# Object Oriented Programming (OOP) Concepts

CSE 3223

Mir Tafseer Nayeem

Faculty Member, CSE AUST

tafseer.nayeem@gmail.com

# What is OOP?

- Object Oriented Programming is a programming concept that works on the principle that objects are the most important part of your program.
- It allows users create the objects that they want and then create methods to handle those objects.

# Object Orientation

- An object is something that exists in the context of a system.
- An object is an instance of a class.
- There can be multiple instances of a class in a program.

# Classes

- The class is a group of similar entities.
- A class is a category into which an objects can be categorized.
- A class is also a template from which objects can be created.


# Class and Object

- We could define attributes of the car like, ***model, fuel, makeYear*** and behaviors like ***start, break, accelerate*** etc.
- The attributes and behavior that we are specifying are not specific to just one model of car.
- To generalize a car by stating that the car which we are going to model in our program will have these number of attributes and behavior.

```
public class Car{
    private string _color;
    private string _model;
    private string _makeYear;
    private string _fuelType;

    public void Start(){
        ..
    }

    public void Stop(){
        ..
    }

    public void Accelerate(){
        ..
    }
}
```

# Class and Object

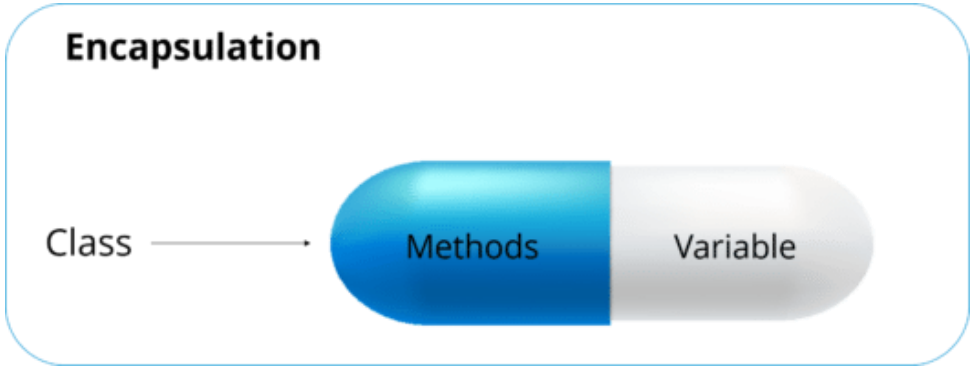
- Using the same class "Car" we can create different objects having ***variation in model***, ***fuel type*** and ***make year*** while having the same common behavior.
- OOP allows you to easily model real world complex system behavior. With OOP, data and functions (attributes and methods) are bundled together within the object.
- Which is a core difference between the object oriented and procedural approaches.

| **Object 1** | |
|---|---|
| Model | Volkswagen Polo |
| Fuel | Petrol |
| Make | 2017 |
| Start()<br>Break()<br>Accelerate() | |

| **Object 2** | |
|---|---|
| Model | Volkswagen Vento |
| Fuel | Diesel |
| Make | 2017 |
| Start()<br>Break()<br>Accelerate() | |

# Encapsulation

- Encapsulation is a mechanism where you bind your data and code together as a single unit.
- It also means to hide your data in order to make it safe from any modification.
- Similarly, through encapsulation the methods and variables of a class are well hidden and safe.



# Encapsulation

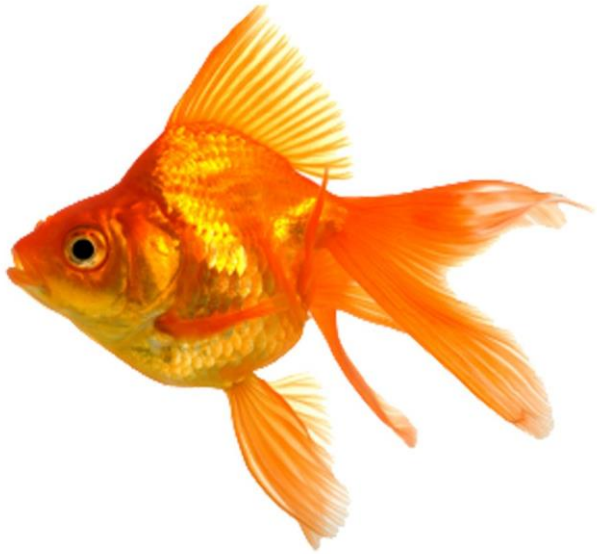
We can achieve encapsulation in Java by:

- Declaring the variables of a class as private.
- Providing public ***setter and getter methods*** to modify and view the variables values.

```
1  public class Employee {
2   private String name;
3   public String getName() {
4   return name;
5   }
6   public void setName(String name) {
7   this.name = name;
8   }
9   public static void main(String[] args) {
10  }
11 }
```

# Attributes and Operations

- An Object contains both the data and the function, which operates on the data.
- Classes (and the objects within a class) have:
  - **Attributes**: Properties
    - Size, color, gender
  - **Operations**: Functionality
    - Swim, eat, be eaten

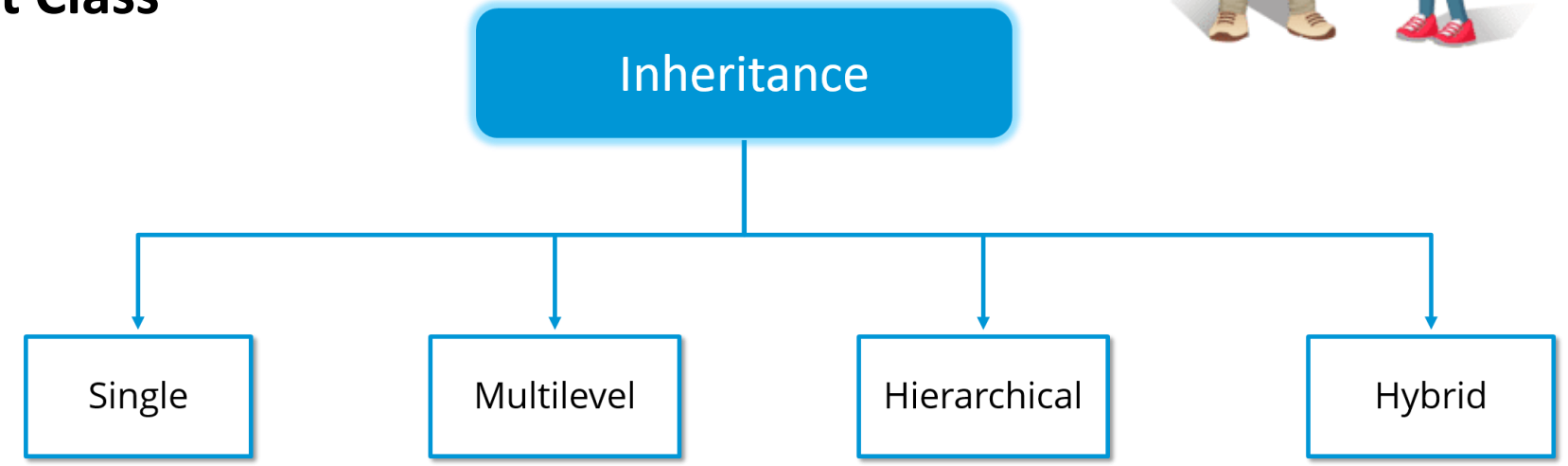


# Inheritance

- Inheritance is a ***powerful feature*** of OOP languages.
- Inheritance helps in ***organizing classes into a hierarchy*** and enabling these classes to ***inherit attributes and behavior*** from classes above in the hierarchy.
- Inheritance describes an "***IS A***" relationship.
- Inheritance is a mechanism for ***code reuse*** and can help in ***reducing duplication*** of code.

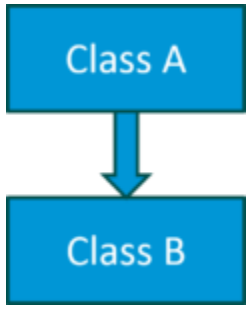
# Inheritance

- **Parent Class** (Super or Base class)
- **Child Class** (Subclass or Derived class )
- A class which inherits the properties is known as **Child Class** whereas a class whose properties are inherited is known as **Parent Class**

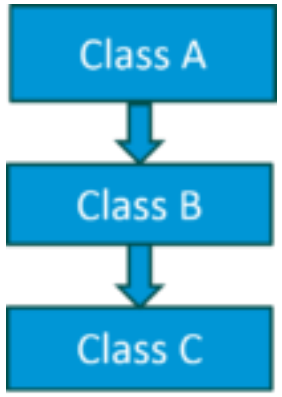
# Inheritance

- **Single Inheritance**




```
1  Class A
2  {
3  ---
4  }
5  Class B extends A {
6  ---
7  }
```

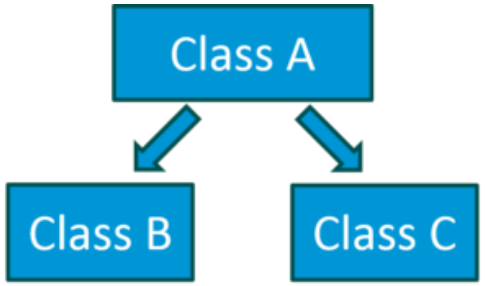
- **Multilevel Inheritance**




```
1  Class A{
2  ---
3  }
4  Class B extends A{
5  ---
6  }
7  Class C extends B{
8  ---
9  }
```

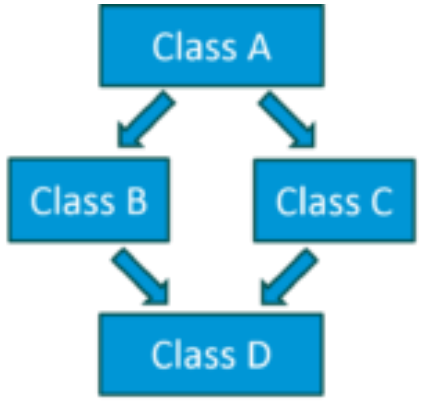
# Inheritance

- **Hierarchical Inheritance**



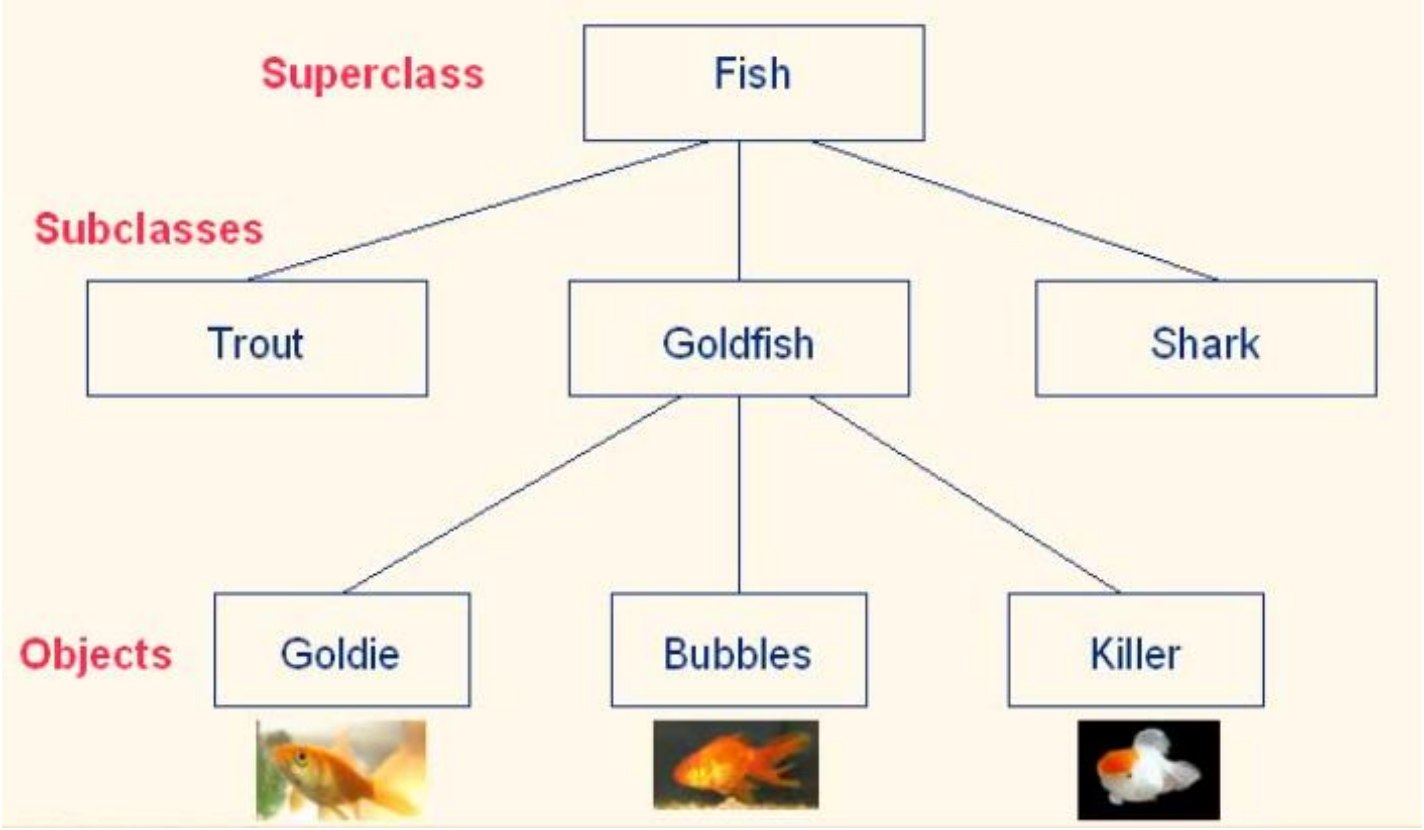

```
1  Class A{
2  ---
3  }
4  Class B extends A{
5  ---
6  }
7  Class C extends A{
8  ---
9  }
```

- **Hybrid Inheritance**




- Hybrid inheritance is a combination of *multiple* inheritance and *multilevel* inheritance.

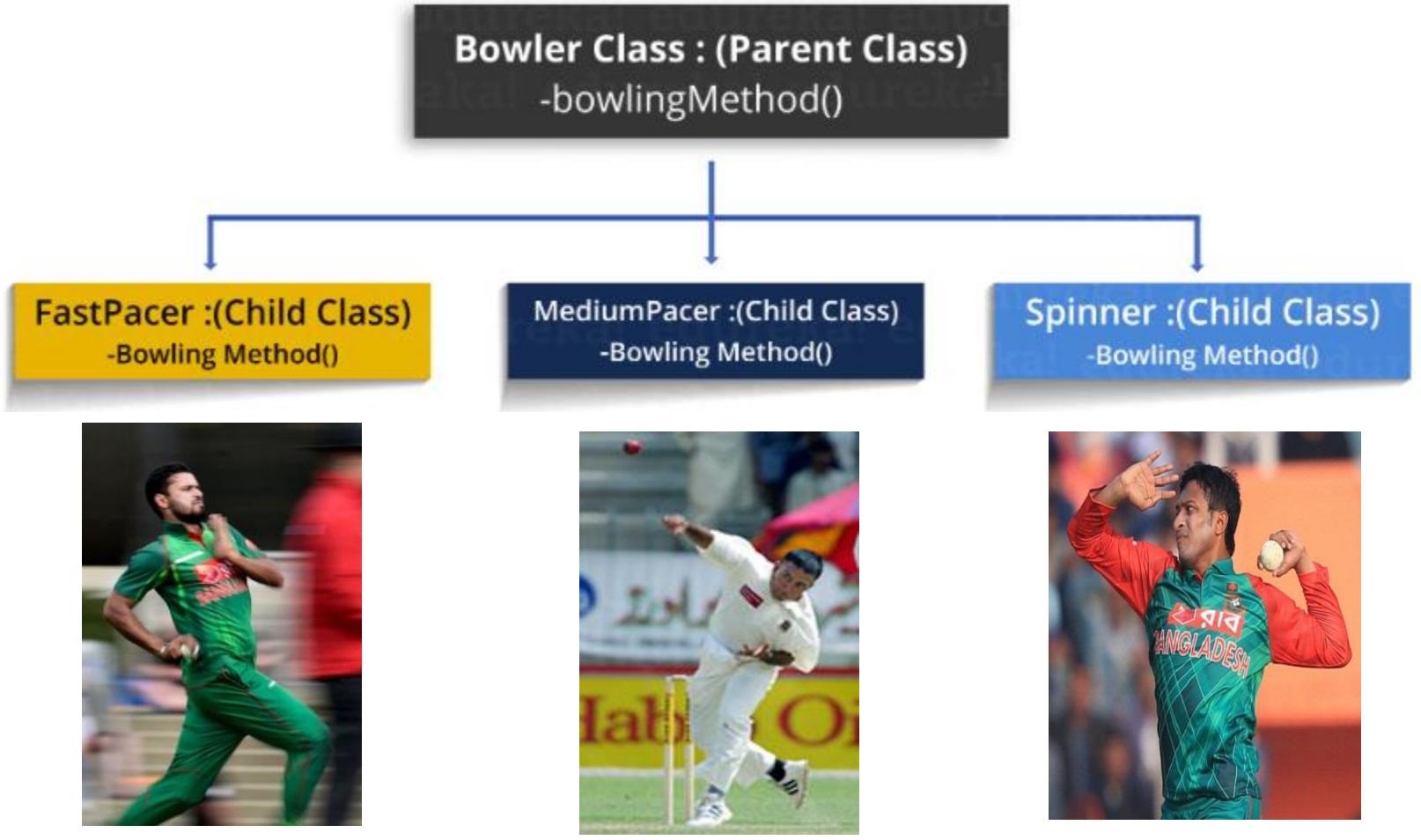
# Fish Example



# Polymorphism

- Polymorphism means taking **many forms**, where '**poly**' means many and '**morph**' means forms.
- Polymorphism allows you define one interface or method and have multiple implementations.
- For example, if you needed to write a message on a piece of paper, you could use a pen, pencil, marker or even a colored chalk.

# Example


Bowler Class : (Parent Class)
-bowlingMethod()
FastPacer :(Child Class)
-Bowling Method()
MediumPacer :(Child Class)
-Bowling Method()
Spinner :(Child Class)
-Bowling Method()
Mashrafee
Sujon
Shakib

# END