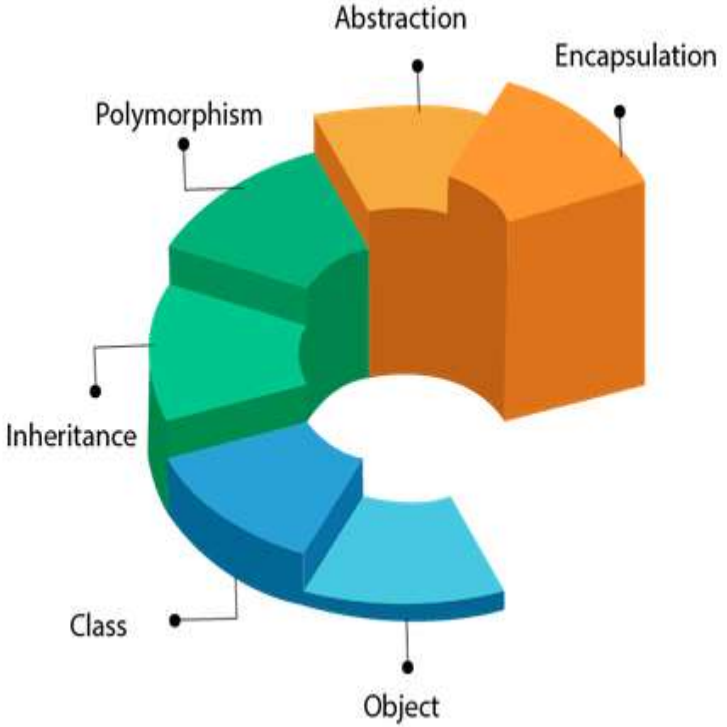

Object
Oriented
Programming

CONTENT

Object Oriented Programming

- The object-oriented paradigm is a programming methodology that promotes the efficient design and development of software systems using reusable components that can be quickly and safely assembled into larger systems.

- The main aim of object-oriented programming is to implement real-world concepts like
- Object → real world entity
- Classes → Templates/ Blueprints
- Abstraction → Visibility Controls
- Inheritance → Backward Compapatibilty , parent child relation
- Polymorphism →Many forms



Worst thing is that

Requirement

always change

WHY OOP

- Break down requirements into objects with responsibilities, not into functional steps
  Procedural approach → Object Oriented Approach
- Easier to Model Real things
- To make software projects more manageable and predictable.
- For more re-use code and prevent 'reinvention of wheel' every time.

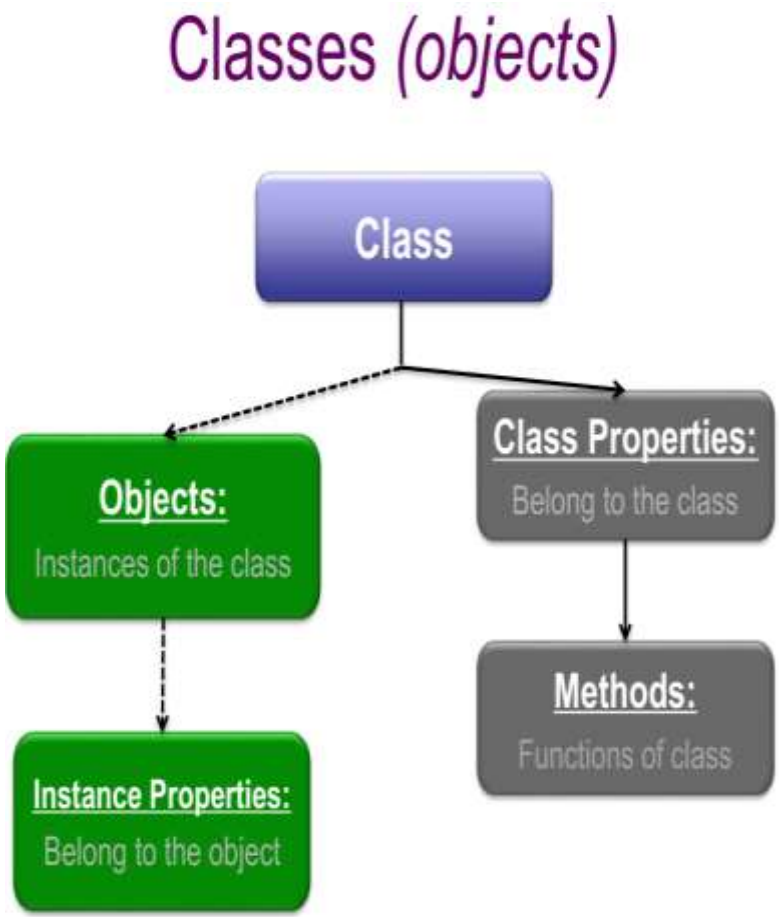
Components Of OOP

Class

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity.
-It is non primitive data type.
-It can't be physical(no memory space)
-Class members are access modifiers, objects , Methods , Instance variable and constructors.

Object

- An Object is an Instance of a class
Any entity that has state and behavior is known as an object.
For example a chair, pen, table, keyboard, bike, etc. It can be physical or logical.



Programming Representation Of a Class and Object

```
//Defining a Student class.
class Student{
 //defining fields
 int id;                //field or data member or instance variable
 String name;
 //creating main method inside the Student class
 public static void main(String args[]){
  //Creating an object or instance
  Student s1=new Student();//creating an object of Student
  //Printing values of the object
  System.out.println(s1.id);//accessing member through reference variable
  System.out.println(s1.name);
 }
}
```

Principles of OOPs

Inheritance:

- Inheritance is a mechanism in which one object acquires all the properties and behaviors of a parent object.
- Inheritance represents the **IS-A relationship** which is also known as a ***parent-child* relationship**.
- Like Animal is a Mammals , Reptiles or Birds.

➢ **Terms used in Inheritance**

- Sub Class/Child Class/drived/extended → inherits from other class
- Super Class/Parent Class/ Base class → Superclass is the class from where a subclass inherits the features.

- **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

➢ **Types Of inheritance:**

- **Single → Class B → Class A**
- **Multilevel → class C → Class B → class A**
- **Hierarchical → Class B → class A , Class C → class A**
- **Multiple → Class C → class A , class C → class B (not supported by java, ambiguity)**
- **Hybrid → class D → class B and C , Class B and C → Class A**

Programming Representation of Inheritance

- **The syntax of Java Inheritance**

```
class Subclass-name extends Superclass-name
{
   //methods and fields
}
```

Example:

```
class Employee{
 float salary=40000;
}
class Programmer extends Employee{
 int bonus=10000;
 public static void main(String args[]){
   Programmer p=new Programmer();
   System.out.println("Programmer salary is:"+p.salary);
   System.out.println("Bonus of Programmer is:"+p.bonus);
}
}
```

OUTPUT: Programmer salary is:40000.0 , Bonus of programmer is:10000

Principles of OOPs

Polymorphism

If *one task is performed by different ways*, it is known as polymorphism.

For example: To convince the customer differently, to draw something, like shape, triangle, rectangle , a cat speaks meow, dog barks woof, etc.

- Polymorphism present a method that can have many definitions. Polymorphism is related to
- Overloading → Compile time polymorphism/run time polymorphism
- Overriding → Run time polymorphism/ Dynamic polymorphism
- **Syntax**

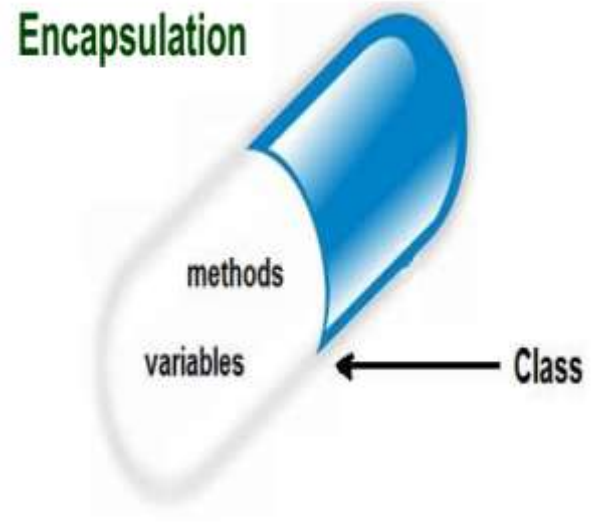
  getPrice()

  getPrice(string name)

Principles of OOP

Encapsulation

- Encapsulation is the integration of data and operations into a class.
  - Encapsulation is hiding the functional details from the object calling it.
- **Examples**
- A capsule
- **Can you drive the car?**
  - Yes, I can!
  - **So, how does acceleration work?**
  - Huh?
  - Details encapsulated (hidden) from the driver.



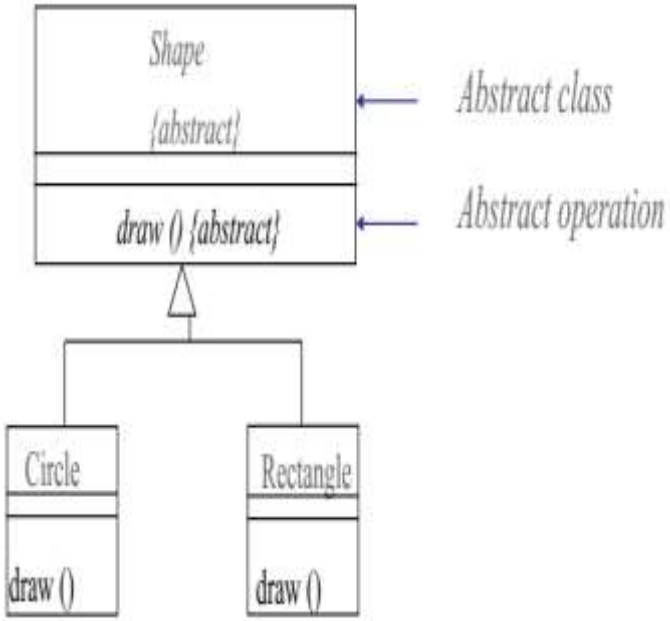
Principles of OOP

Abstraction

Abstraction is basically hiding the implementation and gain access to there functionality by exposing by extend keyword.

-An abstract class is a class that may not have any direct instances.

-An abstract operation is an operation that it is incomplete and requires a child to supply an implementation of the operation.



THANKS!