oops - Object oriented programming concepts.

Programming languages

_____

2 Types

---

1. Procedure Orienteered
    a. Mainly deals with methods / functions to solve problems.
    b. Top down approach is followed
    c. Pascal, C , Fortran
2. Object Oriented
    a. Focused on the structure of the data and functions/methods to handle the data
    b. Bottom up approach is followed
    c. Java, Dotnet (pure Object oriented Language)
    d. C++ (has the support for implementing oops)

Basic building blocks oops

1.class

- User defined data type
    - Data type – specifies the type and size of the data to be stored in a variable
    - Example
        - int Input → input variable can hold value of type integer of size 32 bit
        - Distance d1 ; → Distance - class, d1 - objectt
    - Distance d1 (it is not possible to declare d1 of the distance which has 2 parts (feet, inches))
    - Distance data type with 2 members(feet, inches) can be defined by using a class (user defined data type)
    - Class can have data members(feet, inches) and member functions(readDistance(), processDistance(), displayDistance()) to handle the data
    - Data members and member functions are wrapped into single unit is called class
    - Data Member → the attributes (feet and inches)which describe the entity (real time entity -- Distance)
    - Member functions → methods / functions (readDistance(), processDistance(), displayDistance()) which can be used to handle the data members

2. object

- Int = 10; --- it is not possible
- Int input = 10; // to hold the integer value , there is a need for a variable of the type int
- Similar to hold the value of distance , there is a need for aDistance object

- Distance d1; d1 – object which can hold a distance value

Features

_____
Reusability
Modularity

Properties

A → Abstraction
P →Polymorphism
I →Inheritance
E →Encapsulations

static / non static

static is common value for all the objects from the class (belongs to class)

- example min mark , max mark, pass mark are common for all students
- so it should be declared as static
- only one value (memory) is created.
- There is no need to create object to access the static members, it can be access by using either class name or object name but it keeps the same value.
    - Example Math.PI (Math class name and PI is static data member of Math class)

non static belongs to object

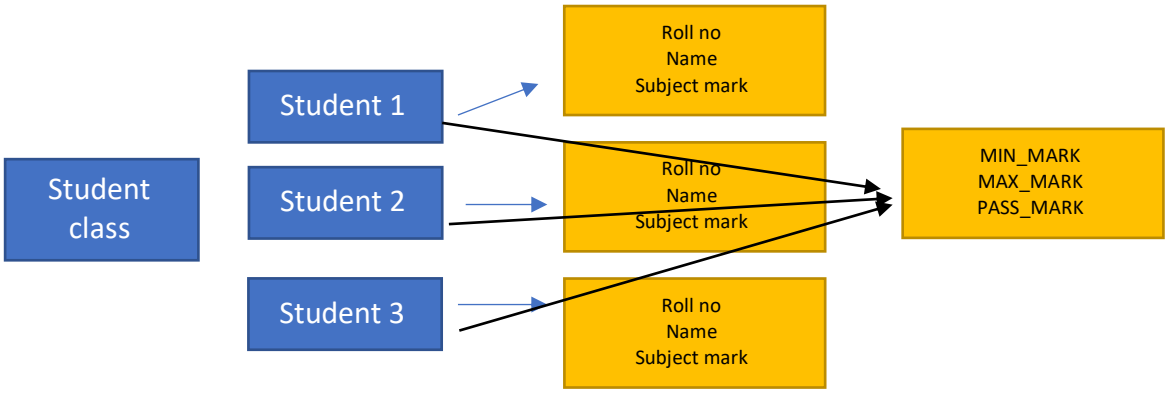
- example each and every student have their own subject marks, name and roll no
- and it should be declared as non-static
- each every object has a value (memory)

data member

a. Static data members
b. Non static data members

member functions

a. Static member functions
b. Non static member functions



Constructor

Constructor is a method
it should not have return type even void
it should have name same as to class name.
it can be used to initialize the data members.
It will be invoked when an object is created.
One class can have multiple constructor but it should have different parameters. The Parameters may be differed in type or nos or order or any one of combinations , overloaded constructor
If there is no constructor is defined, then default constructor(provided by JVM) will be invoked and all data members will be initialized by default value.
If any constructor is defined then the default constructor will be vanished.

this

- this refers the current object
- current object is nothing but, the object by in which the method is called.

Example time1.addTime(time2)

```
Public void addTime(time t){
        this.h = t.h + 10;
}
```

Now **this.h refers time1.h**

Inheritance

- A new class can be defined from an existing class
- New class is known as child class / derived class / sub class
- Existing class is known as parent class / super class / base class
- New class can have its own properties as well as the properties of parent class
- When there are common properties exists among different entities, that common properties must be declared in the base class
- Reusability
- Different type of inheritance
  - Simple Inhe

Super

Super keyword can be used to refer the base class.

Type of Inheritance

```
class A{

}
class B extends A{
        // simple inheritance
        // can have the properties of A
}
class C extends B{
                // multilevel inheritance
                // a new class can be derived from another sub class
                // can have the properties of B and A

}


class D {

}

class E extends A,D
{
        // multiple inheritance, Not supported in Java
        // more than one base class
}


// tree structure hybrid inheritance
```

http://www.net-informations.com/faq/oops/inheritancetype.htm#:~:text=%20Different%20Types%20of%20Inheritance%20%201%20Single,and%20multi%20level%29%20is%20called%20as...%20More%20

Polymorphism

- Means more than one form
- 3 ways of implementing polymorphism
    a. **Function /Method overloading**
        - More than one method can have same name with different parameters with in same class
        - Execution will be depends on the parameter passed
        - Early binding / compile time polymorphism
        - If the methods differs only in return type, gives compilation error

b. Function /Method overriding

- More than one method can have same method name, parameter and return type.
- One method definition will be in the base class another will be in the sub class.
- Execution will be depends on the object by in which the method is called. If it is called by base class object, base class method will be invoked. If is called by sub class object , sub class method will be invoked.
- Runtime polymorphism / Late binding

c. Operator over Loading

- Assigning more than one task to an Operator is called operator overloading
- + (val1 + val2 → used for adding 2 integers)
- "Welcome " + "You " → used for concatenation of 2 string
- Distance1 = Distance2 + Distance3 Distance1 = Distance.addDistance(Distance1,Distance2) (**Java doesn't support for operator over loading**)

Encapsulations

- Encapsulations is nothing but data hiding
  - By wrapping up of data members and member functions into a single unit, we can avoid the access of data members from outside the class.
  - By assigning different access modifiers public , private, protected and friendly.

packages

- Packages are nothing but similar to folder structure in operating system
- It is container for java class files and other related files
- Java has N number of build in packages - java.util.*, java.io.*, java.net.*
- Supports for creating used defined or custom packages
- package package_name;
- import package_name.class name;
- package should be the very first statement in the program

Access Modifiers

- **public, friendly, protected, private**
- **public** – can be accessed every where
- **friendly** – default, act as public in same package
- **protected** – act as public in same package, can be accessed in sub class of different package

- **private** – only with in the same class

Access Packages	Same package			Different Packages	
Access location	Same class	Sub class	Other class	Sub class	Other class
public	YES	YES	YES	YES	YES
protected	YES	YES	YES	YES	NO
friendly	YES	YES	YES	NO	NO
private	YES	NO	NO	NO	NO

Abstraction

- Express the essential things, not much in detail
- Abstraction is applicable in class and method level
- abstract class and abstract method

  abstract method

  a method without definition/body is called abstract method

  public abstract void method_name(int I,int j);

  abstract class

  - abstract class is class, which can have abstract methods, non-abstract methods, static and non-static data members.
  - Abstract class cannot have objects
  - Abstract class can be act as a base class, if abstract class is extended to a sub class, that sub class must override all the abstract methods in the abstract base class.
  - If class contains an abstract methods that class must be declared as an abstract class.

  abstract class class_name{
  data members
  static data members
  abstract methods ()
  non abstract methods()
  }

final

- final is just inverse to abstract
- applicable with class, member function, data member
- final class
  - can't be inherited
- final method
  - can't be override
- final data member
  - it is just like constant
  - can't be changed
  - must be initialized while declared

interface

- Interface is also just like abstract class
  - All member functions are abstract by default
  - All data members are static final by default
- Interface can be implemented in class (just like extending an abstract class)
  - While implementing an interface, the class must override all the abstract methods which are declared in all base interface
- Interface can’t have object
- Multiple inheritance can be implemented by using interface
  - That means we can have more than base interface
- interface can be extended from more than interface
  - it is not possible with class
  - that means a class can't have more than one base class
- class can be extended from a base class and implements any number of interface (Multiple Inheritance)