**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 can 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. Create new object from previous object or Copy constructor

* Inheritance
* Polymorphism
* Abstraction

Other supportive concepts:

* Coupling
* Cohesion
* Association
* Aggregation

Scope Resolution:

* **Access Specifier** 
  + private
  + public
  + protected
  + default (no keyword) : 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 : is keyword to current object
  + final : is keyword which declare variable with read only access (constant)
    - * data should be assign during the declaration of variable
      * final function cannot be override
      * final class cannot be extended/inherit
  + static : - static data member and function can be access without object

- static allocates single memory

* + super

**Nest class / wrapper class : class inside class**