



# **SNS COLLEGE OF TECHNOLOGY**

**Coimbatore-35**  
**An Autonomous Institution**



Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade  
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

## **DEPARTMENT OF INFORMATION TECHNOLOGY**

### **OBJECT ORIENTED PROGRAMMING USING JAVA**

**I YEAR - II SEM**

#### **UNIT 1 – Introduction to Object Oriented Programming**

##### **TOPIC 1 – Introduction**



# Introduction



- Object Oriented Programming:
- Object-Oriented Programming System(OOPs) is a programming paradigm based on the concept of “objects” that contain **data** and **methods**.
- Object oriented programming brings together data and its behaviour(methods) in a single location(object) makes it easier to understand how a program works



## Why Object Oriented Programming?



- OOP language allows to break the program into the bit-sized problems that can be solved easily (one object at a time).
- The new technology promises greater programmer productivity, better quality of software and lesser maintenance cost.
- OOP systems can be easily upgraded from small to large systems.



# Introduction - JAVA

- Java is a Object oriented Programming Language
- Java is developed by James Gosling
- First released by Sun Microsystems in the early 1990s
- Formally it realized by oracle in 1995's

Java

	1970's C : structured programming	
	1980's C ++ : OOPS concept	
	1990's Java : platform independent	

James Gosling  
Father of Java



## Introduction - JAVA

- Java is one of the most popular programming languages used to create Web applications and platforms
- It was designed for flexibility, allowing developers to write code that would run on any machine, regardless of architecture or platform
- OOP concepts in Java are the main ideas behind Java's Object Oriented Programming.
- They are - abstraction, encapsulation, inheritance, and polymorphism



# OOP vs PP



## OBJECT ORIENTED PROGRAMMING

- Bottom Up approach
- Divided into objects
- Has *Access Modifiers*
- Objects can move & communicate with each other through member functions
- More secure
- Supports *overloading*

## PROCEDURAL PROGRAMMING

- Top Down Approach
- Divided into functions
- Doesn't have Access Modifiers
- Data can move freely from function to function in the system
- Less Secure
- Do not support overloading



# OOPs Concepts

Object-oriented programming has several advantages over procedural programming:

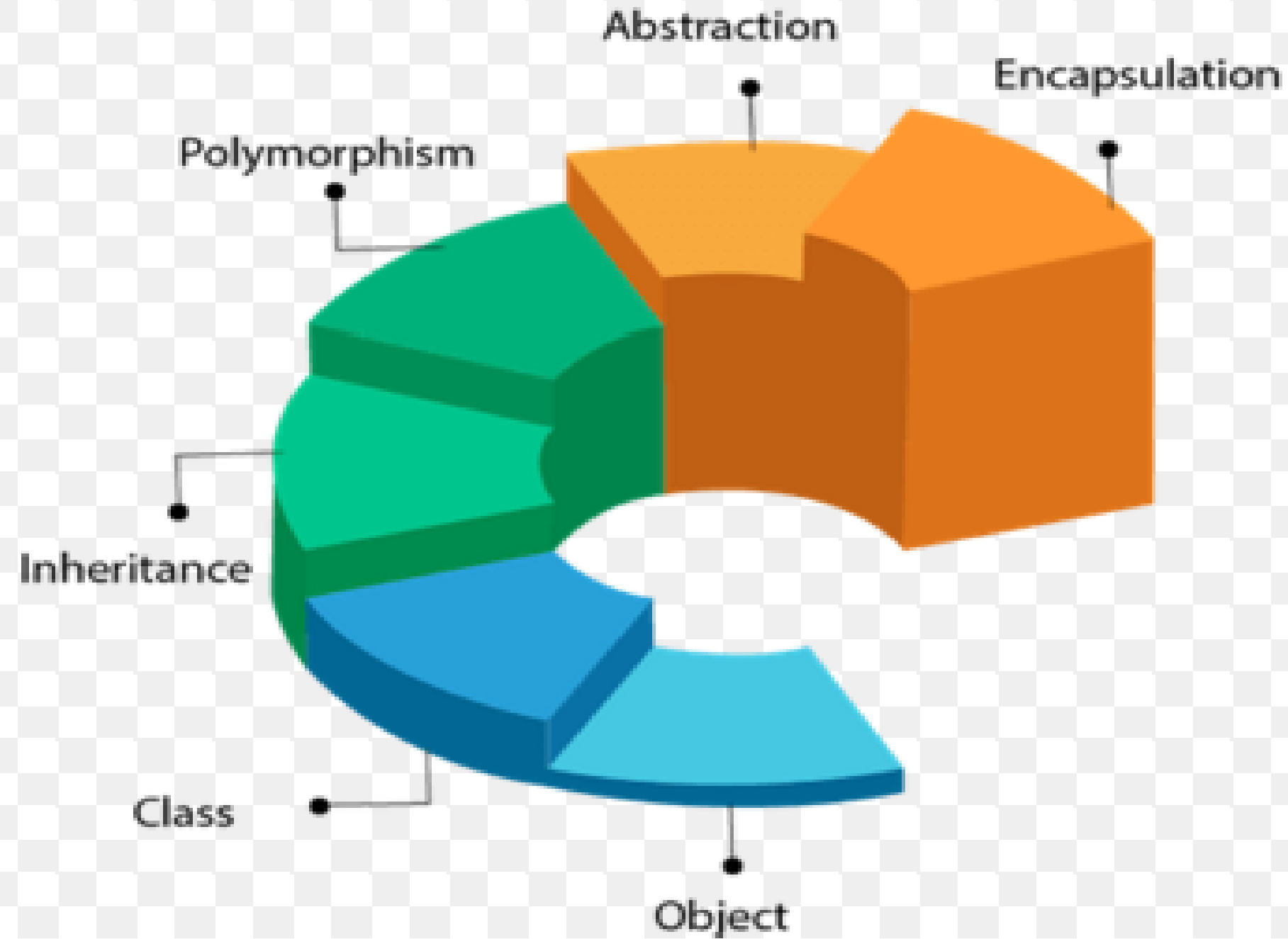
- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

## What are Classes and Objects?

Classes and objects are the two main aspects of object-oriented programming.



# Object Oriented Programming concepts







# Class

➤ A class is a group of common properties

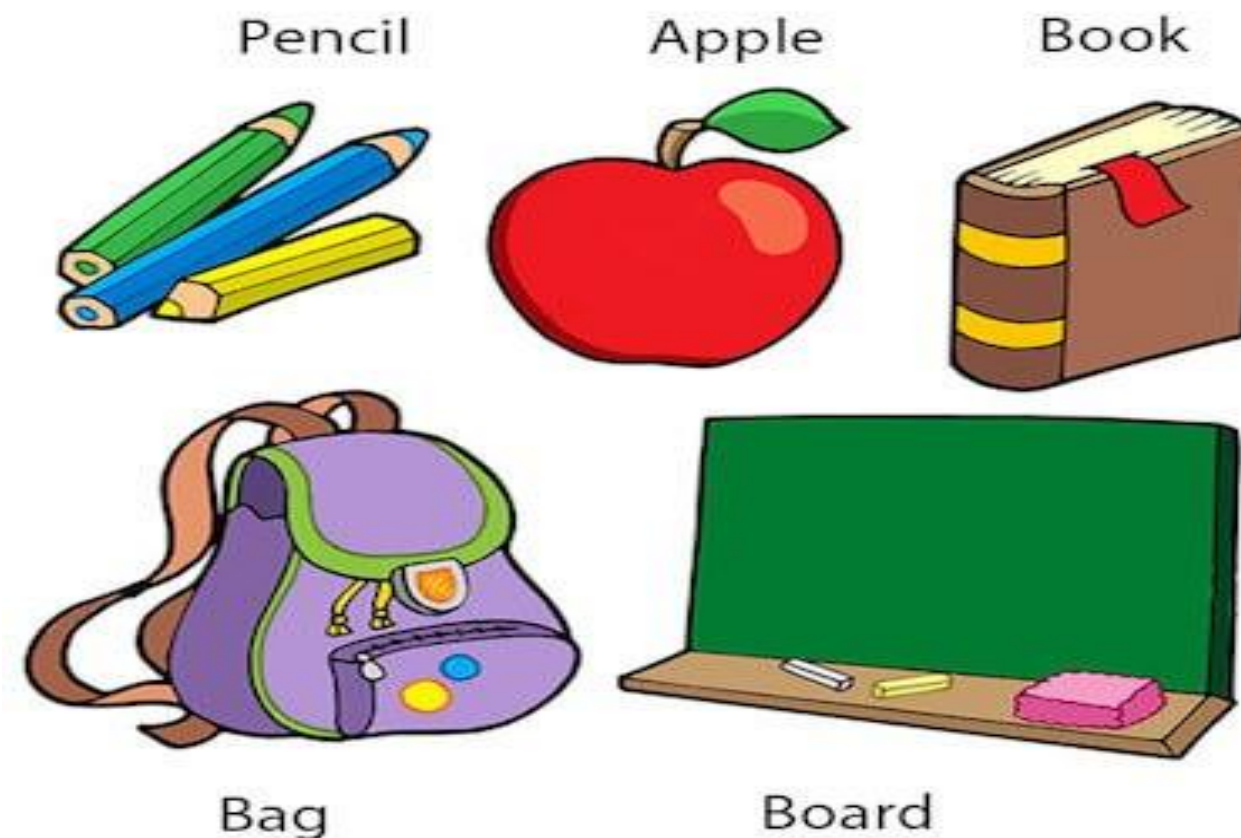




# Object

- An object is a real-world entity
- An object is a runtime entity
- The object is an entity which has state and behavior
- The object is an instance of a class

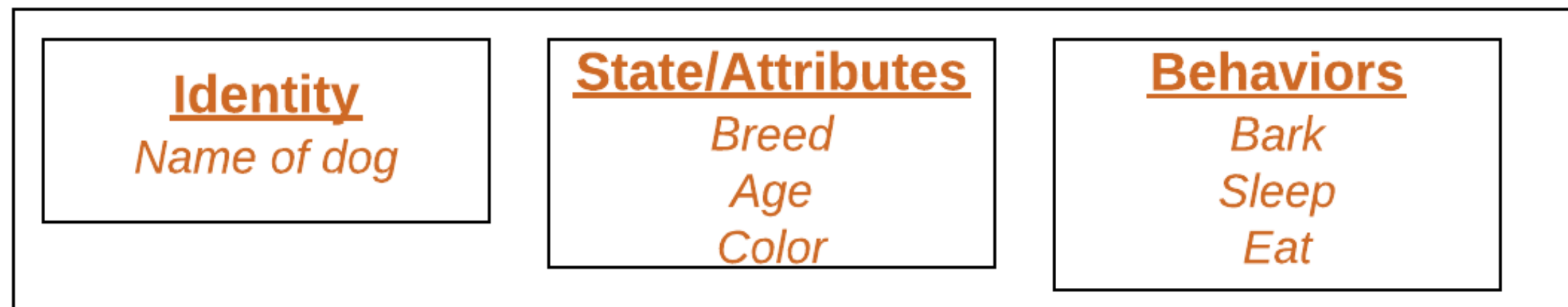
## Objects: Real World Examples





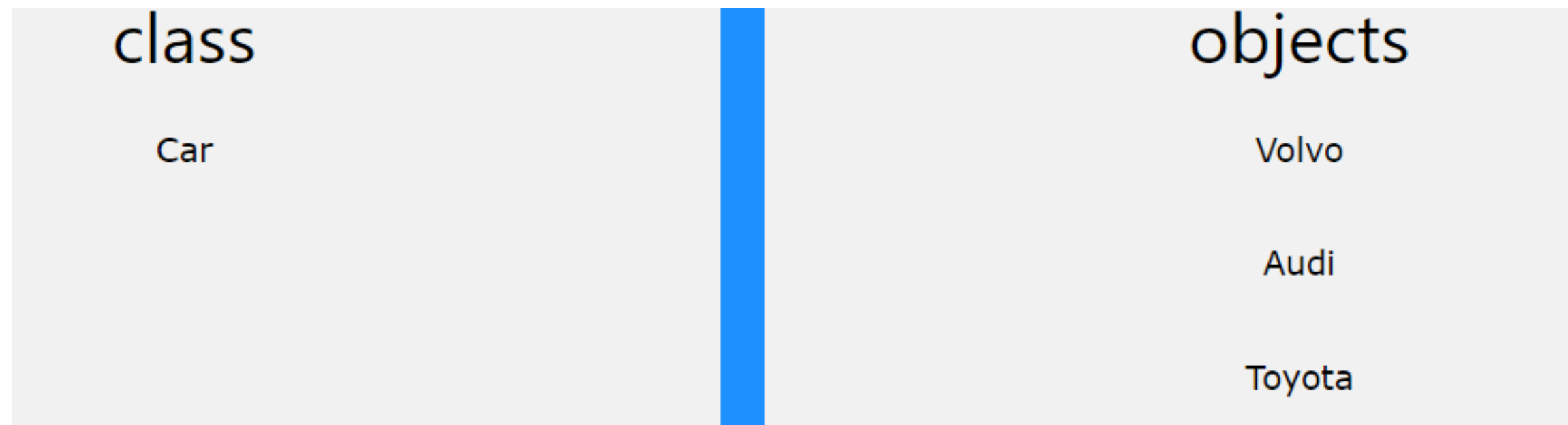
# Object

- Object consists of :
- State : It is represented by attributes of an object. It also reflects the properties of an object
- Behavior : It is represented by methods of an object. It also reflects the response of an object with other objects
- Identity : It gives a unique name to an object and enables one object to interact with other objects





# Object



So, a class is a template for objects, and an object is an instance of a class.

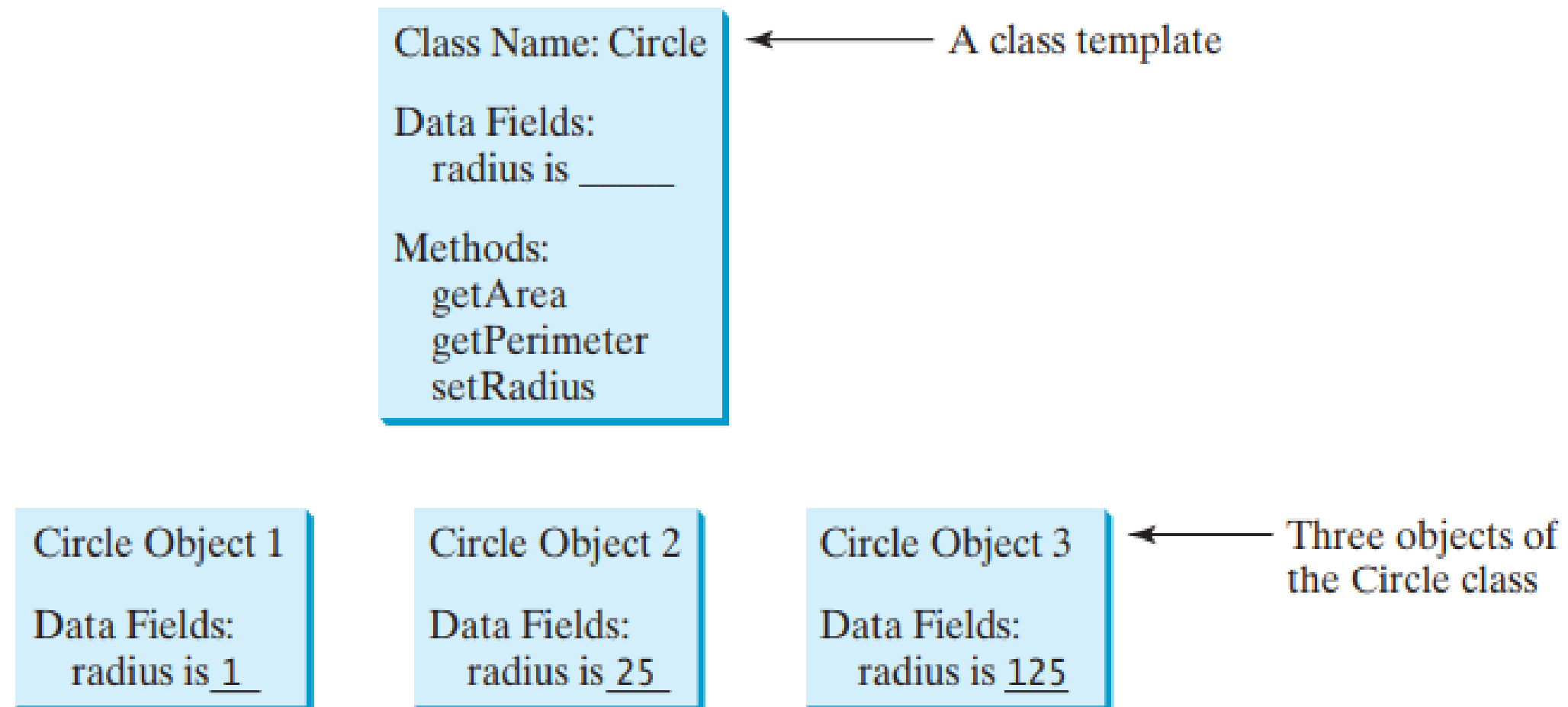
When the individual objects are created, they inherit all the variables and methods from the class.



# Object

## A class is a template for creating objects

Below diagram shows a **Circle** class which is a template to create three objects:





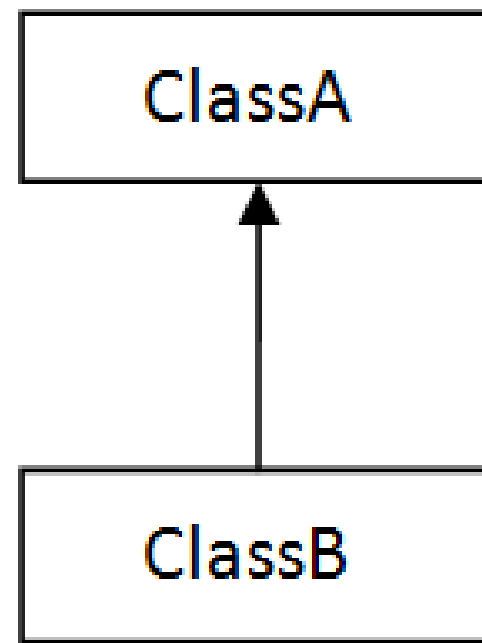
# Inheritance



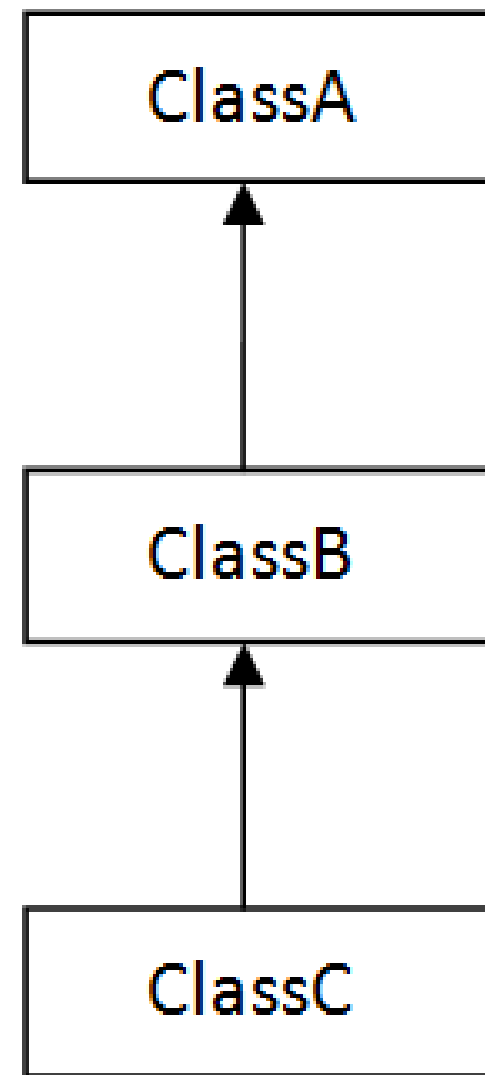
- Deriving a new class from existing class.
- Types of Inheritance:
  - Single
  - Multiple
  - Multilevel
  - Hierarchical
  - Hybrid



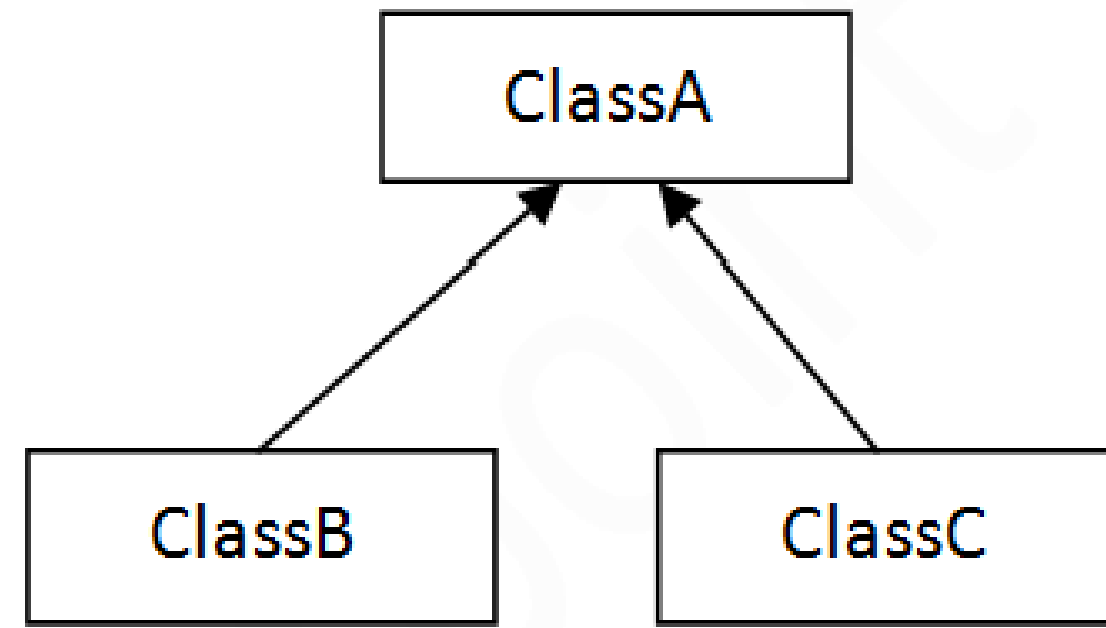
# Inheritance



1) Single



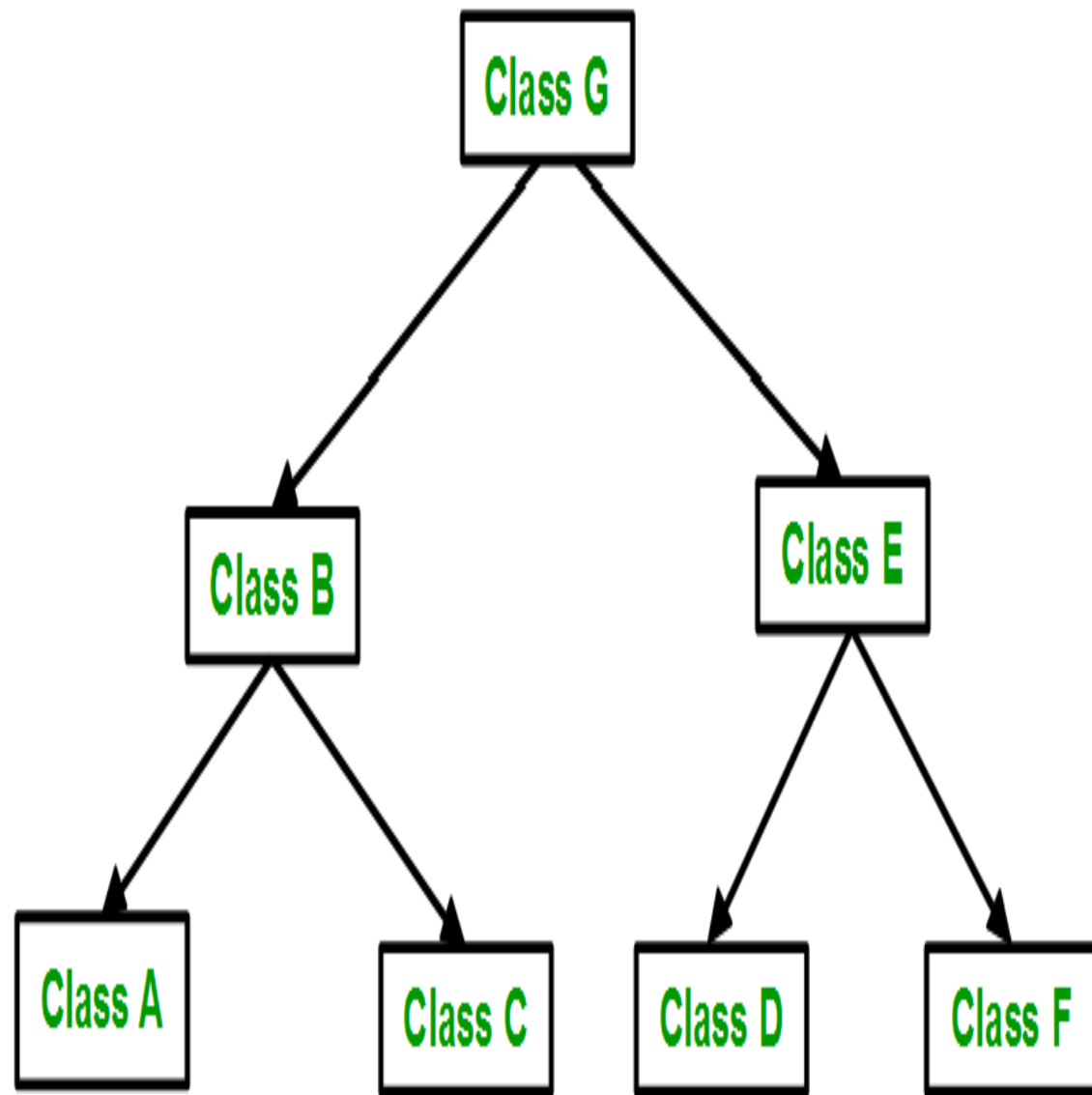
2) Multilevel



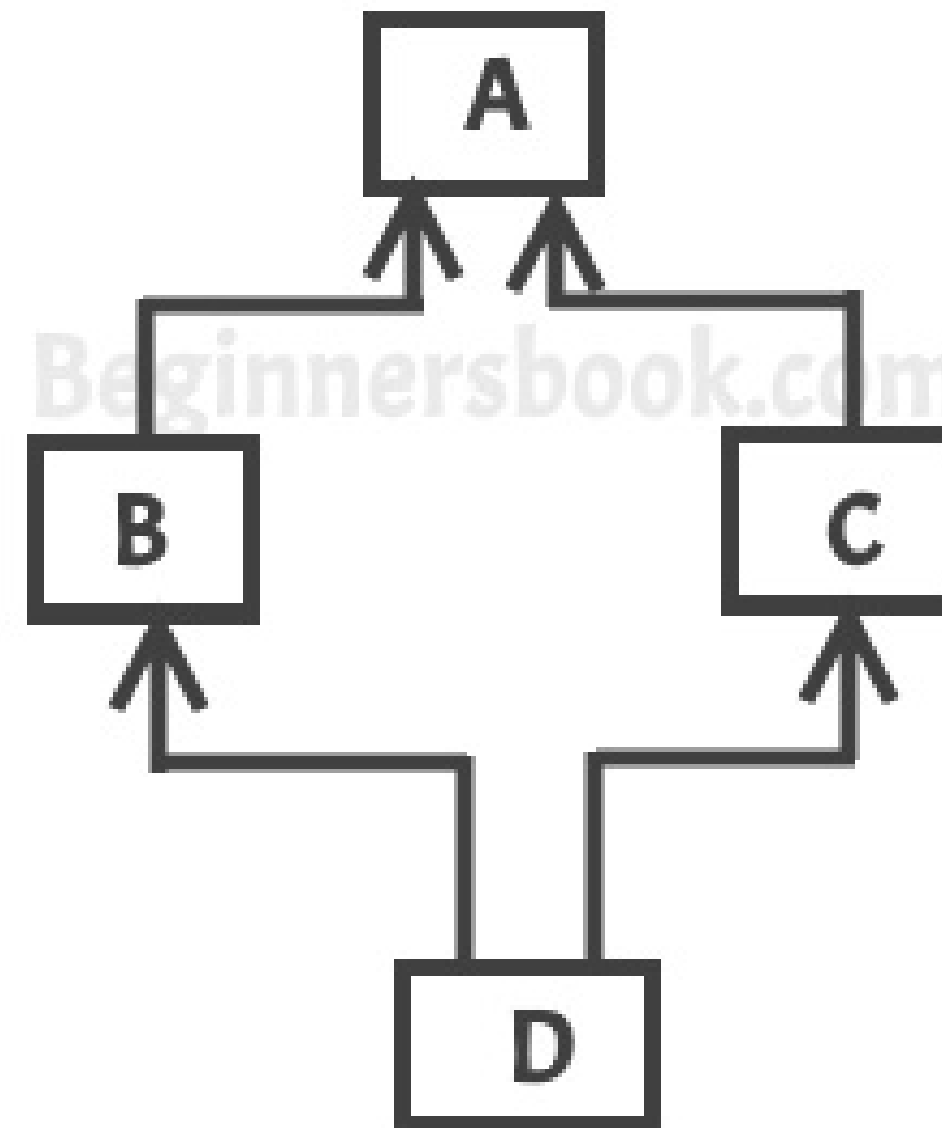
3) Hierarchical



# Inheritance



**Hierarchical Inheritance**



**Hybrid Inheritance**

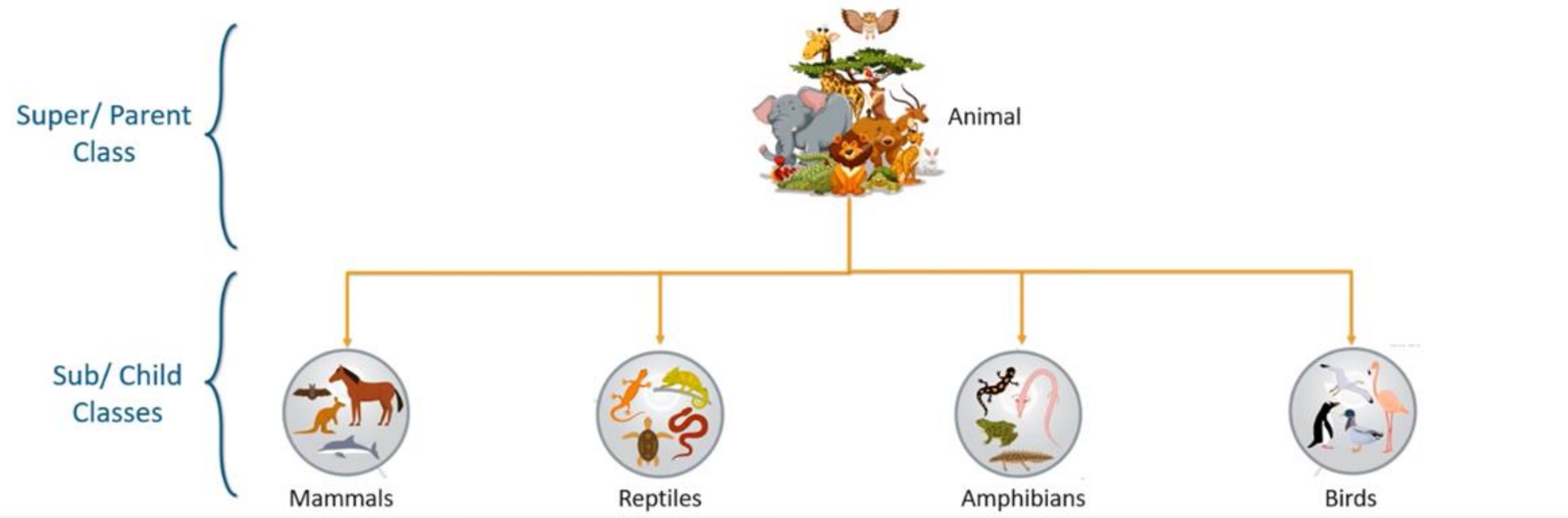




# Inheritance

Inheritance is the property of an object to acquire all the properties and behavior of its parent object

Inheritance represents the **IS-A** relationship which is also known as a parent-child relationship





# Inheritance



## Syntax

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

## Advantages



Code Reusability

Extensibility

Overriding

Data Hiding



# Polymorphism

- The word “**Polymorphism**” means “**Many Forms**”. It came from Greek “Poly”(means many) and “morphos”.(means forms)

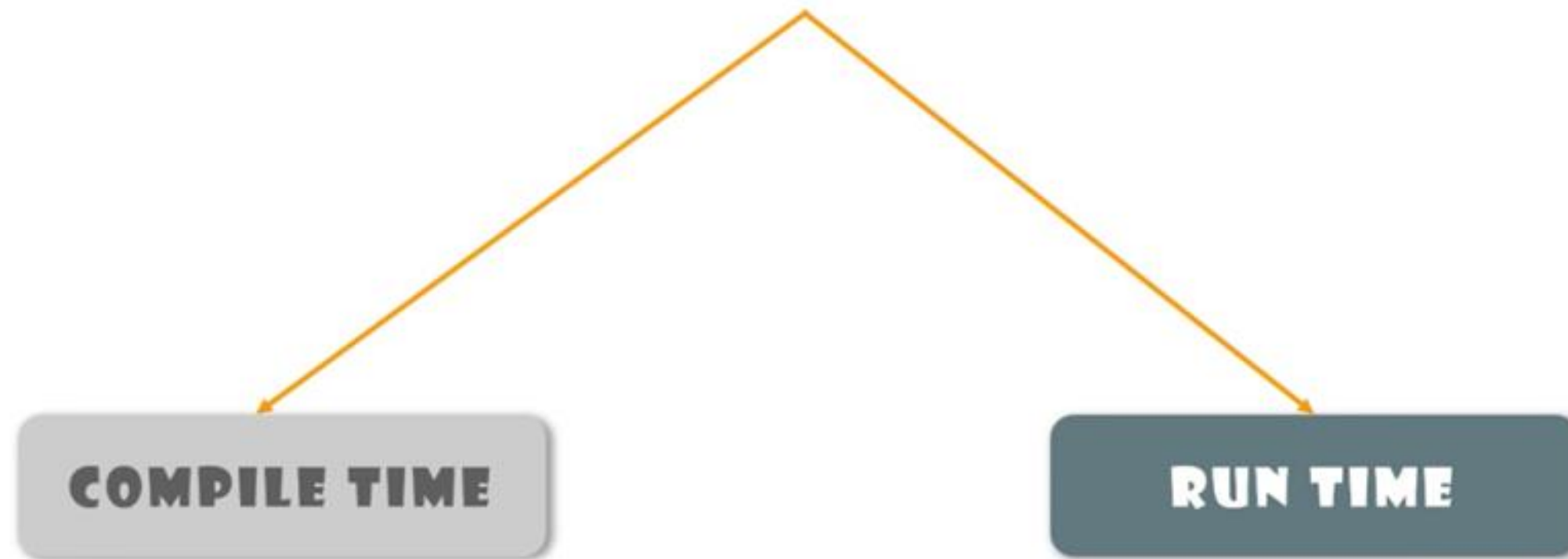




# Polymorphism

Polymorphism is the property of an object which allows it to take multiple forms

## Types Of Polymorphism in Java





# Polymorphism

*Compile Time Polymorphism* or *Static Polymorphism* is resolved during compiler time

*Overloading* is an example of compile time polymorphism

**COMPILE TIME**

## Rules For Overloading

1. Overloaded methods must have different argument list
2. It can have different return types if argument list is different
3. It can throw different exceptions
4. It can have different access modifiers



# Polymorphism

*Run Time Polymorphism* or *Dynamic Polymorphism* is resolved during run time

*Method Overriding* is an example of run time polymorphism

An overridden method is called through the reference variable of a superclass

**RUN TIME**

## Rules For Overloading

1. Overriding method argument list must match the overridden method
2. The return type must be the same or subtype of overridden method
3. Access level cannot be more restrictive than overridden method



# Abstraction

- Abstraction means hiding lower-level details and exposing only the essential and relevant details to the users

Consider an ATM Machine; All are performing operations on the ATM machine like cash withdrawal, money transfer, retrieve mini-statement...etc. but we can't know internal details about ATM.



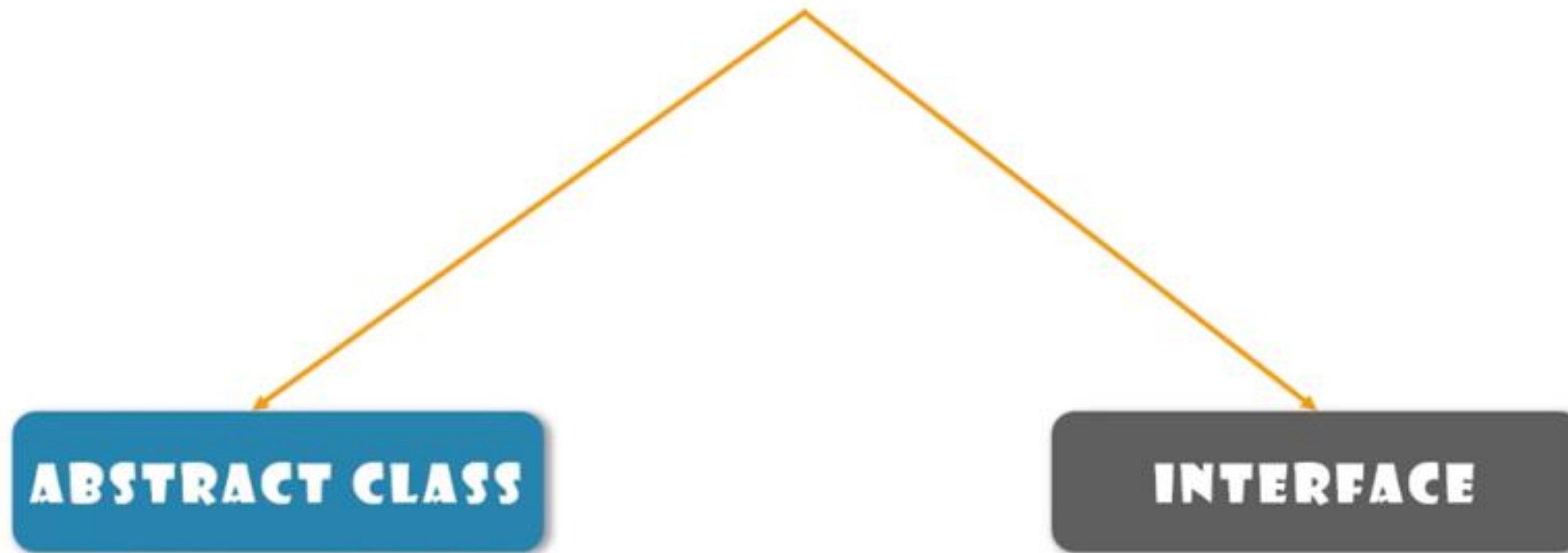
Real Life Example of Abstraction



# Abstraction

Abstraction is the methodology of hiding the implementation details from the user and only providing the functionality to them

## Ways to achieve Abstraction







# Abstraction

An abstract class is a template definition to methods and variables of a class that contains one or more abstracted methods

It can provide from 0 to 100% of abstraction

## ABSTRACT CLASS

- Must be declared with an abstract keyword
- Can have abstract and non-abstract methods
- Cannot be instantiated
- Can have constructors and static methods
- Can have final methods



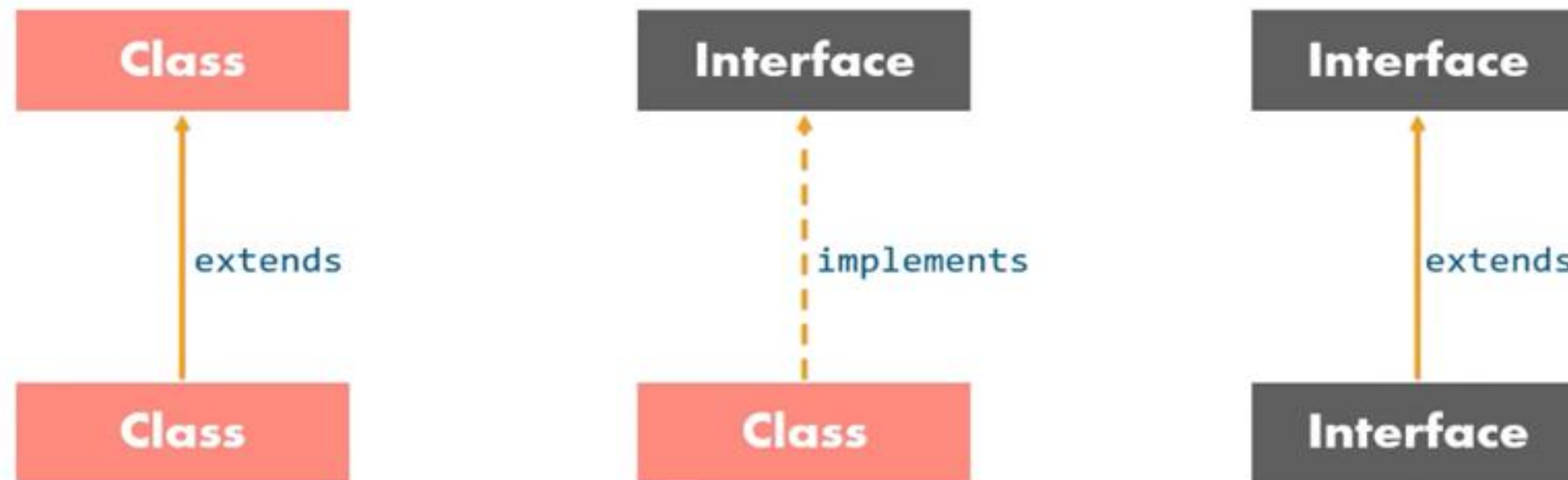
# Abstraction

An interface in java is a blueprint of a class which contains static constants and abstract methods

It Enables Multiple Inheritance and helps in achieving loose coupling

It provides 100% of abstraction

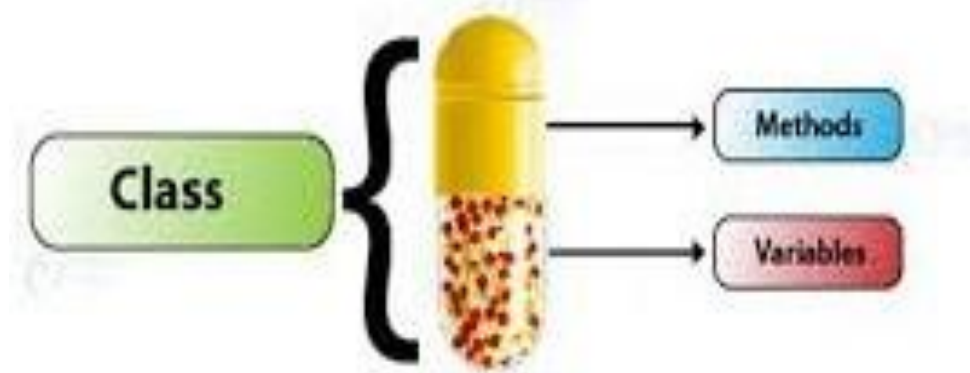
**INTERFACE**





# Encapsulation

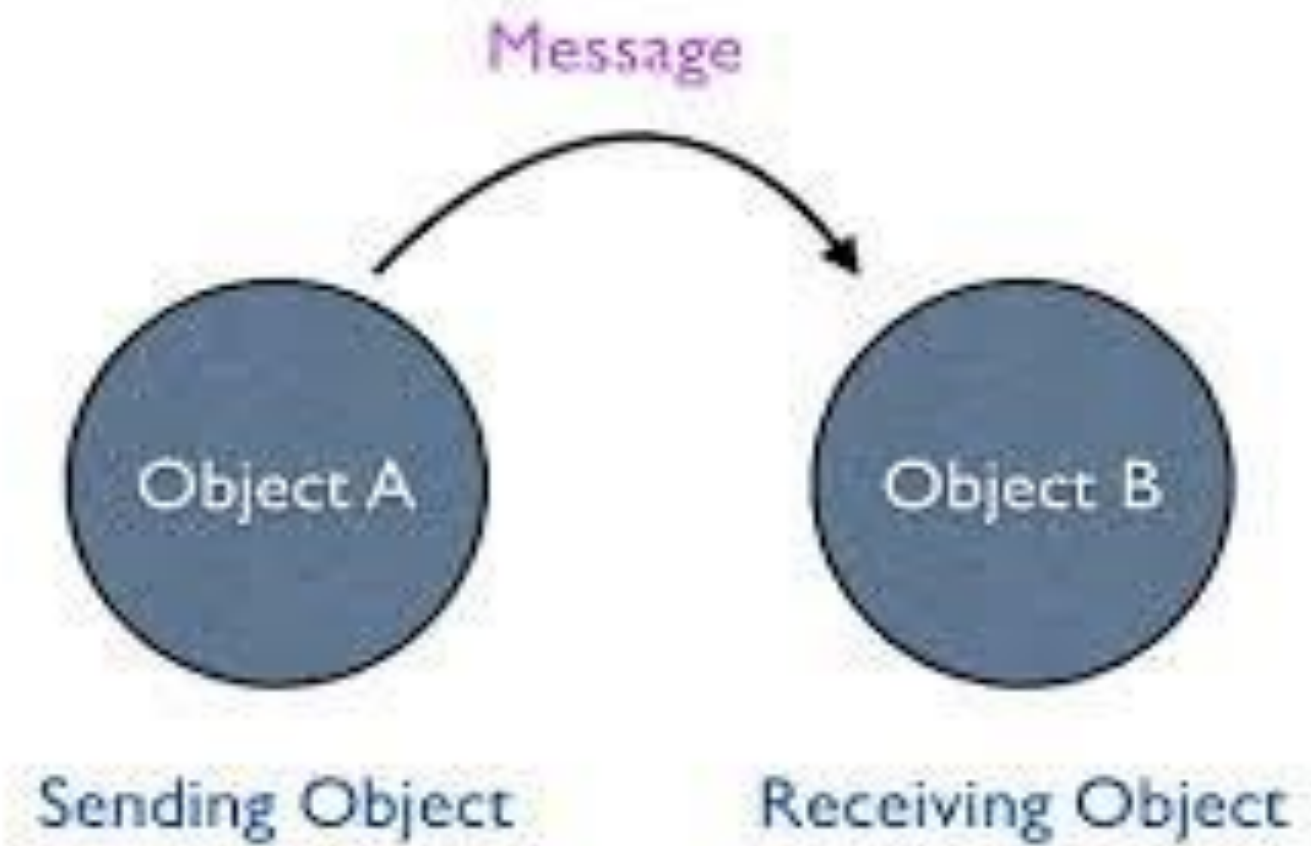
- Process of Wrapping up of data and methods into one single unit is called as encapsulation.
- It is achieved by declaring the variables of a class as a private and then providing public access methods to modify and view the variable values.
- Capsule, it is wrapped with different medicines.
- In a capsule, all medicine is encapsulated inside a capsule.





## Message Binding

- Message passing in Java is like sending an object i.e. message from one object to another object.
- It is used when threads do not have shared memory and are unable to share monitors





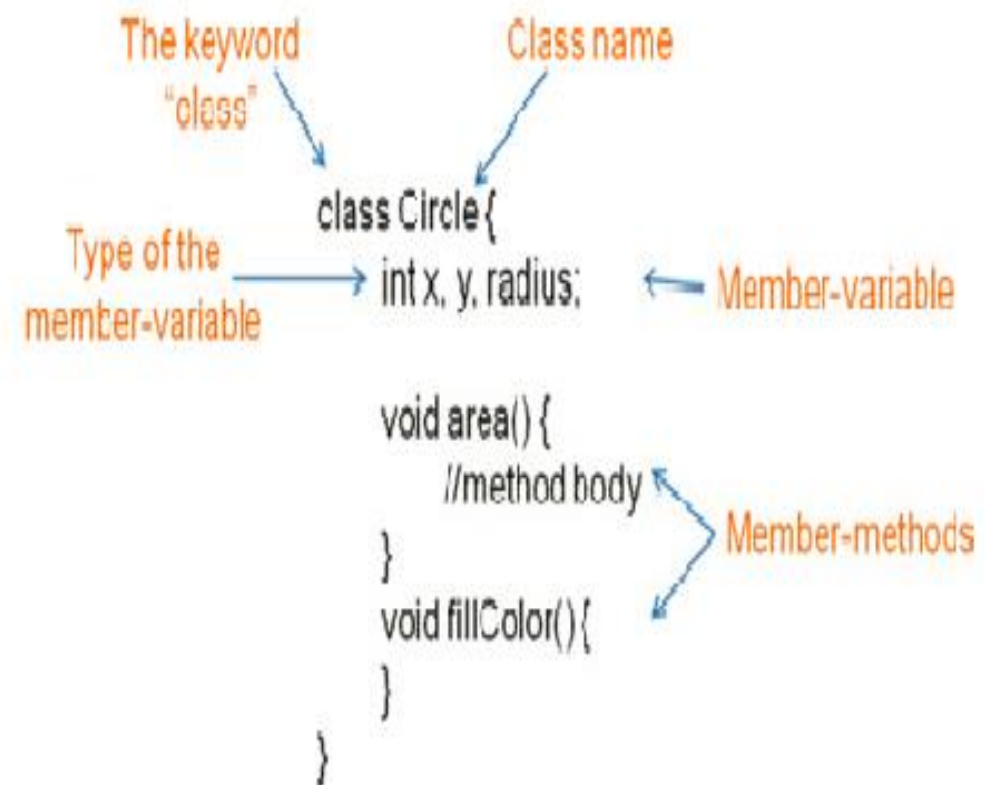
# Objects

- Syntax

Keyword      User Defined name

Class Classname

```
{  
Data member1; // Variable Declaration  
Data Member2;  
.  
.  
method name() // Method or Function Definition  
{  
Statement;  
}  
}
```





# Object Declaration

- In OOP, objects are used to access data members and methods in the class.
- Syntax:                      keyword                      Constructor
- Classname objectname = new classname();
- Example:
- Student stud = new Student();
- Keyword “new” is used to allocate memory for that class with help of constructor.