

## **1. Write a Program for printing Hello World in java**

### **Program:**

```
package sample;

public class Simple {

    public static void main(String[] args)

    {

        System.out.println("Welcome to  Java");

        int a=10;

        int b=10;

        int c=a+b;

        System.out.println("The Sum of A and B is: "+c);

    }

}
```

## **2. Write a Program for Variables and Data Types in Java**

### **Program:**

```
package sample;

public class Variables {

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

```
String name = "Aiswarya";
```

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

```
//Number
```

```
int myNum;
```

```
myNum = 03;
```

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

```
//UpdateNumber
```

```
int myNums = 03;
```

```
myNums = 10; // myNum is now 20
```

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

```
//Character
```

```
char myLetter = 'A';
```

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

```
//Float
```

```
float myFloatNum = 5.99f;
```

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

```
//Boolean
```

```
boolean myBool = true;
```

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

```
//Integer Datatypes
```

```
//byte
```

```
byte mynum = 100;
```

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

```
//Double
```

```
double myNumr = 19.99d;
```

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

```
//Short
```

```
short mynums = 5000;
```

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

```
//Long
```

```
long myNume = 15000000000L;
```

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

```
//Scientific Numbers
```

```
float f1 = 35e3f;
```

```
double d1 = 12E4d;
```

```
System.out.println(f1);
```

```
System.out.println(d1);
```

```
//Boolean
```

```
boolean Javaisprogramminglanguage = true;
```

```
boolean isjavaaprocedurallanguage= false;
```

```
System.out.println(Javaisprogramminglanguage);    // Outputs  
true
```

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

```
//Character
```

```
char myVar1 = 65, myVar2 = 66, myVar3 = 67;
```

```
System.out.println(myVar1);
```

```
System.out.println(myVar2);
```

```
System.out.println(myVar3);
```

```
//Printing Variables
```

```
//String
```

```
String names = "Aiswarya";
```

```
System.out.println("Good Morning " + names);
```

```
//Integer
```

```
int x = 5;
```

```
int y = 5;
```

```
System.out.println(x + y); // Print the value of x + y
```

```
//One Value to Multiple Variables
```

```
int a, b, c;
```

```
a = b = c = 50;
```

```
System.out.println(a + b + c);
```

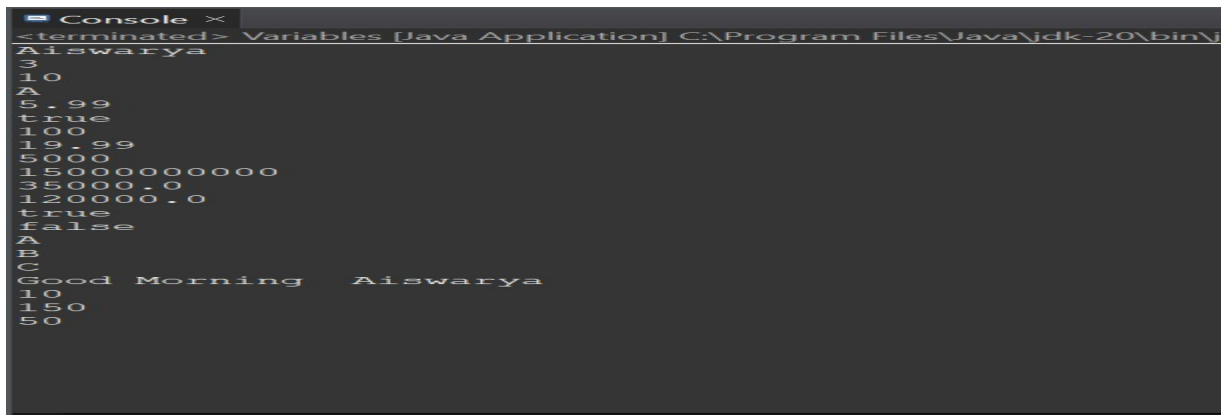
```
//Declare Many Variables
```

```
int p = 5, q = 5, r = 40;
```

```
System.out.println(p + q + r);
```

```
}
```

```
}
```



```
Console x
<terminated> Variables [Java Application] C:\Program Files\Java\jdk-20\bin\j
Aiswarya
3
10
A
5.99
true
100
19.99
5000
1500000000000
35000.0
1200000.0
true
false
A
B
C
Good Morning Aiswarya
10
150
50
```

**Output:**

### 3.Operators In Java:

Java arithmetic operators are used to perform addition, subtraction, multiplication, and division. They act as basic mathematical operations.

**Program:**

```
package sample;

public class Operators {

    public static void main(String[] args) {

        //Arithmetic Operators

        //Addition

        int xk = 5;

        int yk = 5;

        System.out.println(xk + yk);


        //Subtraction

        int xl = 8;

        int yl = 6;

        System.out.println(xl - yl);


        //Multiplication

        int xm = 2;

        int ym = 8;

        System.out.println(xm * ym);

        //Division

        int xn = 10;

        int yn = 2;
```

```
System.out.println(xn / yn);
```

```
//Modulus
```

```
int xo = 3;
```

```
int yo = 2;
```

```
System.out.println(xo % yo);
```

```
//Increment
```

```
int xp = 9;
```

```
++xp;
```

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

```
//Post Increment
```

```
int xq = 8;
```

```
xp++;
```

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

```
//Decrement
```

```
int xr = 5;
```

```
--xr;
```

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

//Post Decrement

```
int xs = 7;
```

```
xs--;
```

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

**//Assignment Operator**

```
int x = 3;
```

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

//Addition Assignment Operator

```
int y = 10;
```

```
y += 5;
```

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

//Subtraction Assignment Operator

```
int z = 5;
```

```
z -= 3;
```

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

//Multiplication Assignment Operator



```
int p = 5;  
p *= 3;  
System.out.println(p);
```

//Division Assignment Operator

```
double q = 5;  
q /= 3;  
System.out.println(q);
```

//Modulus Assignment Operator

```
int r = 5;  
r %= 3;  
System.out.println(r);
```

//OR equal to

```
int s = 5;  
s |= 3;  
System.out.println(s);
```

//Greater Than Assignment Operator

```
int t = 5;
```

```
t >>= 3;
```

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

```
//Lesser Than Assignment Operator
```

```
int u = 5;
```

```
u <<= 3;
```

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

```
//Equality and Relational Operators(Comparison Operator)
```

```
//Equal to Operator
```

```
int xa = 5;
```

```
int ya = 3;
```

```
System.out.println(xa == ya);
```

```
//Equal to Operator
```

```
int xb = 5;
```

```
int yb = 5;
```

```
System.out.println(xb == yb);
```

```
//Not Equal to Operator
```

```
int xd = 6;
```

```
int yd = 3;
```

```
System.out.println(xd != yd); // returns true because 5 is not equal  
to 3
```

```
//Greater Than Operator
```

```
int xc = 5;
```

```
int yc = 3;
```

```
System.out.println(xc > yc); // returns true because 5 is greater  
than 3
```

```
//Lesser Than Operator
```

```
int xe = 8;
```

```
int ye = 2;
```

```
System.out.println(xe < ye); // returns false because 5 is not less  
than 3
```

```
//Greater Than or Equal To Operator
```

```
int xf = 10;
```

```
int yf = 6;
```

```
System.out.println(xf >= yf); // returns true because 5 is greater, or  
equal, to 3
```

//Lesser Than or Equal To Operator

```
int xg = 6;
```

```
int yg = 4;
```

```
System.out.println(xg <= yg); // returns false because 5 is neither  
less than or equal to 3
```

## **//Logical Operators**

//Logical and

```
int xh = 5;
```

```
System.out.println(xh > 3 && xh < 10); // returns true because 5 is  
greater than 3 AND 5 is less than 10
```

//Logical or

```
int xi = 5;
```

```
System.out.println(xi > 3 || xi < 4); // returns true because one of  
the conditions are true (5 is greater than 3, but 5 is not less than 4)
```

//Logical not

```
int xj = 5;
```

```
System.out.println(!(xj > 3 && xj < 10)); // returns false because !  
(not) is used to reverse the result
```

### **//Bitwise Operator**

```
int n1 = 3, n2 = 6, n3 =5;
```

#### **//Bitwise AND**

```
System.out.println(n1 & n2);
```

#### **//Bitwise OR**

```
System.out.println(n1 | n2);
```

#### **//Bitwise XOR**

```
System.out.println(n1 ^ n2);
```

#### **//Binary Complement Operator**

```
System.out.println( ~n1 );
```

#### **//Left Shift Operator**

```
int ab=4;
```

```
System.out.println(ab<<1);
```

```
int ac=16;
```

```
System.out.println(ac<<4);
```

```
//Binary Left Shift Operator
```

```
n3 = n1 << 2;
```

```
System.out.println(n3);
```

```
//Binary Right Shift Operator
```

```
n3 = n1 >> 3;
```

```
System.out.println(n3);
```

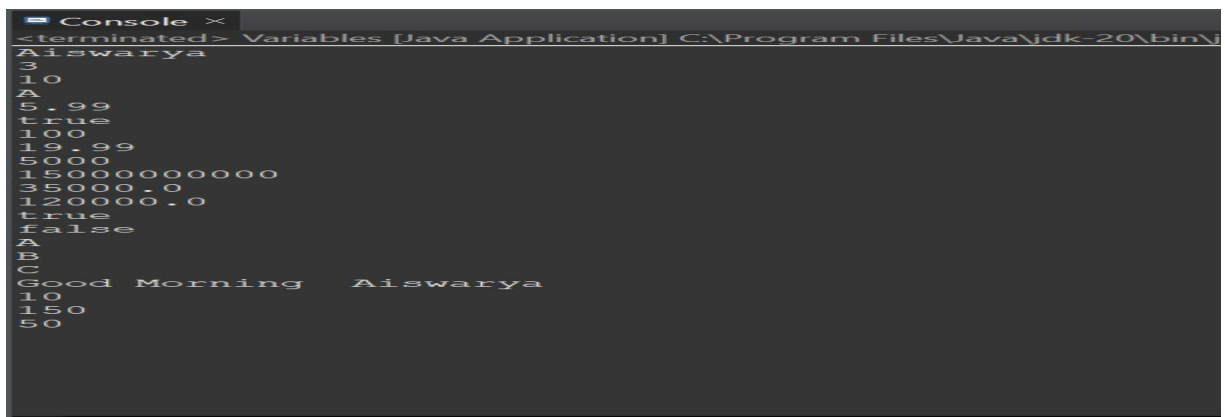
```
//Shift right zero fill operator
```

```
n3 = n1 >>> 4;
```

```
System.out.println(n3);
```

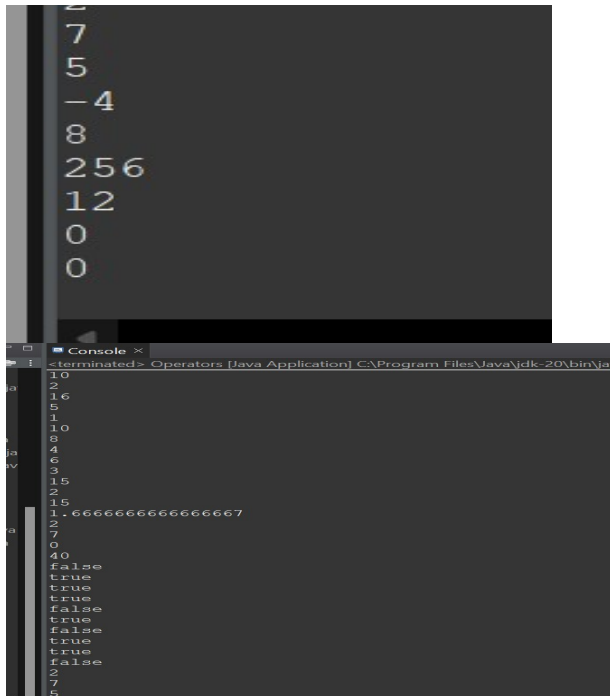
```
}
```

```
}
```



```
Console x
<terminated> Variables [Java Application] C:\Program Files\Java\jdk-20\bin\
Aiswarya
3
10
A
5.99
true
100
19.99
50000
1500000000000
35000.0
120000.0
true
false
A
B
C
Good Morning Aiswarya
10
150
50
```

**Output:**



The image shows a Java application window titled "Operators [Java Application] C:\Program Files\Java\jdk-20\bin\jav". The window displays the results of various arithmetic and logical operations. The arithmetic operations shown are: 10 + 2 = 12, 16 - 5 = 11, 10 \* 8 = 80, 4 / 6 = 0.6666666666666667, 15 % 2 = 1, and 7 < 0 = false. The logical operations shown are: true & true = true, true & false = false, false & true = false, true & true = true, false & false = false, true & false = false, false & true = false, and true & true = true.

## Decision Making Statements:

- \*Java compiler executes the code from top to bottom.
- \*The statements in the code are executed according to the order in which they appear. However, Java provides statements that can be used to control the flow of Java code. Such statements are called control flow statements.
- \*It is one of the fundamental features of Java, which provides a smooth flow of program.

## Java provides three types of control flow statements.

### 1. Decision Making statements

- \*if statements
- \*switch statement

## 2. Loop statements

- \*do while loop

- \*while loop

- \*for loop

- \*for-each loop

## 3. Jump statements

- \*break statement

- \*continue statement

## **1. Decision Making statements:**

- \*If Statement:**

**Program:**

```
package dcstatements;  
  
public class DecisionMaking  
{  
    public static void main(String[] args)  
    {  
        int age=18;
```



```
if (age>=18)
{
System.out.println("Eligible To Vote");

}

}
```

### **Else If Statement:**

#### **Program:**

```
package dcstatements;

public class DMElseif
{
public static void main(String[] args)
{
int age=7;
if (age>=18)
{
System.out.println("Eligible To Vote");
}
else
{
```

```
        System.out.println("Not Eligible To Vote");
    }
}
}
```

## **If ElseIf Statements:**

### **Program:**

```
package dcstatements;

public class DMIfElseIf
{
    public static void main(String[] args)
    {
        int num=8;
        if (num>0)
        {
            System.out.println("The Given Number is Positive.");
        }
        else if (num>=0) {
            System.out.println("The Given Number is Zero.");
        }
        else
        {

```

```
System.out.println("The Given Number is Negative");  
}  
}  
}
```

## **NestedIf Statement:**

### **Program:**

```
package dcstatements;  
  
public class DMNestedIf  
{  
    public static void main(String[] args)  
    {  
        int n1 = 2, n2 = 4, n3 = 8, largest;  
        if (n1 >= n2)  
        {  
            if (n1 >= n3)  
            {  
                largest = n1;  
                System.out.println("Number N1 is Greater"+n1);  
            }  
            else
```

```
{
largest = n3;
}
} else {
if (n2 >= n3) {
largest = n2;
System.out.println("Number N2 is Greater"+n2);
}
else {
largest = n3;
System.out.println("Number N3 is Greater"+n3);
}
}
}
```

## **Switch Statement:**

### **Program:**

```
package dcstatements;

import java.util.*;

public class DMSwitchstatement
{
```

```
public static void main(String[] args)
{
Scanner ab= new Scanner(System.in);
System.out.print("Enter The Month: ");
int month= ab.nextInt();
//int month = 4;
switch(month)
{
case 1:
System.out.println("January");
break;
case 2:
System.out.println("February");
break;
case 3:
System.out.println("March");
break;
case 4:
System.out.println("April");
break;
case 5:
```

```
System.out.println("May");
```

```
break;
```

```
case 6:
```

```
System.out.println("June");
```

```
break;
```

```
case 7:
```

```
System.out.println("July");
```

```
break;
```

```
case 8:
```

```
System.out.println("August");
```

```
break;
```

```
case 9:
```

```
System.out.println("September");
```

```
break;
```

```
case 10:
```

```
System.out.println("October");
```

```
break;
```

```
case 11:
```

```
System.out.println("November");
```

```
break;
```

```
case 12:
```

```
System.out.println("December");  
  
break;  
  
default:  
  
System.out.println("Hurray Its Happy New Year!!!");  
  
}  
  
}  
  
}
```

## **2. Loop statements**

### **do while loop:**

#### **Program:**

```
package dcstatements;  
  
public class DMdowhilestatement  
{  
  
    public static void main(String args[])  
    {  
  
        int num = 1;  
  
        do  
  
        {  
  
            System.out.print("The numbers are : " + num );
```

```
num++;  
  
System.out.print("\n");  
  
}while( num < 10 );  
  
}  
  
}
```

## **For Loop Statement:**

### **Program:**

```
package dcstatements;  
  
public class DMForstatement {  
    public static void main(String[] args) {  
        for (int i = 3; i <= 10; i++) {  
            System.out.println(i);  
        }  
        for (int i = 1; i <= 5; i++)  
            System.out.println("Hello World");  
    }  
}
```

## **While Loop Statement:**



## **Program:**

```
package dcstatements;

public class DMWhileststement
{
    public static void main(String[] args)
    {
        int i = 3;
        while (i < 10)
        {
            System.out.println(i);
            i++;
        }
    }
}
```

## **Pramid For Loop:**

### **Program:**

```
package dcstatements;

public class Pyramid
{

```

```
public static void main(String[] args) {  
    for(int i=1;i<=5;i++){  
        for(int j=1;j<=i;j++){  
            System.out.print("* ");  
        }  
        System.out.println();  
    }  
}  
}
```

### **3.Jump statements**

#### **\*Break Statement:**

#### **Program:**

```
package dcstatements;  
  
public class DMBreakstatement  
{  
    public static void main(String[] args)  
    {  
        for(int i=1;i<=10;i++)
```

```
{  
if(i==6)  
{  
break;  
}  
System.out.println(i);  
}  
}
```

### **Continue Statement:**

)

### **Program:**

```
package dcstatements;  
  
public class DMContinuestatement {  
    public static void main(String[] args) {  
        for (int i = 0; i <= 10; i++) {  
            if (i == 4) {  
                continue;  
            }  
            System.out.println(i);  
        }  
    }  
}
```

}

## **OOP's Concepts**

Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

1.Object

2.Class

3.Inheritance

4.Polymorphism

5.Abstraction

6.Encapsulation

**Object-oriented programming has several advantages over procedural programming:**

1.OOP is faster and easier to execute

2.OOP provides a clear structure for the programs

3.OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug

4.OOP makes it possible to create full reusable applications with less code and shorter development time

## 1.Class Method:

### Program:

```
package ClassesandObjects;

public class AClssMethods {

    public static void main(String[] args)

    {

        String targetString = "Java is a Programming Language";

        String str1= "JAVA";

        String str2= "Java";

        String str3 = "    Hello Java    ";

        System.out.println("Checking Length: "+
targetString.length());

        char [] charArray = str2.toCharArray();

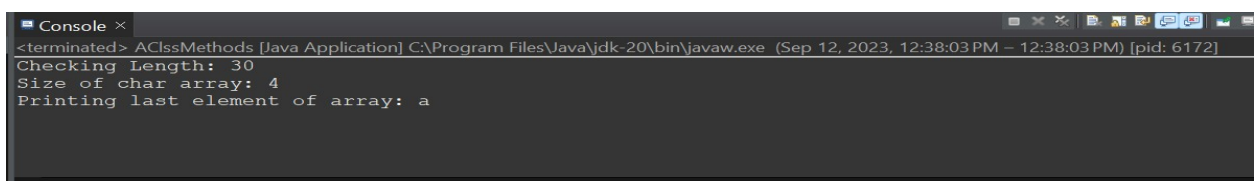
        System.out.println("Size of char array: " + charArray.length);

        System.out.println("Printing last element of array: " +
charArray[3]);

    }

}
```

### Output:

A screenshot of a Windows console window titled "Console". The window shows the output of a Java application. The first line is a system message: "<terminated> AClssMethods [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (Sep 12, 2023, 12:38:03 PM - 12:38:03 PM) [pid: 6172]". The subsequent lines are the program's output: "Checking Length: 30", "Size of char array: 4", and "Printing last element of array: a". The console window has a standard Windows taskbar at the bottom with various icons.

## **Abstraction:**

\*Data abstraction is the process of hiding certain details and showing only essential information to the user.

\*It can be achieved by using abstract classes, methods, and interfaces.

\*An abstract class is a class that cannot be instantiated on its own and is meant to be inherited by concrete classes.

## **Program:**

```
package Oops;
```

```
abstract class Animal {
```

```
    private String name;
```

```
        public Animal(String name) { this.name = name; }
```

```
        public abstract void makeSound();
```

```
        public String getName() { return name; }
```

```
    }
```

```
// Abstracted class
```

```
class Dog extends Animal {  
    public Dog(String name) { super(name); }  
  
    public void makeSound()  
    {  
        System.out.println(getName() + " barks");  
    }  
}  
  
// Abstracted class  
class Cat extends Animal {  
    public Cat(String name) { super(name); }  
    public void makeSound()  
    {  
        System.out.println(getName() + " meows");  
    }  
}  
  
// Driver Class  
public class Abstractionexample{  
    // Main Function  
    public static void main(String[] args)
```

```
{  
    Animal myDog = new Dog("Buddy");  
    Animal myCat = new Cat("Fluffy");  
    myDog.makeSound();  
    myCat.makeSound();  
}  
}
```

### **Program:**

```
package Oops;  
  
abstract class MotorBike {  
    abstract void brake();  
}  
  
class SportsBike extends MotorBike {  
  
    public void brake() {  
        System.out.println("This is Sports Bike Brake");  
    }  
}
```



```
class MountainBike extends MotorBike {  
    public void brake() {  
        System.out.println("This is Mountain Bike Brake");  
    }  
}
```

```
public class Bikes {  
  
    public static void main(String[] args) {  
        MountainBike m1 = new MountainBike();  
        m1.brake();  
        SportsBike s1 = new SportsBike();  
        s1.brake();  
    }  
}
```

### **Program:**

```
package Oops;  
  
public class Books {  
    private String author;
```

```
private String title;
```

```
private int id;
```

```
//public getter and setter method
```

```
public String getAuthor() {
```

```
    return author;
```

```
}
```

```
public void setAuthor(String a) {
```

```
    this.author = a;
```

```
}
```

```
public String getTitle() {
```

```
    return title;
```

```
}
```

```
public void setTitle(String t) {
```

```
    this.title = t;
```

```
}
```

```
public int getId() {
```

```
        return id;
    }

    public void setId(int i) {
        this.id = i;
    }
}
```

## **Convert String to int**

### **Program:**

```
package Oops;

public class ConvertStringtoint {
    public static void main(String args[]) {
        // String literal
        String str = "123";

        // Holds the integer equivalent
        int ans = Integer.parseInt(str);

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

```
    }  
}
```

## **Encapsulation:**

Encapsulation is defined as the wrapping up of data under a single unit.

## **Program:**

```
package Oops;  
  
public class EncapsulationPrgm  
{  
    public static void main(String[] args)  
    {  
        Books b = new Books();  
        b.setAuthor("Jawaharlal Nehru");  
        b.setTitle("An Autobiography");  
        b.setId(23456);  
        System.out.println("Book Title is : " + b.getTitle());  
        System.out.println("Book Author is : " + b.getAuthor());  
        System.out.println("Book Id is : " + b.getId());  
    }  
}
```

## **Program:**

```
package Oops;

public class EncapTest {

    private String name;

    private String idNum;

    private int age;


    public int getAge() {

        return age;

    }


    public String getName() {

        return name;

    }


    public String getIdNum() {

        return idNum;

    }


    public void setAge( int newAge) {

        age = newAge;

    }

}
```

```
public void setName(String newName) {  
    name = newName;  
}
```

```
public void setIdNum( String newId) {  
    idNum = newId;  
}
```

```
}
```

### **Program:**

```
package Oops;
```

```
public class RunEncap {
```

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

```
        EncapTest encap = new EncapTest();
```

```
        encap.setName("James");
```

```
        encap.setAge(20);
```

```
        encap.setIdNum("12343ms");
```

```
        System.out.print("Name : " + encap.getName() + "  
Age : " + encap.getAge());
```

```
    }
```

```
}
```

**Program:**

```
package Oops;

interface DrawShapes

{
void draw();
}

class Rectangle implements DrawShapes {
public void draw() {
System.out.println("Rectangle will be drawn");
}
}

class Circle implements DrawShapes
{
public void draw() {
System.out.println("Circle will be drawn");
}
}

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

```
DrawShapes shape = new Circle();  
shape.draw();  
}  
}
```

## **Inheritance:**

\*Inheritance means creating new classes based on existing ones.

\*A class that inherits from another class can reuse the methods and fields of that class.

\*In addition, you can add new fields and methods to your current class as well.

### **The 4 types of Inheritance in Java:**

- 1.Single-level inheritance.
- 2.Multi-level Inheritance.
- 3.Hierarchical Inheritance.
- 4.Hybrid Inheritance.

#### **1.Single-level inheritance:**

\*In single inheritance, a sub-class is derived from only one super class.

\*It inherits the properties and behavior of a single-parent class. Sometimes it is also known as simple inheritance.

#### **Program:**



```
package Inheritance;

class Person
{
    String name="Aiswarya";

    public void show()
    {
        System.out.println("Student inheriting properties from
Person");
    }
}
```

```
class SingleInherit extends Person
{

    String course = "Studying Course at Edubridge";

    public void show1()
    {
        System.out.println("I am a Student");
    }

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

```
        SingleInherit stu = new SingleInherit();

        stu.show();

        stu.show1();

        System.out.println("Name of student: " +stu.name);

        System.out.println("Course done by the student is : "
+stu.course);

    }

}
```

## **2.Multi-level Inheritance:**

\*In multi-level inheritance, a class is derived from a class which is also derived from another class is called multi-level inheritance.

\* In simple words, we can say that a class that has more than one parent class is called multi-level inheritance.

\*Hence, there exists a single base class and single derived class but multiple intermediate base classes.

### **Program:**

```
package Inheritance;

class Persons

{

    public void show()

    {
```

```
        System.out.println("Student inheriting properties from  
Person");
```

```
    }  
}
```

```
class Students extends Persons
```

```
{  
    public void show1()  
    {  
        System.out.println("I am a Student");  
    }  
}
```

```
//child class
```

```
class EngineeringStudent extends Students
```

```
{  
    // defining additional properties to the child class  
    public void show2()  
    {  
        System.out.println("I am a Engineering Student");  
    }  
}
```

```
public class MultilevelInherit
```

```

{
    public static void main(String args[])
    {
        EngineeringStudent stud = new EngineeringStudent();
        stud.show();
        stud.show1();
        stud.show2();
    }
}

```

### **3.Hierarchical Inheritance:**

If a number of classes are derived from a single base class, it is called hierarchical inheritance.

#### **Program:**

```

package Inheritance;

class Personss
{
    public void show()
    {
        System.out.println("I am a Person");
    }
}

```

```
}
```

```
class Student extends Personss
```

```
{
```

```
    public void show1()
```

```
    {
```

```
        System.out.println("I am a Student");
```

```
    }
```

```
}
```

```
class Teachers extends Personss
```

```
{
```

```
    public void show2()
```

```
    {
```

```
        System.out.println("My Teacher Name is Mohana Priya");
```

```
    }
```

```
}
```

```
class Course extends Person
```

```
{
```

```
    public void show3()
```

```
    {
```

```
        System.out.println("My Course Name is Software Testing");
```

```

    }
}
public class HierarchicalInherit
{
    public static void main(String args[])
    {
        Teachers teacher = new Teachers();
        Student student = new Student();
        Course course = new Course();
        student.show();
        student.show1();
        teacher.show2();
        course.show3();
    }
}

```

## **Arrays:**

\*an array is a collection of similar type of elements which has contiguous memory location.

\*The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.

\*Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.

## **Types of Array in java:**

There are two types of array:

- 1.Single Dimensional Array
- 2.Multidimensional Array

### **1.Single Dimensional Array:**

\*Single Dimensional Array in Java is basically a linear array that allows its user to store multiple values of the same data type.

\*It's a collection of data that stores elements of the same type in a sequentially allocated space in memory.

### **Program:**

```
package Arrays;

public class SimpleArray
{
    //Simple Array Program

    public static void main(String args[]){

        String[] colors = {"Pink", "Blue", "Green", "Red", "Yellow",
        "Orange"};
```

```
System.out.println(colors[3]);

//Length of Array

int a[]=new int[5];

a[0]=10;

a[1]=20;

a[2]=30;

a[3]=40;

a[4]=50;

for(int i=0;i<a.length;i++)

System.out.println(a[i]);

System.out.println("*****For Each Loop
Array*****");

//For Each Loop Array

int arr[]={33,76,42,58};

for(int i:arr)

System.out.println(i);

}

}
```



## 2.Multidimensional Array:

Multidimensional Arrays can be defined in simple words as array of arrays. Data in multidimensional arrays are stored in tabular form (in row major order).

### Program:

```
package Arrays;

public class MultidimensionalArray
{
    public static void main(String[] args)
    {
        int[][] num = { {1, 2, 3}, {4, 5, 6, 9}, {6, 8, 9, 5, 3, 2} };

        System.out.println("The Index of 0 is : " + num[0][2]); //It Prints
        the Output as 3

        System.out.println("The Index of 1 is : " + num[1][3]); //It Prints
        the Output as 9

        System.out.println("The Index of 2 is : " + num[2][5]); //It Prints
        the Output as 2

        //System.out.println("The Index of 2 is : " + num[2][7]); //It Shows
        an Error because the index number of 7 does not have any value

        //.length

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

        for (int i = 0; i < myNumbers.length; ++i) {
```

```
        for(int j = 0; j < myNumbers[i].length; ++j) {  
            System.out.println(myNumbers[i][j]);  
        }  
    }  
}
```

## **Exceptional Handling:**

The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

### **Types of Java Exceptions:**

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

- 1.Checked Exception
- 2.Unchecked Exception
- 3.Error

### **SampleProgram:**

```
package ExceptionsHandling;  
  
public class SampleCode {  
    public static void main(String[] args) {  
        try {
```

```

        int arr[] = {1,2,3,4,5};

        System.out.println(arr[10]);
    } catch (ArrayIndexOutOfBoundsException e)
    {

        System.out.println("ArrayIndexOutOfBoundsException");

        } finally {

            System.out.println("finally block");

        }

    }

}

```

### **Program:**

```

package ExceptionsHandling;

public class Pgm2 {

    public static void main(String args[]) {

        try {

            int a = 15;

```

```
        int b = 0;

        System.out.println("Value of  a = " + a);
        System.out.println("Value of  b = " + b);

        int c = a / b;

        System.out.println("a / b = " + c);
    }
    catch (Exception e) {
        System.out.println("Exception Thrown: " + e);
    }
    finally {
        System.out.println("Finally block executed!");
    }
}
}
```

### **Program:**

```
package ExceptionsHandling;

public class Pgm3 {

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

```
try {  
    int a = 10;  
  
    int b = 5;  
  
    System.out.println("Value of  a = " + a);  
    System.out.println("Value of  b = " + b);  
  
    int c = a / b;  
    System.out.println("a / b = " + c);  
}  
catch (Exception e) {  
    System.out.println("Exception Thrown: " + e);  
}  
finally {  
    System.out.println("Finally block has  
executed!");  
}  
}
```

## Throw Exception:

The Java throw keyword is used to throw an exception explicitly. We specify the exception object which is to be thrown.

## Program:

```
package ExceptionsHandling;

import java.util.Scanner;

public class ThrowExce {

    public static void main(String[] args) {

        Scanner s = new Scanner(System.in);

        System.out.println("Please enter your roll
number");

        int roll = s.nextInt();

        if (roll < 0) {

            throw new ArithmeticException("Roll number
can't be negative");

        } else {

            System.out.println("Valid roll number");

        }

    }

}
```

## Throws Exception:

\*The Java throws keyword is used to declare an exception.

\*It gives an information to the programmer that there may occur an exception.

## Program:

```
package ExceptionsHandling;

public class ThrowsException {

    public static void calculate()

        throws ArithmeticException,
        ArrayIndexOutOfBoundsException {

        int num = 10 / 0;

        // some code that might throw
        ArrayIndexOutOfBoundsException

    }


    public static void main(String[] args) {

        try {

            calculate();

        } catch (ArithmeticException e) {

            System.out.println("Arithmetic Exception
thrown");

            System.out.println(e.getMessage());
```

```
        } catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println("ArrayIndexOutOfBoundsException  
Exception thrown");  
            System.out.println(e.getMessage());  
        }  
    }  
}
```

## **Constructor in Java :**

- \*Constructor in java is used to create the instance of the class.
- \*Constructors are almost similar to methods except for two things - its name is the same as the class name and it has no return type.
- \*Sometimes constructors are also referred to as special methods to initialize an object.

## **Types of Java constructors:**

**There are two types of constructors in Java:**

- 1.Default constructor (no-arg constructor)
- 2.Parameterized constructor

## **Program:**

```
package Constructorsprogms;  
  
public class SampleCons {
```



```

        private String name;

// constructor

        SampleCons() {

            System.out.println("Constructor Called:");

            name = "Aiswarya";

        }

public static void main(String[] args) {

    // constructor is invoked while

    // creating an object of the Main class

        SampleCons obcons = new SampleCons();

        System.out.println("Myname name is " + obcons.name);

    }

}

```

## **Collections:**

\*The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects.

\*Java Collections can achieve all the operations that you perform on a data such as **searching, sorting, insertion, manipulation, and deletion.**

\*Java Collection means a single unit of objects. Java Collection framework provides many **interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).**

**Interfaces:**

Set

List

Queue

Deque

**Classes:**

ArrayList

Vector

LinkedList

PriorityQueue

HashSet

LinkedHashSet

TreeSet

**ArrayList:****Program:**

```
package Collections;  
  
import java.util.ArrayList;  
  
import java.util.Collections;  
  
public class ArrListPrgm {
```

```
public static void main(String[] args) {  
    ArrayList<String> colors = new ArrayList<String>();  
    colors.add("Pink");  
    colors.add("Blue");  
    colors.add("Green");  
    colors.add("Red");  
    colors.add("Yellow");  
    colors.add(1,"Indigo");  
    System.out.println("The Colors that given here are :  
"+colors);  
    //Access an Item  
    //use the get() method and refer to the index number  
    System.out.println("The Color of an Array Index is :  
"+colors.get(1));  
    //ArrayList Size  
    //To find out how many elements an ArrayList have,  
    use the size method  
    // System.out.println("The Size of Colors List are :  
"+colors.size());  
    //Change an Item
```

//To modify an element, use the set() method and refer to the index number

```
colors.set(2, "Merun");
```

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

//Remove an Item

//To remove an element, use the remove() method and refer to the index number

```
colors.remove(2);
```

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

//ArrayList Size

//To find out how many elements an ArrayList have, use the size method

```
System.out.println("The Size of Colors List are :  
"+colors.size());
```

//Loop Through an ArrayList

//Loop through the elements of an ArrayList with a for loop, and use the size() method to specify how many times the loop should run

```
for (int i = 0; i < colors.size(); i++) {
```

```
    System.out.println(colors.get(i));
```

```
}
```

//Sort an ArrayList

```
        Collections.sort(colors);  
    for (String i : colors) {  
        System.out.println(i);  
    }  
  
    //To remove all the elements in the ArrayList, use the clear()  
method  
  
    colors.clear();  
  
    System.out.println(colors);  
}  
}
```

## **Number for Array List:**

### **Program:**

```
package Collections;  
  
import java.util.ArrayList;  
import java.util.Collections;  
  
public class NumbersforArrayList {  
    public static void main(String[] args) {  
        ArrayList<Integer> num = new ArrayList<Integer>();  
        num.add(33);  
        num.add(15);  
    }  
}
```

```
        num.add(20);

        num.add(34);

        num.add(8);

        num.add(12);

Collections.sort(num);

for (int i : num) {

        System.out.println(i);

    }

}
```

## **LinkedList:**

### **Program:**

```
package Collections;

import java.util.LinkedList;

public class Prgm2forLinkedList {

    //This is way is to store values of different datatype

    public static void main(String[] args) {

        LinkedList name = new LinkedList();

        name.add(10);
```

```
name.add("Aiswarya");  
name.add(1.23);  
name.add(true);  
name.add('A');  
System.out.println(name);  
}  
}
```

## **HashMap:**

### **Program:**

```
package Collections;  
import java.util.HashMap;  
public class PrgmforHashMap {  
    public static void main(String[] args) {  
        HashMap<String, String> employeeNames = new  
HashMap<String, String>();  
        employeeNames.put("Keerthana", "Lakshmi");  
        employeeNames.put("Divya", "Sreedhar");  
        employeeNames.put("Karthiga", "Bhavani");  
        employeeNames.put("Gayathri", "Kumar");  
        System.out.println(employeeNames);  
        //printing Print values
```

```

        for (String i : employeeNames.values()) {
            System.out.println(i);
        }

        //For Keyset
        for (String i : employeeNames.keySet()) {
            System.out.println(i);

            //for Key Values
            for (String j : employeeNames.keySet()) {
                System.out.println("key: " + j + " value: " +
employeeNames.get(j));
            }
        }
    }
}

```

## **Activity Programs:**

### **All Pairs of Elements:**

#### **Programs:**

```

package Activity;

public class AllPairsOfElements4
{

```



```

static void findThePairs(int inputArray[], int inputNumber)
{
    System.out.println("Pairs of elements whose sum is
    "+inputNumber+" are : ");
    for (int i = 0; i < inputArray.length; i++)
    {
        for (int j = i+1; j < inputArray.length; j++)
        {
            if(inputArray[i]+inputArray[j] == inputNumber)
            {
                System.out.println(inputArray[i]+" + "+inputArray[j]+" =
                "+inputNumber);
            }
        }
    }
}

public static void main(String[] args)
{
    findThePairs(new int[] {4, 6, 5, -10, 8, 5, 20}, 10);
    findThePairs(new int[] {4, -5, 9, 11, 25, 13, 12, 8}, 20);
    findThePairs(new int[] {12, 13, 40, 15, 8, 10, -15}, 25);
}

```

```
findThePairs(new int[] {12, 23, 125, 41, -75, 38, 27, 11}, 50);  
}  
}
```

Second Largest Number:

Program:

```
import java.util.*;  
  
public class SecondLargestArrayElement  
{  
    public static void main(String[] args)  
    {  
        Scanner sc=new Scanner(System.in);  
  
        int n=sc.nextInt();  
  
        int a[]=new int[n];  
  
        for(int i=0;i<n;i++)  
            a[i]=sc.nextInt();  
  
        Arrays.sort(a);  
  
        System.out.print("The second largest number: "+a[n-2]);  
  
        /2nd Method user defined method  
  
        /*System.out.println("*****User  
        Defined Method*****");  
  
        int t, size;
```

```

int arr[] = {1, 28, 38, 93, 46, 97};

size = arr.length;

for(int i = 0; i<size; i++ ){
    for(int j = i+1; j<size; j++){
        if(arr[i]>arr[j]){
            t = arr[i];
            arr[i] = arr[j];
            arr[j] = t;
        }
    }
}

System.out.println("2nd Largest Number: "+arr[size-2]);

//Define a number array and add the element inside the array:

System.out.println("*****
**By Using Sort
Method*****");

int array[] = {67, 89, 52, 63, 78, 75};

int sizes = arr.length;

Arrays.sort(array);

System.out.println("Array ::"+Arrays.toString(array));

int result = array[sizes-2];

```

```
System.out.println("Second Largest ::"+result);*/  
}  
}
```

## **Reversing an Array:**

### **Program:**

```
package Activity;  
import java.util.*;  
public class ReverseanArray10 {  
    public static void main(String[] args)  
    {  
        int n, res,i,j=0;  
        Scanner s = new Scanner(System.in);  
        System.out.print("Enter number of elements in the  
array:");  
        n = s.nextInt();  
        int array[] = new int[n];  
        int rev[] = new int[n];  
        System.out.println("Enter "+n+" elements ");  
        for( i=0; i < n; i++)  
        {
```

```

        array[i] = s.nextInt();
    }
    System.out.println("Reverse of an array is :");
    for( i=n;i>0 ; i--,j++)
    {
        rev[j] = array[i-1];
        System.out.println(rev[j]);
    }
}
}

```

## **Duplicating an Array Element:**

### **Program:**

```

package Activity;

public class DuplicateArrayElement
{
    public static void main(String[] args)
    {
        int [] num = new int [] {1, 2, 6, 8, 4, 6, 10, 10, 4};
    }
}

```

```
System.out.println("Duplicate elements in given array are: ");  
for(int i = 0; i < num.length; i++)  
{  
    for(int j = i + 1; j < num.length; j++)  
    {  
        if(num[i] == num[j])  
        {  
            System.out.println(num[j]);  
        }  
    }  
}  
}
```

## **Armstrong Number:**

### **Program:**

```
package SimplePrograms;  
  
import java.util.*;  
  
public class ArmstrongNum2UserInput {  
    public static void main(String[] args)  
    {
```

```
int n, count = 0, a, b, c, sum = 0;

Scanner s = new Scanner(System.in);

System.out.print("Enter a number:");

n = s.nextInt();

a = n;

c = n;

while(a > 0)

{

a = a / 10;

count++;

}

while(n > 0)

{

b = n % 10;

sum = (int) (sum+Math.pow(b, count));

n = n / 10;

}

if(sum == c)

{

System.out.println(c+ " is an Armstrong number");

}
```

else

{

System.out.println(c+ " is not an Armstrong number");

}

}

}