OOPS: Object Oriented Programing System (Structure)

There are following objective of OOPS:

* Works on real world entity(object) concept
* Support to modular programing, large code divided in smaller set
* Reusability of source code
* Easy to maintain the source code

There are following concept of OOPS:

* **Object :** is a real world entity or is an instance of class
* **Class :** is wrapper of data member (member variable or attribute), and methods or functions : blueprint of object is called class
* **Encapsulation , and Data Hiding :** is wrapping of data member and method in single unit , or write the data member and function together and hide from outside
* Constructor : is function which invokes automatically when object will create

: Class name and function is same, i.e. function is called constructor

: Constructor cannot have return type

: Constructor cannot invoke explicitly

: There are following of constructor: i. default ii. With parameter iii. Copy constructor

* Inheritance
* Polymorphism
* Abstraction

Other supportive concepts:

* Coupling
* Cohesion
* Association
* Aggregation

Scope Resolution:

* Access Specifier
  + private
  + public
  + protected
  + default (no default) : default is work like public for same package

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Keywords | Within class | In same package | In child class | Anywhere |
| Private | Yes | No | No | no |
| Protected | yes | yes | Yes | no |
| Default | Yes | Yes | No | No |
| Public | Yes | Yes | Yes | Yes |

* Access Modifier
  + this
  + final
  + static
  + super

Inheritance: to extend one class features to another class

* we can access parent class data member and function from child class object
* parent class data member and function can be access directly in child class using super keyword or without super if data member and function is unique

Syntax:

Public class A{

int a;

}

Public class B extends A{

Int a;

Void test(){

This.a =1;

Super.a =44;

}

}

There are following types of inheritance

1. single level : A -> B
2. multi-level : A-> B ->C … -> X
3. tree / hierarchical : A->B,C,D … -> M,N
4. hybrid : multiple inheritance in same project (any two or more than two types inheritance in same project) : A -> B->C , X-> D,F
5. multiple : java dosen’t support multiple inheritance however this can be achieved using interface : A,B,C … -> M

Polymorphism : poly means and morphism means forms

Multiple function have same name but different works

There are following types of polymorphism

1. overloading : multiple function will have same name but different arguments (type of argument , count of argument) / overloading is also known as compile time or static overloading
2. overriding : multiple function will have same name and same argument / overriding is also known as runtime or dynamic

Abstract class : is partial class which contains abstract and non-abstract method

* abstract method cannot be implemented in abstract class
* non abstract method should be implemented in abstract class
* abstract class cannot initialized (object cannot be created of abstract class)

Interface: is fully abstract class, all methods are abstract by default

* to achieve the multiple inheritance
* interface cannot initialized (object cannot be created of interface)