oops practical — Practical No: 01

class and object

```
class complex
{
 int real;
 int imaginary;
  complex ()
 {}
 complex(int r, int i)
 {
  real = r;
  imaginary = i;
 }
 void printComplex()
 {
 system.out.println("complex No is"; + real + "+" +
                    imaginary + "i");
 }
}
 class Main complex
 {
  public static Void main(String args [])
 {
  complex c1 = new) complex (2,3);
 complex c2 = new complex (4,5);
  c1.printComplex();
 c2.printComplex();
 }
}
```

data

= constructor

⇒ method

output:

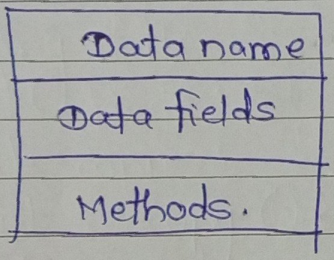
complex No is 2+3i

complex No is 4+5i

class

class: class is a collection of data and the functions that manipulate the data.

- The data components of the class are called data field and the
- The function components of the class are called member function or methods



- Encapsulation means binding of data and method together in single entity called class.

class is used to achieve an encapsulation property

The class that contains main function is called main class

object

- object is an instance of a class.
- object is are basic run-time in oop.
- A single class can create any number of objects.

Syntax for object

```
class_Name object_Name = new class_Name();
```

member variables
a member variable has a name and a type, and holds a new value of a particular type.

Default constructor:-
A constructor with no parameter is called as default constructor. It initializes the member variables to their default values.

practical no: 02

polymorphism

```
import java.util.*;
class publication
{
  string title;
  int price;
  int copies;
   void salecopy()
  { }
}
class Book
{
  string title;
  int price;
  int copies;
 void salecopy (){
   price =250;
  system.out.println(" total sale:"+copies * price);
}
 Void ordercopies(){
```

```
    scanner in = new scanner (system. in);
    system out print In (" Enter copies :");
     copies = in. next In ();
  }
}

 class Magazine
{
 string title;
 int price;
 int copies;
 Void salecopy (){}
 void current Issuee(){}
 Void recieve Issue (){}
}
public class pubs
 {
  public static Void main (string args [])
 {
  Book b= new Book();
   b. order copies ();
   b. sale copy ();
 }
}
```

output:

Enter copies: 5

Total scale: 1250

polymorphism

(~~inn~~ innumerable form)

polymorphism means (many structure)

- polymorphism is form of two different greek words poly and morphs.

- poly means → numerous
- morphs → forms

polymorphism is important features of object oriented programming.

polymorphism allows us to perform a single action in different ways.

polymorphism in java two different methods:
1. method overloading
2. method overridding

1. # method overloading:-

method overloading is the process in which we can create multiple methods of the same name in the same class, and all methods work in different ways.

method overloading occurs when there is more than one method of the same name in the class.

method overloading occurs at compile time.

different number of parameters can be passed to the function.

It may have different return types.

2] method overridding

method overridding is the process in which the subclass or a child class has the same method as declared in parent class.

It may same return types

method overridding occurs at the run time

in method overridding the number of parameters are passed to the function are the same,

polymorphism are two types

1. compile time polymorphism
2. Run time polymorphism.

1) # compile time polymorphism

1. the polymorphism is resolved during compile time is known as compile time polymorphism.
2. method overloading is an example of compile time polymorphism.
3. The compile time polymorphism are also called as Static polymorphism

② Run time polymorphism:-

1) The polymorphism which is resolved during run time is called runtime polymorphism.

2) method overridding is an example of run time polymorphism

3) The run time polymorphism is also called as dynamic polymorphism

practical No: 03

Inheritance

Inheritance is a property by which the new classes are created using old classes.

Inheritance support hierarchical structure

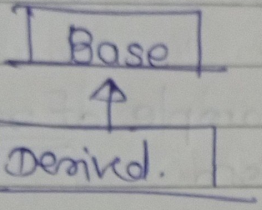
The old classes are refered as base classes and new classes are refered as derived classes

Inheritance may take different forms

① single inheritance
② multiple inheritance
③ Multilevel inheritance
④ Hybrid inheritance.

① single inheritance :- (only one super class)

In single inheritance there is one parent per derived class.

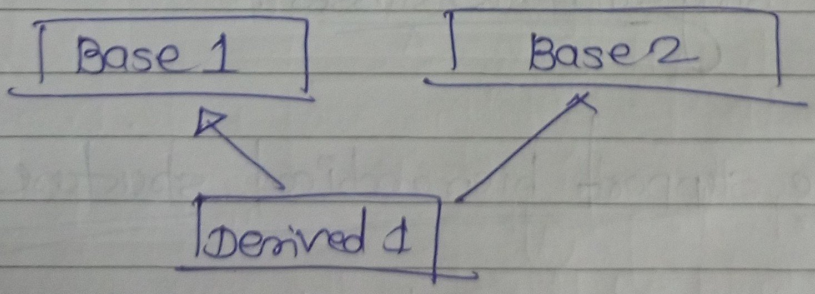


OR

The single inheritance in which the one parent are obtained from derived class.

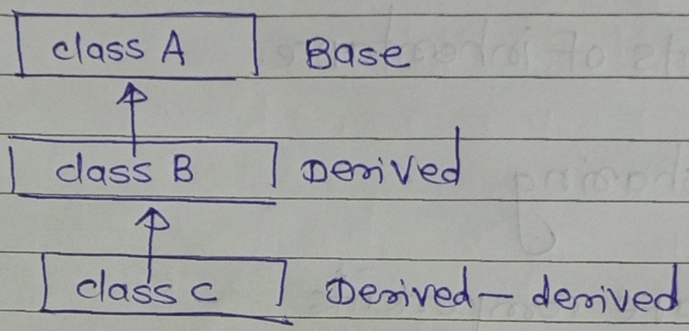
② multiple inheritance (several super class)

* In multiple inheritance the derived class is derived from more than one base class.



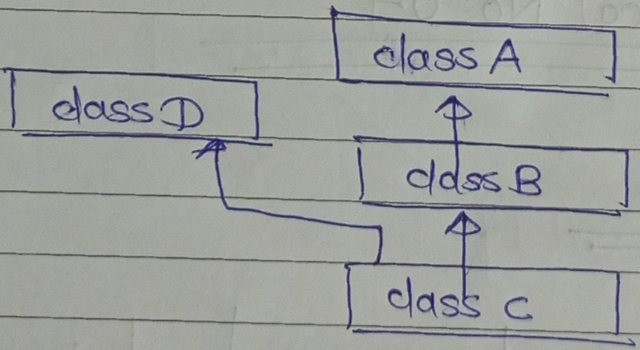
③ multilevel Inheritance

when a derived class is derived from a base class which itself is derived class then that type of inheritance is called multilevel inheritance.



④ # Hybrid Inheritance:

when two or more types of inheritance are combined together then it forms the hybrid Inheritance.



⑤ # Hirerarchical inheritance:-

Deriving many subclass from one super class is known as hiererachical ~~str~~ inheritance.

Need of inheritance:-

1. It helps in reduced code
2. It enhances readability of code.
3. Execution of code is efficient

Benefits of inheritance:

1. code sharing
2. software components
3. Rapid prototyping
4. Information Hiding
5. polymorphism and frameworks.

practical No: 04

Interface

Interface is defined is an abstract type to specify the behaviour of class

The interface may contain data members and methods but the methods are not defined.

The interface makes use of only public access specifier.

The members of interfaces are always declared as final.

syntax of interface

```
access interface name
{
return_type method_name 1
return_type method_name 2
type final_varname 1 = value;
type final_varname 2 = value;
//....
return type method_nameN(parameter_list);
type final_varNameN = value;
}
```

Implementating interface

It is necessary to create a class for every interface

The class must be defined in the following form while using interface.

```
class class_name extends superclass_name
implements interface_name1, interface_name2...
{
  // body of class
}
```

practical NO: 05 (Exception)

exception :- Exception is an abnormal condition.

exception is an abnormal condition that can occur in the program.

The condition is caused due to run time error in the program.

Error

When any kind of serious problem occurs which could not be handled easily like out of memory Error then an error is problem.

Types of errors

① Compile time error :-
② Run time error :-

① Compile time errors :-

compile time errors occur when there are syntactical issues present in application code example missing semicolons or parentheses.

wrong spelling of keywords and identifiers.

missing brackets of classes and methods.

missing double quotes in the string

(2) Run time errors

- Runtime errors occur when sometimes goes wrong in the normal execution of a program.

example

1. converting invalid string to numbers.
2. Accessing the array element which is out of the index.
3. In an expression, divide by zero

Advantages of Exception handling

1. Easy Identification of program code & error handling code
2. Identify error type
3. propagation of errors.

Hierarchy of Java exception

Hierarch inherited by two subclasses.

1. Exception
2. error.

types of exception
1. checked exception
2. unchecked exception
3. Error.

① checked exception:

The classes that does not inherit the runtime exception are classed checked exceptions.

example: IOException, SQLException.

checked at compile time

② unchecked exception:

The classes that inherit the runtime are known as unchecked exception.
example:
a) Arithmetic Exception
b) Null pointer Exception

③ error:
Error is irrecoverable.

example:-
Out of Memory error
Assertion Error etc.