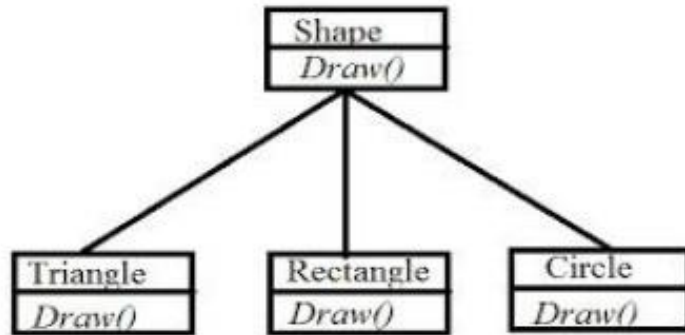


Polymorphism:

Polymorphism is the ability of an object (methods) to take on many forms.

Generally it occurs when we have many classes that are related to each other by inheritance.



In Java, we use method overloading and method overriding to achieve polymorphism.

Differences between OOP and Procedure oriented programming

Basis For comparison

POP

OOP

Basic	Procedure/Structure oriented.	Object oriented.
Approach	Top-down.	Bottom-up.
Basis	Main focus is on "how to get the task done" i.e. on the procedure or structure of a program .	Main focus is on 'data security'. Hence, only objects are permitted to access the entities of a class.
Division	Large program is divided into units called functions.	Entire program is divided into objects.
Entity accessing mode	No access specifier observed.	Access specifier are "private, default, protected & public".
Overloading/Polymorphism	Neither it overload functions nor operators.	It overloads functions, constructors.
Inheritance	There is no provision of inheritance.	Supports Inheritance.
Data hiding & security	There is no proper way of hiding the data, so data is insecure	Data is hidden with private, default & protected. hence data security increases.

variables:

Variables are containers for storing data values.

The value depends on the data type of the variable.

The value can be altered during program execution.

There are three types of variables in java: local, instance and static.

Syntax: datatype variable_name[=default_value];

Example: int x=50;

Expressions

Any unit of code that can be evaluated to a value is an expression.

Example-1: 10+15

Example-2: (x*y)/z;

Type Casting:

Convert a value from one data type to another data type is known as type casting.

Widening Type Casting: Converting a lower data type into a higher one.

It is also known as implicit conversion or casting down. It is done automatically.

It is safe because there is no chance of loss of data.

It takes place when the target type must be larger than the source type.

Example:

```
int x = 7;
```

```
//automatically converts the integer type into long type
```

```
long y = x;
```

Narrowing Type Casting: Converting a higher data type into a lower one.

It is also known as explicit conversion or casting up.

It is done manually by the programmer.

If we do not perform casting then the compiler reports a compile-time error.

Example:

```
double d = 166.66;
```

```
int i = (int) d;           //converting double data type into int data type
```

Example:

```
int a[]=new int[5];  
int a[]={};
```

Multidimensional Array in Java

In such case, data is stored in row and column based index (also known as matrix form).

Syntax to Declare Multidimensional Array in Java

```
dataType[][] arrayRefVar; (or)  
dataType [][]arrayRefVar; (or)  
dataType arrayRefVar[][]; (or)  
dataType []arrayRefVar[];
```

Example to instantiate Multidimensional Array in Java

```
int[][] arr = new int[3][3];
```

Example-1:

```
class Test  
{  
    public static void main(String arg[])  
    {  
        int arr[][] = { {1,2,3}, {2,4,5}, {4,4,5} };  
        for(int i=0;i<3;i++)  
        {  
            for(int j=0;j<3;j++)  
            {  
                System.out.print(arr[i][j]+" ");  
            }  
            System.out.println();  
        }  
    }  
}
```

```

        System.out.println("Sum of matrices:-");
        for ( i= 0 ; i < row ; i++ )
        {
            for ( j= 0 ; j < col ;j++ )
                System.out.print(res[i][j]+"\\t");
            System.out.println();
        }
    }
}

```

Class Example / Working with multiple objects

```

class Student
{
    private int sno;
    private String sname;
    public void init(int x, String y)
    {
        sno=x;
        sname=y;
    }
    public void display()
    {
        System.out.println(sno);
        System.out.println(sname);
    }
}

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

```

Preview from Notesale.co.uk
Page 19 of 42

```

{
    Student s[]=new Student[5];
    s[0]=new student();
    s[0].init(1,"xxx");
    s[0].display();
    s[1]=new student();
    s[1].init(2,"yyy");
    s[1].display();
    s[2]=new student();
    s[2].init(3,"zzz");
    s[2].display();
}
}

```

Method Overloading / Runtime Polymorphism / Compile time Polymorphism / Early Binding / Static Binding:

Compile Time Polymorphism: Linking of function call with function definition can be resolved at compile time itself.

Method Overloading: Defining multiple methods with the same name is called method overloading.

We can overload two methods as follows

1. By varying no. of arguments.
2. By varying data type of the arguments.
3. By varying order of data types.

Note: we can't overload two methods by varying only return types.

“this” key word

The keyword **‘this’** is a reference to the current object.

Example:

```
class Student
{
    private int sno;
    private String sname;
    public void init(int sno, String sname)
    {
        this.sno = sno;
        this.sname = sname;
    }
    public void display()
    {
        System.out.println(sno);
        System.out.println(sname);
    }
}

class Test
{
    public static void main(String args[])
    {
        Student s1 = new Student();
        s1.init(1,"shiva");
        s1.display();
    }
}
```

```
System.out.println(string.indexOf("al", 5)); // -1
```

6. contains():

returns true or false after matching the sequence of char value.

Example:

```
System.out.println("abc".contains("b")); // true
```

7. startsWith() and endsWith():

Example:

```
System.out.println("abc".startsWith("a")); // true
```

```
System.out.println("abc".startsWith("A")); // false
```

```
System.out.println("abc".endsWith("c")); // true
```

```
System.out.println("abc".endsWith("a")); // false
```

8. replace():

The replace() method does a simple search and replace on the string.

Example:

```
System.out.println("abcabc".replace('c', 'A')); // AbcAbc
```

```
System.out.println("abcabc".replace("a", "A")); // AbcAbc
```

9. toLowerCase(): returns a string in lowercase.

Example:

```
System.out.println("Abc123".toLowerCase()); // abc123
```

10. toUpperCase(): returns a string in uppercase.

Example:

```
String string = "animals";
```

```
System.out.println(string.toUpperCase()); // ANIMALS
```

11. trim(): removes whitespace from the beginning and end of a String.

Example:

```
System.out.println("abc".trim()); // abc
```

```
System.out.println("\t a b c\n".trim()); // a b c
```