1. Creating Hello World Example

```
class Simple
{
   System.out.println("Hello Java");
   }

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

To compile: javac Simple.java

To execute: java Simple

2. Example to print the classloader name

```
//Let's see an example to print the classloader name
public class ClassLoaderExample
{
   public static void main(String[] args)
   {
      // Let's print the classloader name of current class.
     //Application/System classloader will load this class
      Class c=ClassLoaderExample.class;
      System.out.println(c.getClassLoader());
      //If we print the classloader name of String, it will print null because it is an
```

```
        //in-built class which is found in rt.jar, so it is loaded by Bootstrap
    classloader
        System.out.println(String.class.getClassLoader());
      }
    }
```

3. Example to understand the types of variables in java

```
class A
{
int data=50;//instance variable
static int m=100;//static variable
void method(){
int n=90;//local variable
}
}//end of class
```

4. Java Variable Example: Add Two Numbers

```
class Simple
{
public static void main(String[] args)
{
int a=10;
int b=10;
```

```
int c=a+b;

System.out.println(c);

}

}
```

5. Java Variable Example: Widening

```
class Simple

{

public static void main(String[] args)

{

int a=10;

float f=a;

System.out.println(a);

System.out.println(f);

}

}
```

6. Java Variable Example: Narrowing (Typecasting)

```
class Simple

{

public static void main(String[] args)

{
```

```
float f=10.5f;

//int a=f;//Compile time error

int a=(int)f;

System.out.println(f);

System.out.println(a);

}

}
```

7. Java Variable Example: Overflow

```
class Simple

{

public static void main(String[] args)

{

//Overflow

int a=130;

byte b=(byte)a;

System.out.println(a);

System.out.println(b);

}

}
```

8. Java Variable Example: Adding Lower Type

```
class Simple

{

public static void main(String[] args)

{

byte a=10;

byte b=10;

//byte c=a+b;//Compile Time Error: because a+b=20 will be int

byte c=(byte)(a+b);

System.out.println(c);

}

}
```

9. Example of static variable

```
public class StaticVarExample
{
  public static String myClassVar="class or static variable";

  public static void main(String args[])
{
    StaticVarExample obj = new StaticVarExample();
    StaticVarExample obj2 = new StaticVarExample();
    StaticVarExample obj3 = new StaticVarExample();

    //All three will display "class or static variable"
    System.out.println(obj.myClassVar);
    System.out.println(obj2.myClassVar);
    System.out.println(obj3.myClassVar);

    //changing the value of static variable using obj2
    obj2.myClassVar = "Changed Text";
```

```
        //All three will display "Changed Text"
        System.out.println(obj.myClassVar);
        System.out.println(obj2.myClassVar);
        System.out.println(obj3.myClassVar);
    }
}
```

10. Example of Instance variable

```
public class InstanceVarExample
{
    String myInstanceVar="instance variable";

    public static void main(String args[])
{
        InstanceVarExample obj = new InstanceVarExample();
        InstanceVarExample obj2 = new InstanceVarExample();
        InstanceVarExample obj3 = new InstanceVarExample();

        System.out.println(obj.myInstanceVar);
        System.out.println(obj2.myInstanceVar);
        System.out.println(obj3.myInstanceVar);

        obj2.myInstanceVar = "Changed Text";

        System.out.println(obj.myInstanceVar);
        System.out.println(obj2.myInstanceVar);
        System.out.println(obj3.myInstanceVar);
    }
}
```

11. Example of Local variable

```
public class VariableExample {
    // instance variable
    public String myVar="instance variable";

    public void myMethod(){
        // local variable
        String myVar = "Inside Method";
        System.out.println(myVar);
```

```
    }
    public static void main(String args[]){
      // Creating object
      VariableExample obj = new VariableExample();

      /* We are calling the method, that changes the
       * value of myVar. We are displaying myVar again after
       * the method call, to demonstrate that the local
       * variable scope is limited to the method itself.
       */
      System.out.println("Calling Method");
      obj.myMethod();
      System.out.println(obj.myVar);
    }
}
```

12. Java Unary Operator Example: ++ and --

```
class OperatorExample{

public static void main(String args[]){

int x=10;

System.out.println(x++);//10 (11)

System.out.println(++x);//12

System.out.println(x--);//12 (11)

System.out.println(--x);//10

}

}
```

13. Java Unary Operator Example 2: ++ and --

```
public static void main(String args[]){
int a=10;
int b=10;
System.out.println(a++ + ++a);//10+12=22
System.out.println(b++ + b++);//10+11=21
}
```

14. Java Arithmetic Operator Example

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
System.out.println(a+b);//15
System.out.println(a-b);//5
System.out.println(a*b);//50
System.out.println(a/b);//2
System.out.println(a%b);//0
}
```

```
}
```

15. **Java Arithmetic Operator Example: Expression**

```
class OperatorExample{
public static void main(String args[]){
System.out.println(10*10/5+3-1*4/2);
}
}
```

16. **Java Left Shift Operator Example**

```
class OperatorExample{
public static void main(String args[]){
System.out.println(10<<2);//10*2^2=10*4=40
System.out.println(10<<3);//10*2^3=10*8=80
System.out.println(20<<2);//20*2^2=20*4=80
System.out.println(15<<4);//15*2^4=15*16=240
}
}
```

17. **Java Right Shift Operator Example**

```
class OperatorExample{
public static void main(String args[]){
System.out.println(10>>2);//10/2^2=10/4=2
System.out.println(20>>2);//20/2^2=20/4=5
System.out.println(20>>3);//20/2^3=20/8=2
}
}
```

18. Java Shift Operator Example: >> vs >>>

```
class OperatorExample{

public static void main(String args[]){

 //For positive number, >> and >>> works same

System.out.println(20>>2);

System.out.println(20>>>2);

 //For negative number, >>> changes parity bit (MSB) to 0

 System.out.println(-20>>2);

  System.out.println(-20>>>2);

}

}
```

19. Java AND Operator Example: Logical && and Bitwise &

```
class OperatorExample{

public static void main(String args[]){

int a=10;

int b=5;

int c=20;

System.out.println(a<b&&a<c);//false && true = false

System.out.println(a<b&a<c);//false & true = false

}
```

```
}
```

20. Java AND Operator Example: Logical && vs Bitwise &

```
class OperatorExample{

public static void main(String args[]){

int a=10;

int b=5;

int c=20;

System.out.println(a<b&&a++<c);//false && true = false

System.out.println(a);//10 because second condition is not checked

System.out.println(a<b&a++<c);//false && true = false

System.out.println(a);//11 because second condition is checked

}

}
```

21. Java OR Operator Example: Logical || and Bitwise |

```
class OperatorExample{

public static void main(String args[]){

int a=10;

int b=5;

int c=20;

System.out.println(a>b||a<c);//true || true = true

System.out.println(a>b|a<c);//true | true = true

//|| vs |
```

```
System.out.println(a>b||a++<c);//true || true = true
System.out.println(a);//10 because second condition is not checked
System.out.println(a>b|a++<c);//true | true = true
System.out.println(a);//11 because second condition is checked
}
}
```

22. Java Ternary Operator Example

```
class OperatorExample{
public static void main(String args[]){
int a=2;
int b=5;
int min=(a<b)?a:b;
System.out.println(min);
}
}
```

Another:

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int min=(a<b)?a:b;
System.out.println(min);
}
```

```
}
```

23. Java Assignment Operator Example

```
class OperatorExample{

public static void main(String args[]){

int a=10;

int b=20;

a+=4;//a=a+4 (a=10+4)

b-=4;//b=b-4 (b=20-4)

System.out.println(a);

System.out.println(b);

}

}
```

24. Java Assignment Operator Example

```
class OperatorExample{

public static void main(String[] args){

int a=10;

a+=3;//10+3

System.out.println(a);

a-=4;//13-4

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

```
a*=2;//9*2

System.out.println(a);

a/=2;//18/2

System.out.println(a);

}

}
```

25. Java Assignment Operator Example: Adding short

```
class OperatorExample{

public static void main(String args[]){

short a=10;

short b=10;

//a+=b;//a=a+b internally so fine

a=a+b;//Compile time error because 10+10=20 now int

System.out.println(a);

}}
```

Another Typecast:

```
class OperatorExample{

public static void main(String args[]){

short a=10;
```

```
short b=10;
a=(short)(a+b);//20 which is int now converted to short
System.out.println(a);
}}
```

26. Miscellaneous Operators

```
public class Test {

   public static void main(String args[]) {
      int a, b;
      a = 10;
      b = (a == 1) ? 20: 30;
      System.out.println( "Value of b is : " +  b );

      b = (a == 10) ? 20: 30;
      System.out.println( "Value of b is : " + b );
   }
}
```

27.

```
public class Test {

   public static void main(String args[]) {
    String name = "James";
    // following will return true since name is type of String
```

```
        boolean result = name instanceof String;
        System.out.println( result );
    }
    }
```

28.
```
class Vehicle {}
public class Car extends Vehicle {

  public static void main(String args[]) {

     Vehicle a = new Car();
     boolean result =  a instanceof Car;
     System.out.println( result );
  }
  }
```

29. Complete-arithmetic operators

```
class Arithmetic_operators1{
public static void main(String as[])
   {
      int a, b, c;
      a=10;
      b=2;
      c=a+b;
      System.out.println("Addtion: "+c);
      c=a-b;
      System.out.println("Substraction: "+c);
```

```
            c=a*b;
            System.out.println("Multiplication: "+c);
            c=a/b;
            System.out.println("Division: "+c);
            b=3;
            c=a%b;
            System.out.println("Remainder: "+c);
            a=++a;
            System.out.println("Increment Operator: "+a);
            a=--a;
            System.out.println("decrement Operator: "+a);

        }

    }
```

30. Relational Operators

```
class Relational_operators1{
public static void main(String as[])
   {
      int a, b;
      a=40;
      b=30;
      System.out.println("a == b = " + (a == b) );
      System.out.println("a != b = " + (a != b) );
      System.out.println("a > b = " + (a > b) );
      System.out.println("a < b = " + (a < b) );
      System.out.println("b >= a = " + (b >= a) );
      System.out.println("b <= a = " + (b <= a) );
   }
```

```
}
```

31. Logical Operator

```
class Logical_operators1{
public static void main(String as[])
   {
      boolean a = true;
      boolean b = false;
      System.out.println("a && b = " + (a&&b));
      System.out.println("a || b = " + (a||b) );
System.out.println("!(a && b) = " + !(a && b));
   }

}
```

32. Bitwise Operator

```
class Bitwise_operators1{
public static void main(String as[])
   {
      int a = 50;
      int b = 25;
      int c = 0;

      c = a & b;
      System.out.println("a & b = " + c );

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

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

            c = ~a;
            System.out.println("~a = " + c );

            c = a << 2;
            System.out.println("a << 2 = " + c );

            c = a >> 2;
            System.out.println("a >>2  = " + c );

            c = a >>> 2;
            System.out.println("a >>> 2 = " + c );
         }

      }
```

33. Assignment Operator

```
class Assignment_operators1{

public static void main(String as[])

{

   int a = 30;

   int b = 10;
```

```
    int c = 0;
      c = a + b;
System.out.println("c = a + b = " + c );
      c += a ;
System.out.println("c += a  = " + c );
      c -= a ;
System.out.println("c -= a = " + c );
      c *= a ;
System.out.println("c *= a = " + c );
      a = 20;
      c = 25;
      c /= a ;
System.out.println("c /= a = " + c );
      a = 20;
      c = 25;
      c %= a ;
System.out.println("c %= a  = " + c );
      c <<= 2 ;
```

```
System.out.println("c <<= 2 = " + c );

    c >>= 2 ;

System.out.println("c >>= 2 = " + c );

    c >>= 2 ;

System.out.println("c >>= 2 = " + c );

    c &= a ;

System.out.println("c &= a  = " + c );

    c ^= a ;

System.out.println("c ^= a   = " + c );

    c |= a ;

System.out.println("c |= a   = " + c );

  }

}
```

34. ----------Conditional----------

```
//Java Program to demonstate the use of if statement.
public class IfExample {
public static void main(String[] args) {
   //defining an 'age' variable
```

```
        int age=20;
        //checking the age
        if(age>18){
           System.out.print("Age is greater than 18");
        }
    }
    }
```

35. ----------

```
//A Java Program to demonstrate the use of if-else statement.
//It is a program of odd and even number.
public class IfElseExample {
public static void main(String[] args) {
   //defining a variable
   int number=13;
   //Check if the number is divisible by 2 or not
   if(number%2==0){
      System.out.println("even number");
   }else{
      System.out.println("odd number");
   }
}
}
```

36. ----------

```
public class LeapYearExample {
public static void main(String[] args) {
   int year=2020;
   if(((year % 4 ==0) && (year % 100 !=0)) || (year % 400==0)){
      System.out.println("LEAP YEAR");
   }
   else{
      System.out.println("COMMON YEAR");
   }
}
}
```

37. ----------

```
//Java Program to demonstrate the use of If else-if ladder.
//It is a program of grading system for fail, D grade, C grade, B grade, A grade and A+.
public class IfElseIfExample {
public static void main(String[] args) {
   int marks=65;

   if(marks<50){
      System.out.println("fail");
   }
   else if(marks>=50 && marks<60){
      System.out.println("D grade");
```

```
    }
    else if(marks>=60 && marks<70){
       System.out.println("C grade");
    }
    else if(marks>=70 && marks<80){
       System.out.println("B grade");
    }
    else if(marks>=80 && marks<90){
       System.out.println("A grade");
    }else if(marks>=90 && marks<100){
       System.out.println("A+ grade");
    }else{
       System.out.println("Invalid!");
    }
  }
  }
```

38. ----------

```
public class PositiveNegativeExample {
public static void main(String[] args) {
   int number=-13;
   if(number>0){
   System.out.println("POSITIVE");
   }else if(number<0){
   System.out.println("NEGATIVE");
```

```
    }else{
    System.out.println("ZERO");
   }
  }
  }
```

39. ----------

```
//Java Program to demonstrate the use of Nested If Statement.
public class JavaNestedIfExample {
public static void main(String[] args) {
   //Creating two variables for age and weight
   int age=20;
   int weight=80;
   //applying condition on age and weight
   if(age>=18){
      if(weight>50){
         System.out.println("You are eligible to donate blood");
      }
   }
}}
```

40. ----------

```
//Java Program to demonstrate the use of Nested If Statement.
public class JavaNestedIfExample2 {
public static void main(String[] args) {
   //Creating two variables for age and weight
   int age=25;
   int weight=48;
   //applying condition on age and weight
   if(age>=18){
      if(weight>50){
         System.out.println("You are eligible to donate blood");
      } else{
         System.out.println("You are not eligible to donate blood");
      }
   } else{
     System.out.println("Age must be greater than 18");
   }
} }
```

41. ----------

Example of if-else-if

```
public class IfElseIfExample {


   public static void main(String args[]){

      int num=1234;
```

```
        if(num <100 && num>=1) {
          System.out.println("Its a two digit number");
        }
        else if(num <1000 && num>=100) {
          System.out.println("Its a three digit number");
        }
        else if(num <10000 && num>=1000) {
          System.out.println("Its a four digit number");
        }
        else if(num <100000 && num>=10000) {
          System.out.println("Its a five digit number");
        }
        else {
          System.out.println("number is not between 1 & 99999");
        }
    }
}
```

42. ----------

```
public class SwitchExample {
public static void main(String[] args) {
   //Declaring a variable for switch expression
   int number=20;
```

```
    //Switch expression
    switch(number){
    //Case statements
    case 10: System.out.println("10");
    break;
    case 20: System.out.println("20");
    break;
    case 30: System.out.println("30");
    break;
    //Default case statement
    default:System.out.println("Not in 10, 20 or 30");
    }
}
}
```

43. ----------

```
//Java Program to demonstrate the example of Switch statement
//where we are printing month name for the given number
public class SwitchMonthExample {
public static void main(String[] args) {
    //Specifying month number
    int month=7;
    String monthString="";
    //Switch statement
    switch(month){
```

```
//case statements within the switch block
case 1: monthString="1 - January";
break;
case 2: monthString="2 - February";
break;
case 3: monthString="3 - March";
break;
case 4: monthString="4 - April";
break;
case 5: monthString="5 - May";
break;
case 6: monthString="6 - June";
break;
case 7: monthString="7 - July";
break;
case 8: monthString="8 - August";
break;
case 9: monthString="9 - September";
break;
case 10: monthString="10 - October";
break;
case 11: monthString="11 - November";
break;
case 12: monthString="12 - December";
break;
```

```
    default:System.out.println("Invalid Month!");
    }
    //Printing month of the given number
    System.out.println(monthString);
  }
  }
```

44. ----------

```
public class SwitchVowelExample {
public static void main(String[] args) {
    char ch='O';
    switch(ch)
    {
       case 'a':
          System.out.println("Vowel");
          break;
       case 'e':
          System.out.println("Vowel");
          break;
       case 'i':
          System.out.println("Vowel");
          break;
       case 'o':
          System.out.println("Vowel");
          break;
```

```
            case 'u':
                System.out.println("Vowel");
                break;
            case 'A':
                System.out.println("Vowel");
                break;
            case 'E':
                System.out.println("Vowel");
                break;
            case 'I':
                System.out.println("Vowel");
                break;
            case 'O':
                System.out.println("Vowel");
                break;
            case 'U':
                System.out.println("Vowel");
                break;
            default:
                System.out.println("Consonant");
        }
    }
    }
```

45. ----------

```
//Java Switch Example where we are omitting the
//break statement
public class SwitchExample2 {
public static void main(String[] args) {
   int number=20;
   //switch expression with int value
   switch(number){
   //switch cases without break statements
   case 10: System.out.println("10");
   case 20: System.out.println("20");
   case 30: System.out.println("30");
   default:System.out.println("Not in 10, 20 or 30");
   }
}
}
```

46. ----------

```
//Java Program to demonstrate the use of Java Switch
//statement with String
public class SwitchStringExample {
public static void main(String[] args) {
   //Declaring String variable
   String levelString="Expert";
   int level=0;
   //Using String in Switch expression
```

```
    switch(levelString){
    //Using String Literal in Switch case
    case "Beginner": level=1;
    break;
    case "Intermediate": level=2;
    break;
    case "Expert": level=3;
    break;
    default: level=0;
    break;
    }
    System.out.println("Your Level is: "+level);
}
}
```

47. ----------

```
//Java Program to demonstrate the use of Java Nested Switch
public class NestedSwitchExample {
    public static void main(String args[])
     {
     //C - CSE, E - ECE, M - Mechanical
      char branch = 'C';
      int collegeYear = 4;
      switch( collegeYear )
      {
```

```
case 1:
   System.out.println("English, Maths, Science");
   break;
case 2:
   switch( branch )
   {
      case 'C':
         System.out.println("Operating System, Java, Data Structure");
         break;
      case 'E':
         System.out.println("Micro processors, Logic switching theory");
         break;
      case 'M':
         System.out.println("Drawing, Manufacturing Machines");
         break;
   }
   break;
case 3:
   switch( branch )
   {
      case 'C':
         System.out.println("Computer Organization, MultiMedia");
         break;
      case 'E':
```

```
                System.out.println("Fundamentals of Logic Design,
Microelectronics");
                break;
            case 'M':
                System.out.println("Internal Combustion Engines, Mechanical
Vibration");
                break;
        }
        break;
      case 4:
        switch( branch )
        {
          case 'C':
                System.out.println("Data Communication and Networks,
MultiMedia");
                break;
            case 'E':
                System.out.println("Embedded System, Image Processing");
                break;
            case 'M':
                System.out.println("Production Technology, Thermal
Engineering");
                break;
        }
        break;
    }
```

```
    }
}
```

48. ----------

```
//Java Program to demonstrate the use of Enum
//in switch statement
public class JavaSwitchEnumExample {
    public enum Day {  Sun, Mon, Tue, Wed, Thu, Fri, Sat  }
    public static void main(String args[])
    {
     Day[] DayNow = Day.values();
      for (Day Now : DayNow)
      {
          switch (Now)
          {
             case Sun:
                System.out.println("Sunday");
                break;
             case Mon:
                System.out.println("Monday");
                break;
             case Tue:
                System.out.println("Tuesday");
                break;
             case Wed:
```

```
                System.out.println("Wednesday");
                break;
            case Thu:
                System.out.println("Thursday");
                break;
            case Fri:
                System.out.println("Friday");
                break;
            case Sat:
                System.out.println("Saturday");
                break;
        }
    }
}
```

49. ----------

```
//Java Program to demonstrate the use of Wrapper class
//in switch statement
public class WrapperInSwitchCaseExample {
    public static void main(String args[])
    {
        Integer age = 18;
        switch (age)
        {
```

```
                case (16):
                    System.out.println("You are under 18.");
                    break;
                case (18):
                    System.out.println("You are eligible for vote.");
                    break;
                case (65):
                    System.out.println("You are senior citizen.");
                    break;
                default:
                    System.out.println("Please give the valid age.");
                    break;
            }
        }
    }
```

50. ----------

```
public static void main(String[] args) {
   String name = "Mango";
   switch(name){
   case "Mango":
      System.out.println("It is a fruit");
      break;
```

```
    case "Tomato":
       System.out.println("It is a vegitable");
       break;
    case "Coke":
       System.out.println("It is cold drink");
    }
}
}
```

----------LOOPS- ----------

51. For loop

```
....//Java Program to demonstrate the example of for loop
//which prints table of 1
public class ForExample {
public static void main(String[] args) {
   //Code of Java for loop
   for(int i=1;i<=10;i++){
      System.out.println(i);
   }
}
```

```
}
```

52. Nested for loop

```
... public class NestedForExample {
public static void main(String[] args) {
//loop of i
for(int i=1;i<=3;i++){
//loop of j
for(int j=1;j<=3;j++){
      System.out.println(i+" "+j);
}//end of i
}//end of j
}
}
```

53. Pyramid

```
..... public class PyramidExample {
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();//new line
}
}
}
```

54. pyramid-1

```
….. public class PyramidExample2 {
public static void main(String[] args) {
int term=6;
for(int i=1;i<=term;i++){
for(int j=term;j>=i;j--){
      System.out.print("* ");
}
System.out.println();//new line
}
}
}
```

55. For each loop

```
…… //Java For-each loop example which prints the
//elements of the array
public class ForEachExample {
public static void main(String[] args) {
   //Declaring an array
   int arr[]={12,23,44,56,78};
   //Printing array using for-each loop
   for(int i:arr){
      System.out.println(i);
   }
}
}
```

56. Java Labeled For Loop

.....

```
//A Java program to demonstrate the use of labeled for loop
public class LabeledForExample {
public static void main(String[] args) {
   //Using Label for outer and for loop
   aa:
      for(int i=1;i<=3;i++){
         bb:
            for(int j=1;j<=3;j++){
               if(i==2&&j==2){
                  break aa;
               }
               System.out.println(i+" "+j);
            }
      }
}
}
```

57. If you use break bb;, it will break inner loop only which is the default behavior of any loop.

....

```
public class LabeledForExample2 {
public static void main(String[] args) {
```

```
    aa:
      for(int i=1;i<=3;i++){
        bb:
          for(int j=1;j<=3;j++){
            if(i==2&&j==2){
              break bb;
            }
            System.out.println(i+" "+j);
          }
      }
}
}
```

58. Java Infinitive For Loop

…..

```
//Java program to demonstrate the use of infinite for loop
//which prints an statement
public class ForExample {
public static void main(String[] args) {
   //Using no condition in for loop
   for(;;){
      System.out.println("infinitive loop");
   }
}
}
```

59. Java While Loop

```
….. public class WhileExample {
public static void main(String[] args) {
   int i=1;
   while(i<=10){
      System.out.println(i);
   i++;
   }
}
}
```

60. Java Infinitive While Loop

```
…. public class WhileExample2 {
public static void main(String[] args) {
   while(true){
      System.out.println("infinitive while loop");
   }
}
}
```

61. Java do-while Loop

```
….
public class DoWhileExample {
public static void main(String[] args) {
   int i=1;
```

```
    do{
       System.out.println(i);
    i++;
    }while(i<=10);
  }
  }
```

62. Java Infinitive do-while Loop

```
  …
  public class DoWhileExample2 {
  public static void main(String[] args) {
    do{
       System.out.println("infinitive do while loop");
    }while(true);
  }
  }
```

63. string…….

```
  public class StringDemo
  {
    public static void main(String args[]) {
      char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.' };
      String helloString = new String(helloArray);
      System.out.println( helloString );
    }
  }
```

64. String length……..

```
public class StringDemo {


  public static void main(String args[]) {
    String palindrome = "Dot saw I was Tod";
    int len = palindrome.length();
    System.out.println( "String Length is : " + len );
  }
}
```

65. Concat of string

```
public class StringDemo
{
  public static void main(String args[]) {
    String string1 = "saw I was ";
    System.out.println("Dot " + string1 + "Tod");
  }
}
```

66. String format

```
System.out.printf("The value of the float variable is " +
          "%f, while the value of the integer " +
          "variable is %d, and the string " +
          "is %s", floatVar, intVar, stringVar);

String fs;
```

```
fs = String.format("The value of the float variable is " +
                   "%f, while the value of the integer " +
                   "variable is %d, and the string " +
                   "is %s", floatVar, intVar, stringVar);
System.out.println(fs);
```

```
67.   String txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
System.out.println("The length of the txt string is: " +
txt.length());
```

```
68.   String txt = "Hello World";
System.out.println(txt.toUpperCase());   // Outputs "HELLO WORLD"
System.out.println(txt.toLowerCase());   // Outputs "hello world"
```

```
69.   String txt = "Please locate where 'locate' occurs!";
System.out.println(txt.indexOf("locate")); // Outputs 7
```

```
70.   String firstName = "John";
String lastName = "Doe";
System.out.println(firstName + " " + lastName);
```

```
71.   ……..
1.  public class TestStringFormatter {
2.  public static void main(String[] args) {
```

```
3.      System.out.println(StringFormatter.reverseString("my name is khan"));
4.      System.out.println(StringFormatter.reverseString("I am sonoo jaiswal"));
5.      }
6.  }
```

72. .....

```
1.  public class StringFormatter {
2.  public static String reverseString(String str){
3.      StringBuilder sb=new StringBuilder(str);
4.      sb.reverse();
5.      return sb.toString();
6.  }
7.  }
```

73. Reverse of a string

```
public class Reverse
{
   public static void main(String[] args) {
      String string = "Dream big";
      //Stores the reverse of given string
      String reversedStr = "";

      //Iterate through the string from last and add each character to variable
reversedStr
      for(int i = string.length()-1; i >= 0; i--){
```

```
            reversedStr = reversedStr + string.charAt(i);
        }

        System.out.println("Original string: " + string);
        //Displays the reverse of given string
        System.out.println("Reverse of given string: " + reversedStr);
    }
}
```

74. N equal parts

```
public class DivideString {
    public static void main(String[] args) {
         String str = "aaaabbbbcccc";
        //Stores the length of the string
        int len = str.length();
        //n determines the variable that divide the string in 'n' equal parts
        int n = 3;
        int temp = 0, chars = len/n;
        //Stores the array of string
        String[] equalStr = new String [n];
        //Check whether a string can be divided into n equal parts
        if(len % n != 0) {
            System.out.println("Sorry this string cannot be divided into "+ n +" equal
parts.");
        }
```

```
            else {
               for(int i = 0; i < len; i = i+chars) {
                  //Dividing string in n equal part using substring()
                  String part = str.substring(i, i+chars);
                  equalStr[temp] = part;
                  temp++;
               }
            System.out.println(n + " equal parts of given string are ");
               for(int i = 0; i < equalStr.length; i++) {
                  System.out.println(equalStr[i]);
                  }
               }
            }
         }
```

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
int x = myNumbers[1][2];
      System.out.println(x);
```

75. ---------Methods----------

```
public class ExampleMinNumber {
```

```
  public static void main(String[] args) {
    int a = 11;
    int b = 6;
    int c = minFunction(a, b);
    System.out.println("Minimum Value = " + c);
  }
  /** returns the minimum of two numbers */
  public static int minFunction(int n1, int n2) {
    int min;
    if (n1 > n2)
      min = n2;
    else
      min = n1;
    return min;
  }
}
```

76. -----classess----------

```
//Java Program to illustrate how to define a class and fields
//Defining a Student class.
```

```
class Student{
 //defining fields
 int id;//field or data member or instance variable
 String name;
 //creating main method inside the Student class
 public static void main(String args[]){
  //Creating an object or instance
  Student s1=new Student();//creating an object of Student
  //Printing values of the object
  System.out.println(s1.id);//accessing member through reference variable
  System.out.println(s1.name);
 }
}
```

77. Object and Class Example: main outside the class

```
//Java Program to demonstrate having the main method in
//another class
//Creating Student class.
class Student{
 int id;
 String name;
}
```

```
//Creating another class TestStudent1 which contains the main method
class TestStudent1{
 public static void main(String args[]){
  Student s1=new Student();
  System.out.println(s1.id);
  System.out.println(s1.name);
 }
}
```

78. Object and Class Example: Initialization through reference

Initializing an object means storing data into the object. Let's see a simple example where we are going to initialize the object through a reference variable.

File: TestStudent2.java

```
class Student{
 int id;
 String name;
}
class TestStudent2{
 public static void main(String args[]){
  Student s1=new Student();
  s1.id=101;
  s1.name="Sonoo";
  System.out.println(s1.id+" "+s1.name);//printing members with a white space
 }
```

```
}
```

79. File: TestStudent3.java

```
class Student{
 int id;
 String name;
}
class TestStudent3{
 public static void main(String args[]){
  //Creating objects
  Student s1=new Student();
  Student s2=new Student();
  //Initializing objects
  s1.id=101;
  s1.name="Sonoo";
  s2.id=102;
  s2.name="Amit";
  //Printing data
  System.out.println(s1.id+" "+s1.name);
  System.out.println(s2.id+" "+s2.name);
 }
}
```

80. Object and Class Example: Initialization through method

In this example, we are creating the two objects of Student class and initializing the value to these objects by invoking the insertRecord method. Here, we are displaying the state (data) of the objects by invoking the displayInformation() method.

File: TestStudent4.java

```
class Student{
 int rollno;
 String name;
 void insertRecord(int r, String n){
  rollno=r;
  name=n;
 }
 void displayInformation(){
System.out.println(rollno+" "+name);}
}
class TestStudent4{
 public static void main(String args[]){
  Student s1=new Student();
  Student s2=new Student();
  s1.insertRecord(111,"Karan");
  s2.insertRecord(222,"Aryan");
  s1.displayInformation();
  s2.displayInformation();
 }
}
```

81. Object and Class Example: Initialization through a constructor

```
class Employee{
    int id;
    String name;
```

```
    float salary;
    Employee(int i, String n, float s) {
        id=i;
        name=n;
        salary=s;
    }
    void display(){System.out.println(id+" "+name+" "+salary);}
}
public class TestEmployee {
public static void main(String[] args) {
    Employee e1=new Employee(101,"ajeet",45000);
    Employee e2=new Employee(102,"irfan",25000);
    Employee e3=new Employee (103,"nakul",55000);
        e1.display();
    e2.display();
    e3.display();
} }
```

82.

File: TestRectangle1.java

```
class Rectangle{
 int length;
 int width;
 void insert(int l, int w){
  length=l;
```

```
  width=w;
 }
 void calculateArea(){System.out.println(length*width);}
}
class TestRectangle1{
 public static void main(String args[]){
  Rectangle r1=new Rectangle();
  Rectangle r2=new Rectangle();
  r1.insert(11,5);
  r2.insert(3,15);
  r1.calculateArea();
  r2.calculateArea();
}
}
```

83. example of an anonymous object in Java.

```
class Calculation{
 void fact(int  n){
  int fact=1;
  for(int i=1;i<=n;i++){
   fact=fact*i;
  }
 System.out.println("factorial is "+fact);
}
public static void main(String args[]){
```

```
 new Calculation().fact(5);//calling method with anonymous object
}
}
```

84.

```
//Java Program to illustrate the use of Rectangle class which
//has length and width data members
class Rectangle{
 int length;
 int width;
 void insert(int l,int w){
  length=l;
  width=w;
 }
 void calculateArea(){System.out.println(length*width);}
}
class TestRectangle2{
 public static void main(String args[]){
  Rectangle r1=new Rectangle(),r2=new Rectangle();//creating two objects
  r1.insert(11,5);
  r2.insert(3,15);
  r1.calculateArea();
  r2.calculateArea();
}
}
```

85. Real World Example: Account

File: TestAccount.java

```
//Java Program to demonstrate the working of a banking-system

//where we deposit and withdraw amount from our account.

//Creating an Account class which has deposit() and withdraw() methods

class Account{

int acc_no;

String name;

float amount;

//Method to initialize object

void insert(int a,String n,float amt){

acc_no=a;

name=n;

amount=amt;

}

//deposit method

void deposit(float amt){

amount=amount+amt;

System.out.println(amt+" deposited");

}
```

```
//withdraw method
void withdraw(float amt){
if(amount<amt){
System.out.println("Insufficient Balance");
}else{
amount=amount-amt;
System.out.println(amt+" withdrawn");
}
}
//method to check the balance of the account
void checkBalance(){System.out.println("Balance is: "+amount);}
//method to display the values of an object
void display(){System.out.println(acc_no+" "+name+" "+amount);}
}
//Creating a test class to deposit and withdraw amount
class TestAccount{
public static void main(String[] args){
Account a1=new Account();
a1.insert(832345,"Ankit",1000);
a1.display();
a1.checkBalance();
```

```
a1.deposit(40000);

a1.checkBalance();

a1.withdraw(15000);

a1.checkBalance();

}}
```

----------Constructors in Java---------

86. Example of default constructor

In this example, we are creating the no-arg constructor in the Bike class. It will be invoked at the time of object creation.

```
1. //Java Program to create and call a default constructor
2. class Bike1{
3. //creating a default constructor
4. Bike1(){System.out.println("Bike is created");}
5. //main method
6. public static void main(String args[]){
7. //calling a default constructor
8. Bike1 b=new Bike1();
9. }
10.}
```

87. Example of default constructor that displays the default values

//Let us see another example of default constructor

//which displays the default values

class Student3{

```
int id;
String name;
//method to display the value of id and name
void display(){System.out.println(id+" "+name);}

public static void main(String args[]){
//creating objects
Student3 s1=new Student3();
Student3 s2=new Student3();
//displaying values of the object
s1.display();
s2.display();
}
}
```

88. //Java Program to demonstrate the use of the parameterized constructor.

```
class Student4{
    int id;
    String name;
    //creating a parameterized constructor
    Student4(int i,String n){
    id = i;
```

```
    name = n;
    }
    //method to display the values
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    //creating objects and passing values
    Student4 s1 = new Student4(111,"Karan");
    Student4 s2 = new Student4(222,"Aryan");
    //calling method to display the values of object
    s1.display();
    s2.display();
   }
}
```

89. Example of Constructor Overloading

```
1. //Java program to overload constructors
2. class Student5{
3.     int id;
4.     String name;
5.     int age;
6.     //creating two arg constructor
7.     Student5(int i,String n){
```

```
8.      id = i;
9.      name = n;
10.     }
11.     //creating three arg constructor
12.     Student5(int i,String n,int a){
13.     id = i;
14.     name = n;
15.     age=a;
16.     }
17.     void display(){System.out.println(id+" "+name+" "+age);}
18.
19.     public static void main(String args[]){
20.     Student5 s1 = new Student5(111,"Karan");
21.     Student5 s2 = new Student5(222,"Aryan",25);
22.     s1.display();
23.     s2.display();
24.    }
25.}
```

90. //Java program to initialize the values from one object to another object.

```
1.  class Student6{
2.      int id;
3.      String name;
4.      //constructor to initialize integer and string
5.      Student6(int i,String n){
6.      id = i;
7.      name = n;
8.      }
9.      //constructor to initialize another object
```

```
10.    Student6(Student6 s){
11.    id = s.id;
12.    name =s.name;
13.    }
14.    void display(){System.out.println(id+" "+name);}
15.
16.    public static void main(String args[]){
17.    Student6 s1 = new Student6(111,"Karan");
18.    Student6 s2 = new Student6(s1);
19.    s1.display();
20.    s2.display();
21.   }
22.}
```

91. Copying values without constructor

We can copy the values of one object into another by assigning the objects values to another object. In this case, there is no need to create the constructor.

```
1.  class Student7{
2.     int id;
3.     String name;
4.     Student7(int i,String n){
5.     id = i;
6.     name = n;
7.     }
8.     Student7(){}
9.     void display(){System.out.println(id+" "+name);}
10.
11.    public static void main(String args[]){
```

```
12.    Student7 s1 = new Student7(111,"Karan");
13.    Student7 s2 = new Student7();
14.    s2.id=s1.id;
15.    s2.name=s1.name;
16.    s1.display();
17.    s2.display();
18.   }
19.}
```

92. Example of static variable

```
1.  //Java Program to demonstrate the use of static variable
2.  class Student{
3.     int rollno;//instance variable
4.     String name;
5.     static String college ="ITS";//static variable
6.     //constructor
7.     Student(int r, String n){
8.     rollno = r;
9.     name = n;
10.   }
11.   //method to display the values
12.   void display (){System.out.println(rollno+" "+name+" "+college);}
13.}
14. //Test class to show the values of objects
15. public class TestStaticVariable1{
16.  public static void main(String args[]){
17. Student s1 = new Student(111,"Karan");
18. Student s2 = new Student(222,"Aryan");
```

```
19. //we can change the college of all objects by the single line of code
20. //Student.college="BBDIT";
21. s1.display();
22. s2.display();
23. }
24.}
```

93. Program of the counter without static variable

```
1. //Java Program to demonstrate the use of an instance variable
2. //which get memory each time when we create an object of the class.
3. class Counter{
4. zsw21`
5. }
6.
7. public static void main(String args[]){
8. //Creating objects
9. Counter c1=new Counter();
10. Counter c2=new Counter();
11. Counter c3=new Counter();
12.}
13.}
```

94. Program of counter by static variable

```
1. //Java Program to illustrate the use of static variable which
2. //is shared with all objects.
3. class Counter2{
4. static int count=0;//will get memory only once and retain its value
5.
6. Counter2(){
```

```
7.  count++;//incrementing the value of static variable
8.  System.out.println(count);
9.  }
10.
11. public static void main(String args[]){
12. //creating objects
13. Counter2 c1=new Counter2();
14. Counter2 c2=new Counter2();
15. Counter2 c3=new Counter2();
16. }
17. }
```

95. Example of static method

```
1.  //Java Program to demonstrate the use of a static method.
2.  class Student{
3.      int rollno;
4.      String name;
5.      static String college = "ITS";
6.      //static method to change the value of static variable
7.      static void change(){
8.      college = "BBDIT";
9.      }
10.     //constructor to initialize the variable
11.     Student(int r, String n){
12.     rollno = r;
13.     name = n;
14.     }
15.     //method to display values
```

```
16.     void display(){System.out.println(rollno+" "+name+" "+college);}
17.}
18.//Test class to create and display the values of object
19.public class TestStaticMethod{
20.    public static void main(String args[]){
21.    Student.change();//calling change method
22.    //creating objects
23.    Student s1 = new Student(111,"Karan");
24.    Student s2 = new Student(222,"Aryan");
25.    Student s3 = new Student(333,"Sonoo");
26.    //calling display method
27.    s1.display();
28.    s2.display();
29.    s3.display();
30.    }
31.}
```

96. //Java Program to get the cube of a given number using the static method

```
1.
2.  class Calculate{
3.    static int cube(int x){
4.    return x*x*x;
5.    }
6.
7.    public static void main(String args[]){
8.    int result=Calculate.cube(5);
9.    System.out.println(result);
10.  }
11.}
```

97. Example of static block

```
1. class A2{
2.   static{System.out.println("static block is invoked");}
3.   public static void main(String args[]){
4.    System.out.println("Hello main");
5.   }
6. }
```

98. class A3{

```
1.   static{
2.   System.out.println("static block is invoked");
3.   System.exit(0);
4.   }
5. }
```

99. understand the problem if we don't use this keyword by the example given below:

```
1. class Student{
2. int rollno;
3. String name;
4. float fee;
5. Student(int rollno,String name,float fee){
6. rollno=rollno;
7. name=name;
8. fee=fee;
9. }
10. void display(){System.out.println(rollno+" "+name+" "+fee);}
11. }
12. class TestThis1{
```

```
13. public static void main(String args[]){
14. Student s1=new Student(111,"ankit",5000f);
15. Student s2=new Student(112,"sumit",6000f);
16. s1.display();
17. s2.display();
18. }}
```

100. Solution of the above problem by this keyword

```
1.  class Student{
2.  int rollno;
3.  String name;
4.  float fee;
5.  Student(int rollno,String name,float fee){
6.  this.rollno=rollno;
7.  this.name=name;
8.  this.fee=fee;
9.  }
10. void display(){System.out.println(rollno+" "+name+" "+fee);}
11. }
12.
13. class TestThis2{
14. public static void main(String args[]){
15. Student s1=new Student(111,"ankit",5000f);
16. Student s2=new Student(112,"sumit",6000f);
17. s1.display();
18. s2.display();
19. }}
```

101. Program where this keyword is not required

```
class Student{
int rollno;
String name;
float fee;
Student(int r,String n,float f){
rollno=r;
name=n;
fee=f;
}
void display(){System.out.println(rollno+" "+name+" "+fee);}
}

class TestThis3{
public static void main(String args[]){
Student s1=new Student(111,"ankit",5000f);
Student s2=new Student(112,"sumit",6000f);
s1.display();
s2.display();
}}
```

102. this: to invoke current class method

```
class A{
void m(){System.out.println("hello m");}
void n(){
System.out.println("hello n");
//m();//same as this.m()
this.m();
}
}
```

```
9. class TestThis4{
10. public static void main(String args[]){
11. A a=new A();
12. a.n();
13. }}
```

103. this() : to invoke current class constructor

Calling default constructor from parameterized constructor:

```
1. class A{
2. A(){System.out.println("hello a");}
3. A(int x){
4. this();
5. System.out.println(x);
6. }
7. }
8. class TestThis5{
9. public static void main(String args[]){
10. A a=new A(10);
11. }}
```

104. Calling parameterized constructor from default constructor:

```
1. class A{
2. A(){
3. this(5);
4. System.out.println("hello a");
5. }
6. A(int x){
7. System.out.println(x);
```

```
8.  }
9.  }
10. class TestThis6{
11. public static void main(String args[]){
12. A a=new A();
13. }}
```

105. Real usage of this() constructor call

The this() constructor call should be used to reuse the constructor from the constructor. It maintains the chain between the constructors i.e. it is used for constructor chaining. Let's see the example given below that displays the actual use of this keyword.

```
1.  class Student{
2.  int rollno;
3.  String name,course;
4.  float fee;
5.  Student(int rollno,String name,String course){
6.  this.rollno=rollno;
7.  this.name=name;
8.  this.course=course;
9.  }
10. Student(int rollno,String name,String course,float fee){
11. this(rollno,name,course);//reusing constructor
12. this.fee=fee;
13. }
14. void display(){System.out.println(rollno+" "+name+" "+course+" "+fee);}
15. }
16. class TestThis7{
17. public static void main(String args[]){
```

```
18. Student s1=new Student(111,"ankit","java");
19. Student s2=new Student(112,"sumit","java",6000f);
20. s1.display();
21. s2.display();
22. }}
```

106. .....

```
1.  class Student{
2.  int rollno;
3.  String name,course;
4.  float fee;
5.  Student(int rollno,String name,String course){
6.  this.rollno=rollno;
7.  this.name=name;
8.  this.course=course;
9.  }
10. Student(int rollno,String name,String course,float fee){
11. this.fee=fee;
12. this(rollno,name,course);//C.T.Error
13. }
14. void display(){System.out.println(rollno+" "+name+" "+course+" "+fee);}
15. }
16. class TestThis8{
17. public static void main(String args[]){
18. Student s1=new Student(111,"ankit","java");
19. Student s2=new Student(112,"sumit","java",6000f);
20. s1.display();
21. s2.display();
22. }}
```

107. this: to pass as an argument in the method

```
1. class S2{
2.   void m(S2 obj){
3.   System.out.println("method is invoked");
4.   }
5.   void p(){
6.   m(this);
7.   }
8.   public static void main(String args[]){
9.   S2 s1 = new S2();
10.  s1.p();
11.  }
12.}
```

108. this: to pass as argument in the constructor call

```
1. class B{
2.   A4 obj;
3.   B(A4 obj){
4.     this.obj=obj;
5.   }
6.   void display(){
7.     System.out.println(obj.data);//using data member of A4 class
8.   }
9.  }
10.
11.class A4{
12.  int data=10;
13.  A4(){
```

```
14.   B b=new B(this);
15.   b.display();
16.  }
17.  public static void main(String args[]){
18.   A4 a=new A4();
19.  }
20.}
```

109. this keyword can be used to return current class instance

Example of this keyword that you return as a statement from the method

```
1.  class A{
2.  A getA(){
3.  return this;
4.  }
5.  void msg(){System.out.println("Hello java");}
6.  }
7.  class Test1{
8.  public static void main(String args[]){
9.  new A().getA().msg();
10.}
11.}
```

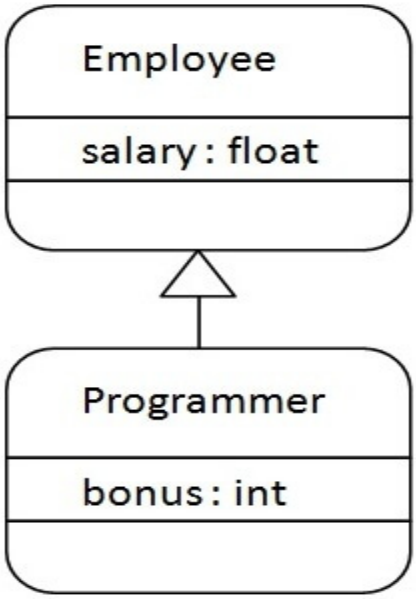
110. Proving this keyword

Let's prove that this keyword refers to the current class instance variable. In this program, we are printing the reference variable and this, output of both variables are same.

```
1. class A5{
2. void m(){
3. System.out.println(this);//prints same reference ID
4. }
5. public static void main(String args[]){
6. A5 obj=new A5();
7. System.out.println(obj);//prints the reference ID
8. obj.m();
9. }
10.}
```

----------Inheritance----------

111. …




```
1. class Employee{
2.  float salary=40000;
3. }
4. class Programmer extends Employee{
5.  int bonus=10000;
6.  public static void main(String args[]){
7.    Programmer p=new Programmer();
8.    System.out.println("Programmer salary
   is:"+p.salary);
9.    System.out.println("Bonus of Programmer
   is:"+p.bonus);
10.}
11.}
12.
```

112. Single Inheritance Example

```
1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
5. void bark(){System.out.println("barking...");}
6. }
7. class TestInheritance{
8. public static void main(String args[]){
9. Dog d=new Dog();
10. d.bark();
11. d.eat();
12. }}
```

113. Multilevel Inheritance Example

```
1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
5. void bark(){System.out.println("barking...");}
6. }
7. class BabyDog extends Dog{
8. void weep(){System.out.println("weeping...");}
9. }
10. class TestInheritance2{
11. public static void main(String args[]){
12. BabyDog d=new BabyDog();
13. d.weep();
```

```
14. d.bark();
15. d.eat();
16. }}
```

114. Hierarchical Inheritance Example

```
1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
5. void bark(){System.out.println("barking...");}
6. }
7. class Cat extends Animal{
8. void meow(){System.out.println("meowing...");}
9. }
10. class TestInheritance3{
11. public static void main(String args[]){
12. Cat c=new Cat();
13. c.meow();
14. c.eat();
15. //c.bark();//C.T.Error
16. }}
```

115. multiple inheritance (not supported by java - compile time error)

```
1. class A{
2. void msg(){System.out.println("Hello");}
3. }
4. class B{
```

```
5.  void msg(){System.out.println("Welcome");}
6.  }
7.  class C extends A,B{//suppose if it were
8.
9.   public static void main(String args[]){
10.   C obj=new C();
11.   obj.msg();//Now which msg() method would be invoked?
12.}
13.}
```

116. Example of Aggregation

```
1.  class Operation{
2.   int square(int n){
3.    return n*n;
4.   }
5.  }
6.
7.  class Circle{
8.   Operation op;//aggregation
9.   double pi=3.14;
10.
11. double area(int radius){
12.   op=new Operation();
13.   int rsquare=op.square(radius);//code reusability (i.e. delegates the method
      call).
14.   return pi*rsquare;
15. }
16.
17.
```

```
18.
19. public static void main(String args[]){
20.   Circle c=new Circle();
21.   double result=c.area(5);
22.   System.out.println(result);
23. }
24.}
```

117. Address.java

```
1.  public class Address {
2.  String city,state,country;
3.
4.  public Address(String city, String state, String country) {
5.      this.city = city;
6.      this.state = state;
7.      this.country = country;
8.  }
9.
10.}
```

Emp.java

```
1.  public class Emp {
2.  int id;
3.  String name;
4.  Address address;
5.
6.  public Emp(int id, String name,Address address) {
7.      this.id = id;
8.      this.name = name;
```

```
9.      this.address=address;
10.}
11.
12.void display(){
13.System.out.println(id+" "+name);
14.System.out.println(address.city+" "+address.state+" "+address.country);
15.}
16.
17.public static void main(String[] args) {
18.Address address1=new Address("gzb","UP","india");
19.Address address2=new Address("gno","UP","india");
20.
21.Emp e=new Emp(111,"varun",address1);
22.Emp e2=new Emp(112,"arun",address2);
23.
24.e.display();
25.e2.display();
26.
27.}
28.}
```

118. …..

```
/* File name : Employee.java */
public class Employee {
   private String name;
   private String address;
   private int number;
```

```
public Employee(String name, String address, int number) {
  System.out.println("Constructing an Employee");
  this.name = name;
  this.address = address;
  this.number = number;
}

public void mailCheck() {
  System.out.println("Mailing a check to " + this.name + " " + this.address);
}

public String toString() {
  return name + " " + address + " " + number;
}

public String getName() {
  return name;
}

public String getAddress() {
  return address;
}

public void setAddress(String newAddress) {
  address = newAddress;
```

```
  }

  public int getNumber() {
    return number;
  }
}

/* File name : Salary.java */
public class Salary extends Employee {
  private double salary; // Annual salary

  public Salary(String name, String address, int number, double salary) {
    super(name, address, number);
    setSalary(salary);
  }

  public void mailCheck() {
    System.out.println("Within mailCheck of Salary class ");
    System.out.println("Mailing check to " + getName()
    + " with salary " + salary);
  }

  public double getSalary() {
    return salary;
  }
```

```
   public void setSalary(double newSalary) {
      if(newSalary >= 0.0) {
         salary = newSalary;
      }
   }

   public double computePay() {
      System.out.println("Computing salary pay for " + getName());
      return salary/52;
   }
}

/* File name : VirtualDemo.java */
public class VirtualDemo {

   public static void main(String [] args) {
      Salary s = new Salary("Mohd Mohtashim", "Ambehta, UP", 3, 3600.00);
      Employee e = new Salary("John Adams", "Boston, MA", 2, 2400.00);
      System.out.println("Call mailCheck using Salary reference --");
      s.mailCheck();
      System.out.println("\n Call mailCheck using Employee reference--");
      e.mailCheck();
   }
}
```

119. Simple example of Covariant Return Type

FileName: B1.java

```
1.class A{
2.A get()
3.{
4.return this;
5.}
6.}
7.
8.class B1 extends A{
9.@Override
10.B1 get()
11.{
12.return this;
13.}
14.void message(){System.out.println("welcome to covariant return type");}
15.
16.public static void main(String args[]){
```

```
17.new B1().get().message();
18.}
}
```

120.
```
class A1
{
    A1 foo()
    {
        return this;
    }

    void print()
    {
        System.out.println("Inside the class A1");
    }
}


// A2 is the child class of A1
class A2 extends A1
{
    @Override
    A1 foo()
    {
        return this;
    }

```

```
23.    void print()
24.    {
25.        System.out.println("Inside the class A2");
26.    }
27.}
28.
29.// A3 is the child class of A2
30.class A3 extends A2
31.{
32.    @Override
33.    A1 foo()
34.    {
35.        return this;
36.    }
37.
38.    @Override
39.    void print()
40.    {
41.        System.out.println("Inside the class A3");
42.    }
43.}
44.
45.public class CovariantExample
46.{
47.    // main method
48.    public static void main(String argvs[])
49.    {
50.      A1 a1 = new A1();
51.
```

```
52.      // this is ok
53.      a1.foo().print();
54.
55.      A2 a2 = new A2();
56.
57.      // we need to do the type casting to make it
58.      // more clear to reader about the kind of object created
59.      ((A2)a2.foo()).print();
60.
61.      A3 a3 = new A3();
62.
63.      // doing the type casting
64.      ((A3)a3.foo()).print();
65.
66.   }
67.}
```

121. **FileName:** CovariantExample.java

```
1.  class A1
2.  {
3.     A1 foo()
4.     {
5.        return this;
6.     }
7.
8.     void print()
9.     {
10.       System.out.println("Inside the class A1");
11.    }
```

```
}


// A2 is the child class of A1
class A2 extends A1
{
    @Override
    A2 foo()
    {
        return this;
    }

    void print()
    {
        System.out.println("Inside the class A2");
    }
}

// A3 is the child class of A2
class A3 extends A2
{
    @Override
    A3 foo()
    {
        return this;
    }

    @Override
    void print()
```

```
41.    {
42.        System.out.println("Inside the class A3");
43.    }
44.}
45.
46.public class CovariantExample
47.{
48.    // main method
49.    public static void main(String argvs[])
50.    {
51.        A1 a1 = new A1();
52.
53.        a1.foo().print();
54.
55.        A2 a2 = new A2();
56.
57.        a2.foo().print();
58.
59.        A3 a3 = new A3();
60.
61.        a3.foo().print();
62.
63.    }
64.}
```

------file- handlings-----

122. CreateFile.java

```
1. // Importing File class
2. import java.io.File;
3. // Importing the IOException class for handling errors
4. import java.io.IOException;
5.  class CreateFile {
6.             public static void main(String args[]) {
7.             try {
8.                 // Creating an object of a file
9.                 File f0 = new File("D:FileOperationExample.txt");
10.                if (f0.createNewFile()) {
11.                     System.out.println("File " + f0.getName() + " is created
    successfully.");
12.                } else {
13.                     System.out.println("File is already exist in the directory.");
14.                }
15.              } catch (IOException exception) {
16.                   System.out.println("An unexpected error is occurred.");
17.                   exception.printStackTrace();
18.             }
19.      }
20.}
```

123. FileInfo.java

```
1. // Import the File class
2. import java.io.File;
```

```
class FileInfo {
    public static void main(String[] args) {
        // Creating file object
        File f0 = new File("D:FileOperationExample.txt");
        if (f0.exists()) {
            // Getting file name
            System.out.println("The name of the file is: " + f0.getName());

            // Getting path of the file
            System.out.println("The absolute path of the file is: " +
    f0.getAbsolutePath());

            // Checking whether the file is writable or not
            System.out.println("Is file writeable?: " + f0.canWrite());

            // Checking whether the file is readable or not
            System.out.println("Is file readable " + f0.canRead());

            // Getting the length of the file in bytes
            System.out.println("The size of the file in bytes is: " + f0.length());
        } else {
            System.out.println("The file does not exist.");
        }
    }
}
```

124. WriteToFile.java

```
// Importing the FileWriter class
import java.io.FileWriter;
```

```
3.
4.  // Importing the IOException class for handling errors
5.  import java.io.IOException;
6.
7.  class WriteToFile {
8.     public static void main(String[] args) {
9.
10.    try {
11.       FileWriter fwrite = new FileWriter("D:FileOperationExample.txt");
12.       // writing the content into the FileOperationExample.txt file
13.       fwrite.write("A named location used to store related information is referred
      to as a File.");
14.
15.       // Closing the stream
16.       fwrite.close();
17.       System.out.println("Content is successfully wrote to the file.");
18.    } catch (IOException e) {
19.       System.out.println("Unexpected error occurred");
20.       e.printStackTrace();
21.       }
22.    }
23.}
```

125. **ReadFromFile.java**

```
1.  // Importing the File class
2.  import java.io.File;
3.  // Importing FileNotFoundException class for handling errors
4.  import java.io.FileNotFoundException;
5.  // Importing the Scanner class for reading text files
```

```
6.  import java.util.Scanner;
7.
8.  class ReadFromFile {
9.     public static void main(String[] args) {
10.       try {
11.          // Create f1 object of the file to read data
12.          File f1 = new File("D:FileOperationExample.txt");
13.          Scanner dataReader = new Scanner(f1);
14.          while (dataReader.hasNextLine()) {
15.             String fileData = dataReader.nextLine();
16.             System.out.println(fileData);
17.          }
18.          dataReader.close();
19.       } catch (FileNotFoundException exception) {
20.          System.out.println("Unexcpected error occurred!");
21.          exception.printStackTrace();
22.       }
23.   }
24.}
```

126. DeleteFile.java

```
1.  // Importing the File class
2.  import java.io.File;
3.  class DeleteFile {
4.    public static void main(String[] args) {
5.     File f0 = new File("D:FileOperationExample.txt");
6.     if (f0.delete()) {
7.       System.out.println(f0.getName()+ " file is deleted successfully.");
8.     } else {
```

```
9.          System.out.println("Unexpected error found in deletion of the file.");
10.    }
11.  }
12.}
```