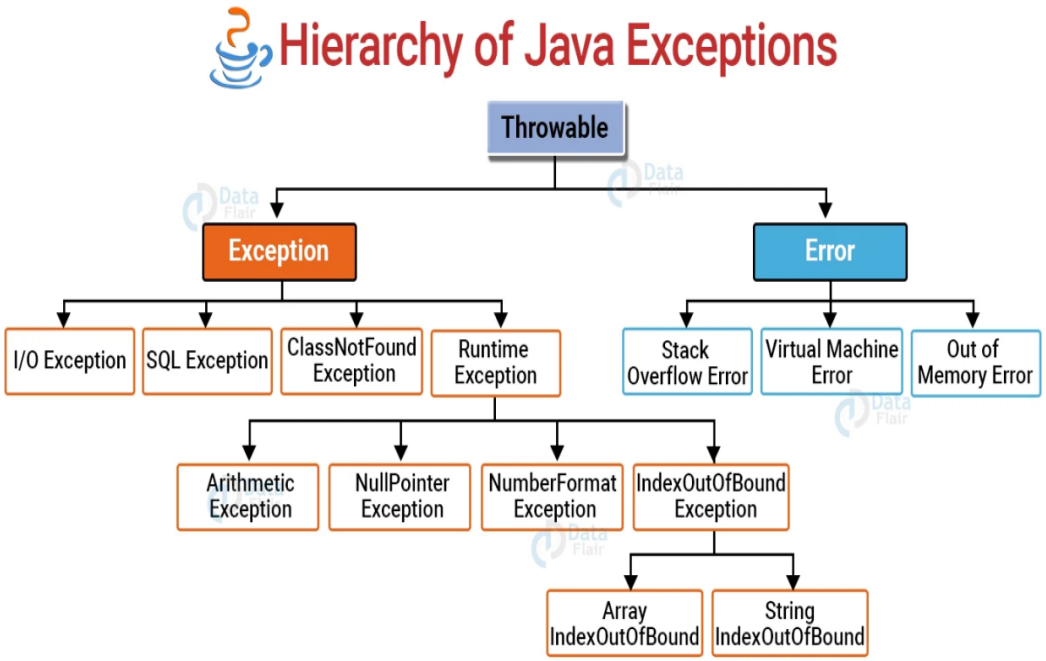
18) EXCEPTION HANDLING

Exception Handling: Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

➔ The core advantage of exception handling is **to maintain the normal flow of the application**.

➔ Exception is an abnormal condition.

➔ The **java.lang.Throwable** class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error.



Types of Java Exceptions:

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception.

1. **Checked Exception-** The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.
2. **Unchecked Exception-** The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.
3. **Error-** Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.

Java Exception Keywords: Java provides some keywords to handle exception

Syntax for try catch block:

```
try{

//code that may throw an exception

}catch(Exception_class_Name ref){}
```

Syntax for try finally block:

```
try{
//code that may throw an exception
}finally{}
```

Example for exception handling:

```
public class JavaExceptionExample{
  public static void main(String args[]){
   try{  //try keyword is used to place the exception code
      //code that may raise exception
      int data=100/0;
   }catch(ArithmeticException e){ //catch keyword is used to handle exception
    System.out.println(e);}
   //rest code of the program
   System.out.println("rest of the code...");
  }  }
```

Mutliple catch block:

```
public class MultipleCatchBlock1 {
  public static void main(String[] args) {
        try{
             int a[]=new int[5];
             a[5]=30/0;
            }
            catch(ArithmeticException e)
              {
               System.out.println("Arithmetic Exception occurs");
              }
```

```
            catch(ArrayIndexOutOfBoundsException e)
              {
               System.out.println("ArrayIndexOutOfBounds Exception occurs");
              }
            catch(Exception e)
              {
               System.out.println("Parent Exception occurs");
              }
            System.out.println("rest of the code");
   }  }
```

Nested try block syntax: A TRY block inside the try block is called nested try block

```
//main try block
try
{
   statement 1;
   statement 2;
//try catch block within another try block
   try
   {
      statement 3;
      statement 4;
//try catch block within nested try block
      try
      {
         statement 5;
```

```
            statement 6;
     }
        catch(Exception e2)
        {
//exception message
        }   }
    catch(Exception e1)
    {
//exception message
    }
}
//catch block of parent (outer) try block
catch(Exception e3)
{
//exception message
}
```

JAVA Finally block:

```
class TestFinallyBlock {
  public static void main(String args[]){
  try{
//below code do not throw any exception
   int data=25/5;
   System.out.println(data);
  }
//catch won't be executed
```

```
 catch(NullPointerException e){
System.out.println(e);
}
//executed regardless of exception occurred or not
 finally {
System.out.println("finally block is always executed");
}    System.out.println("rest of the code...");
  }
}
```

** The basic difference between final, finally and finalize is that the **final** is an access modifier, **finally** is the block in Exception Handling and **finalize** is the method of object class. **

Keyword	Description
try	The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.

1) A scenario where **ArithmeticException** occurs

If we divide any number by zero, there occurs an ArithmeticException.

Example: int a=50/0;//ArithmeticException

2) A scenario where **NullPointerException** occurs

If we have a null value in any variable, performing any operation on the variable throws a NullPointerException.

Example: String s=null;

System.out.println(s.length()); //NullPointerException

3) A scenario where **NumberFormatException** occurs

If the formatting of any variable or number is mismatched, it may result into NumberFormatException. Suppose we have a string variable that has characters; converting this variable into digit will cause NumberFormatException.

Example: String s="abc";

int i=Integer.parseInt(s);//NumberFormatException

//parseInt mainly used to convert string into the integer

4) A scenario where **ArrayIndexOutOfBoundsException** occurs

When an array exceeds to it's size, the ArrayIndexOutOfBoundsException occurs. there may be other reasons to occur ArrayIndexOutOfBoundsException. Consider the following statements.

int a[]=new int[5]; //here size is 5 but if it is greater than 5 it will throw arrayindexoutofbounds

a[10]=50; //ArrayIndexOutOfBoundsException

19) PACKAGES IN JAVA:

A java package is a group of similar types of classes, interfaces and sub-packages.

→Package in java can be categorized in 2 types , **built-in package**(packages from the Java API) and **user-defined package**(create your own packages)

→There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Advantages of Java Package:

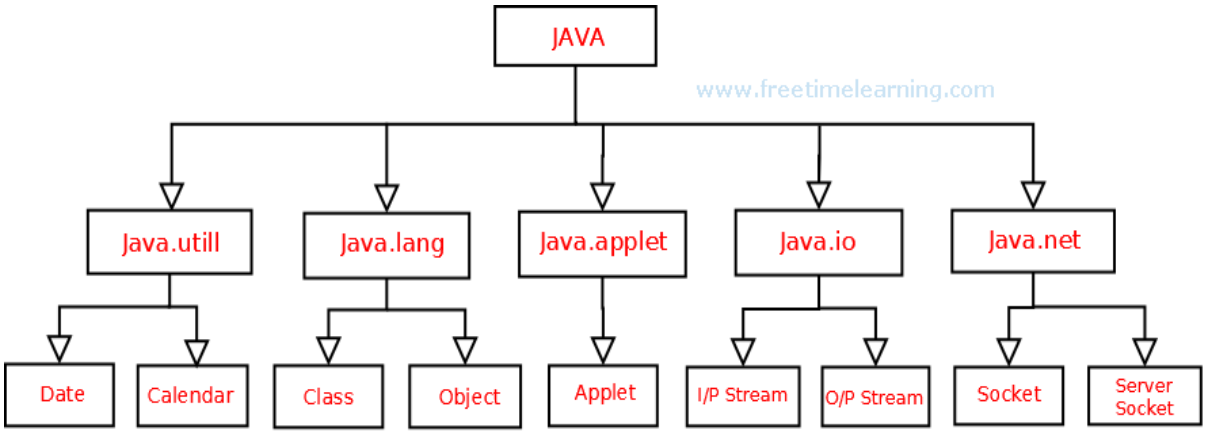
1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision**.**

Why do we need packages:

- ➔ To avoid name conflicts, and to write a better maintainable code.
- ➔ Mainly packages are used for the developers to store and organize their source code.



- ✓ **How to run the package program in java**

To Compile: javac -d . Simple.java

To Run: java mypack.Simple

To compile the java program use: javac filename.java and the extension is .java