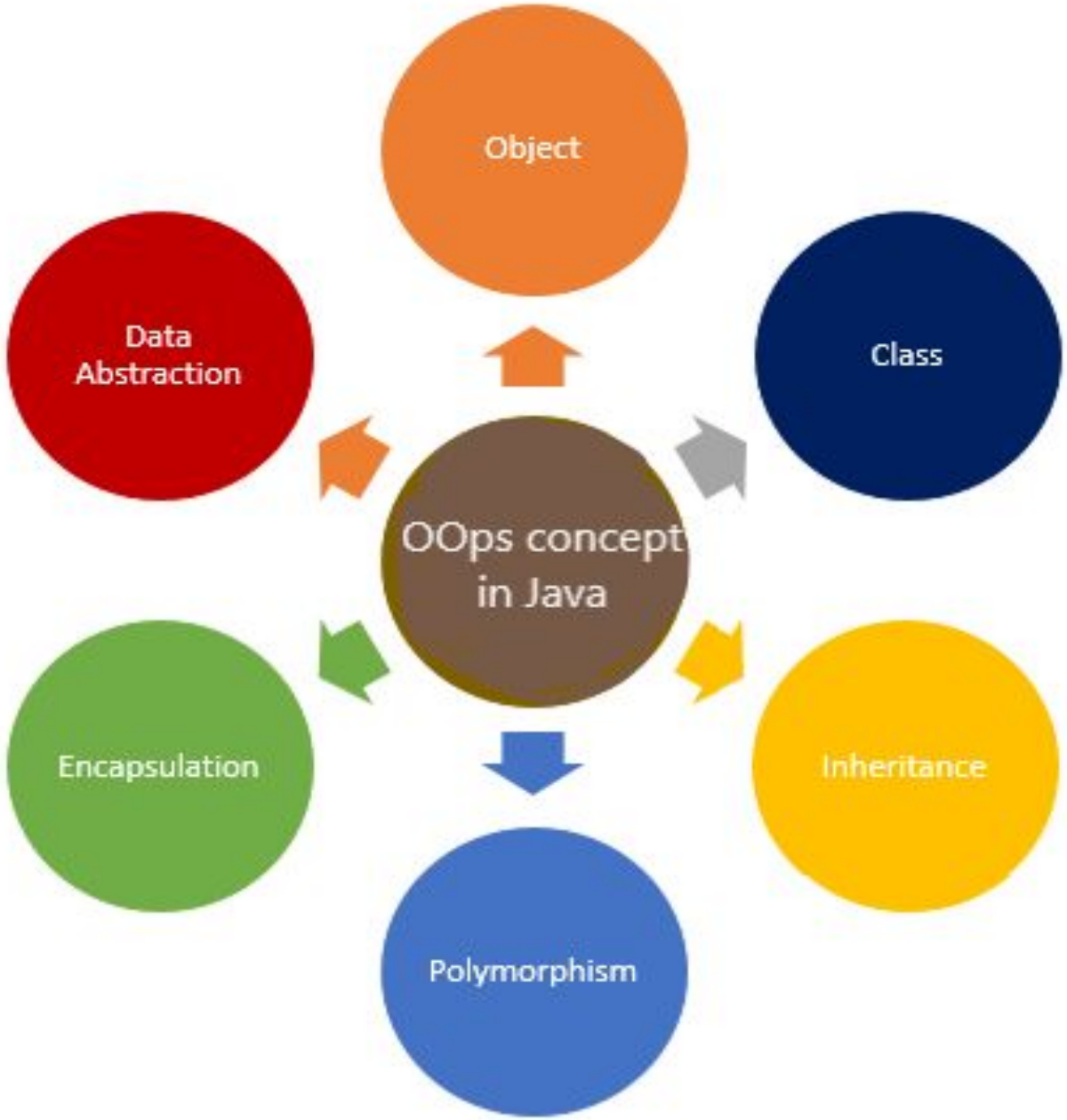


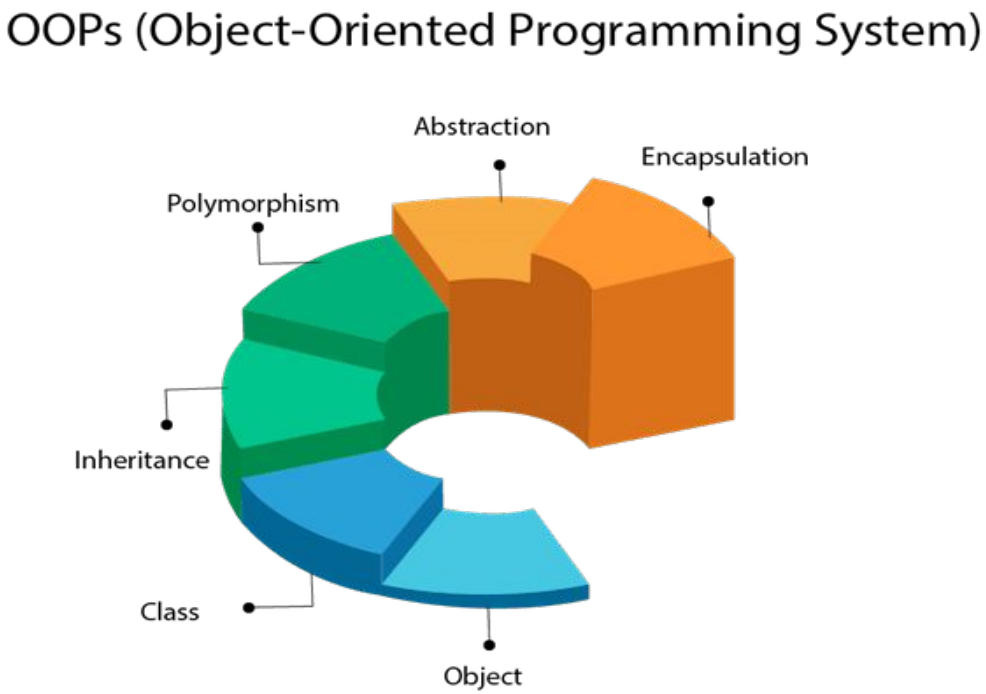
CS20004: Object Oriented Programming using Java

Lec-2

January 12, 2024

In this Discussion . . .

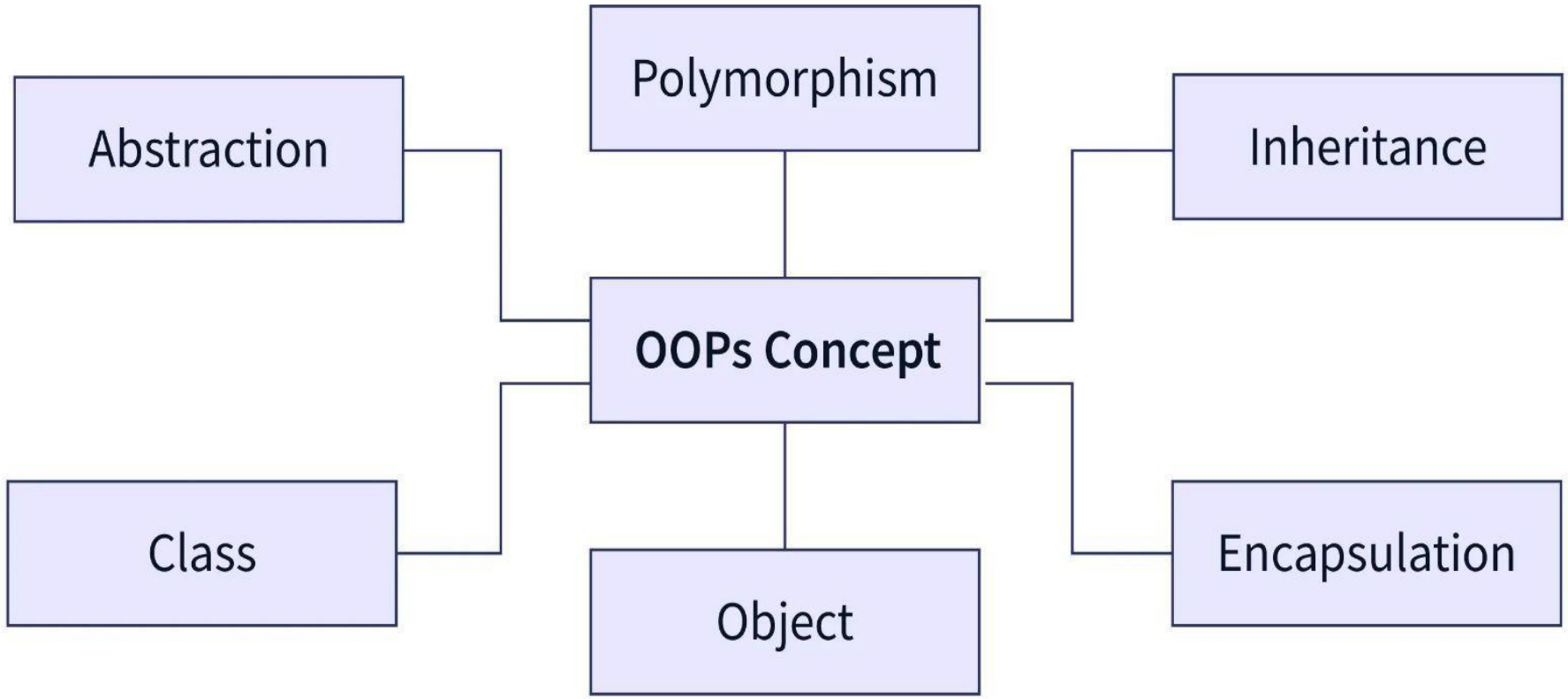
- OOP concept:
  - Object
  - Class
  - Polymorphism
  - Inheritance
  - Encapsulation
  - Abstraction
- References



OOPS Concept

- As the name suggests, objects in programming.
  - A programming model is based upon the concepts of objects, or where everything is represented as an object.
  - In OOP, we try to see the whole world as an object or in terms of objects.
  - **For example**, dogs have states or data members ***like*** color, hungry, and breed, **and** the behaviors or methods (actions they can perform) are barking, wagging their tail, eating, sleeping, etc.

OOPS Concept



Object

- **Object** is an instance of a class.
- An object in OOPS is a self-contained component which consists of methods and properties to make a particular type of data useful.
  - For example color name, table, bag, barking.
  - When you send a message to an object, you are asking the object to invoke or execute one of its methods as defined in the class.

Syntax:

```
ClassName ReferenceVariable = new ClassName();
```

Class

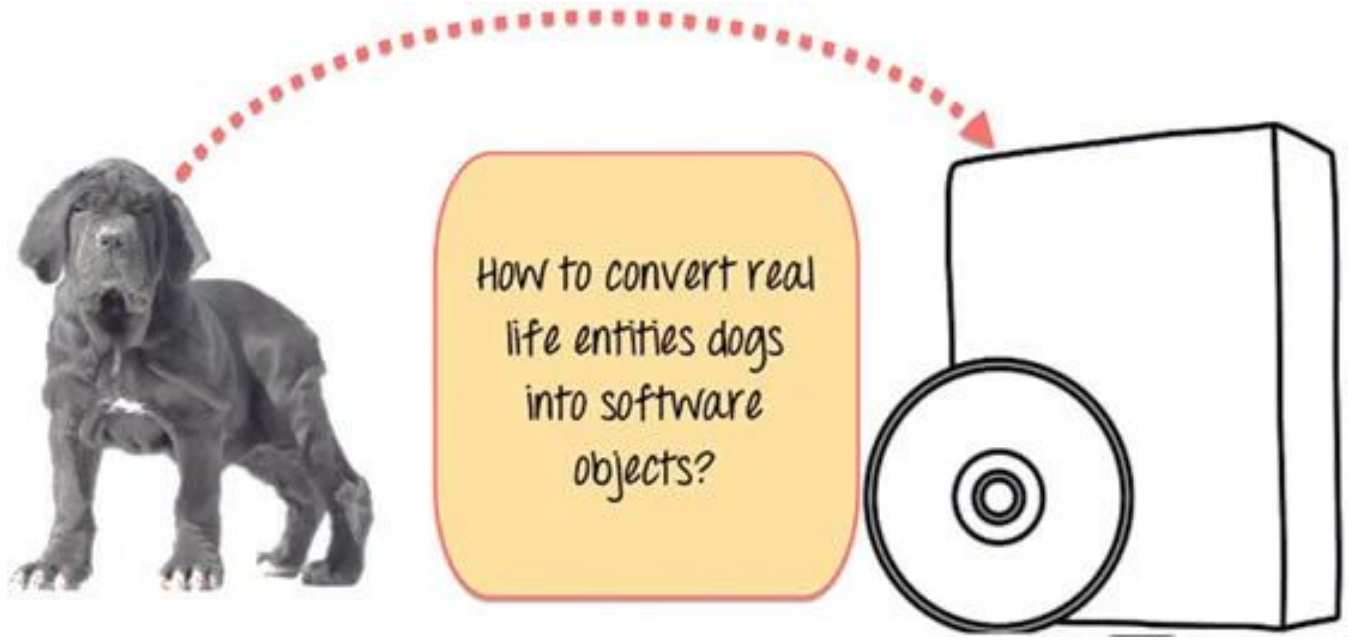
- Class are a blueprint or a set of instructions to build a specific type of object.
- It is a basic concept of OOPs which revolve around the real-life entities.
- Class in Java determines how an object will behave and what the object will contain.

Syntax:

```
class <class_name>
{
    field;  // or states
    method;  //or behaviors
}
```

Understanding the concept of Object and Class

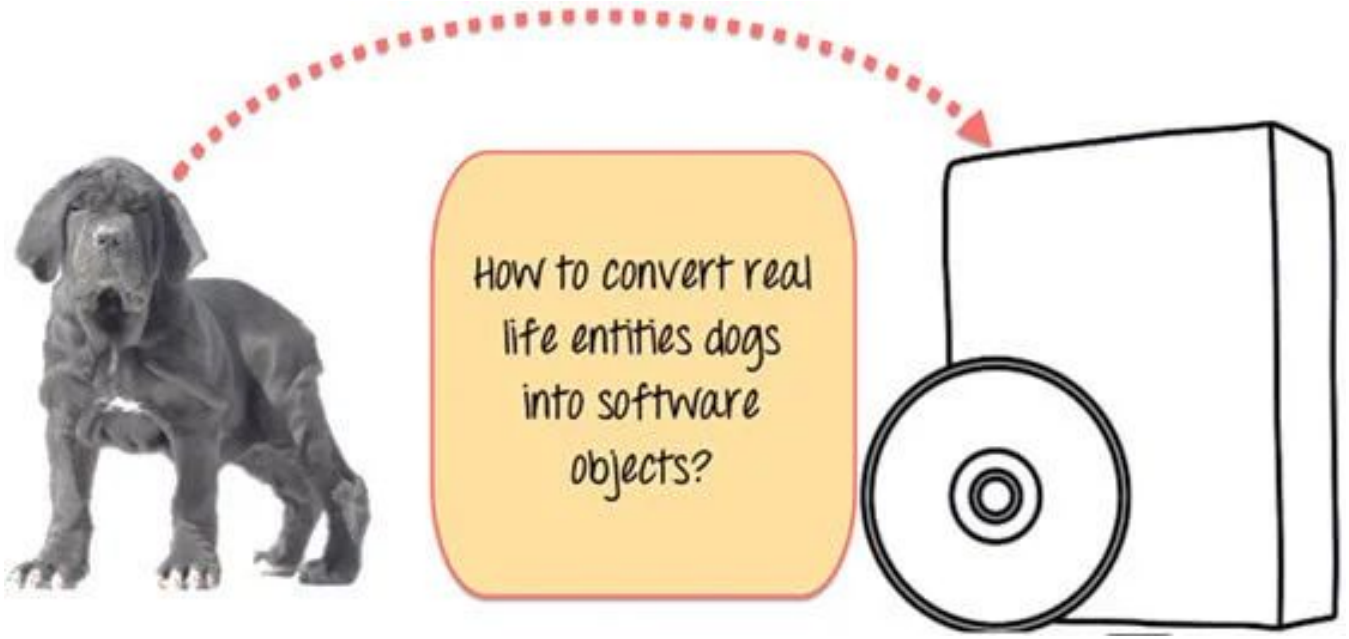
(Ex- Pet Management System)



Understanding the concept of Object and Class

(Ex- Pet Management System)

- Specific to dogs, we will need various information about the dogs like different breeds of the dogs, the age, size, etc. We need to model real-life beings, i.e., dogs into software entities.

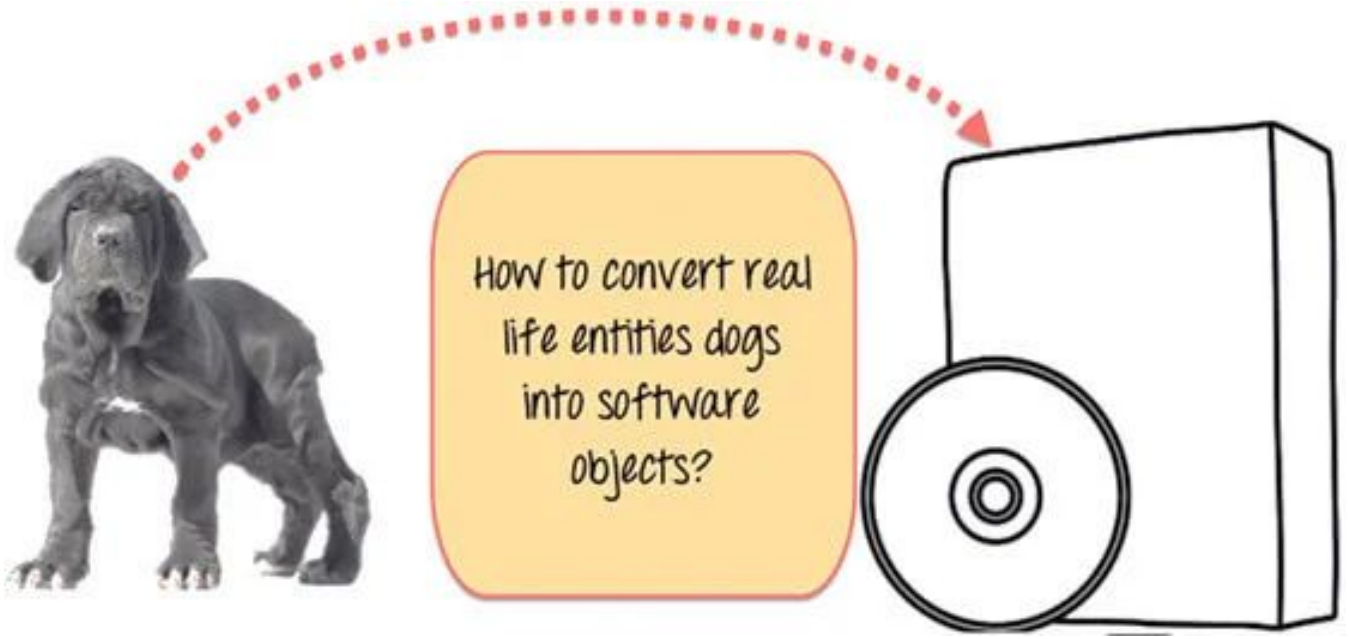


Understanding the concept of Object and Class

(Ex- Pet Management System)

- Specific to dogs, we will need various information about the dogs like different breeds of the dogs, the age, size, etc. We need to model real-life beings, i.e., dogs into software entities.

How to Design such software?



Understanding the concept of Object and Class

(Ex- Pet Management System)

Look at the picture of three different breeds of dogs above.

Understanding the concept of Object and Class

(Ex- Pet Management System)

Let’s List down the differences between them.

Understanding the concept of Object and Class

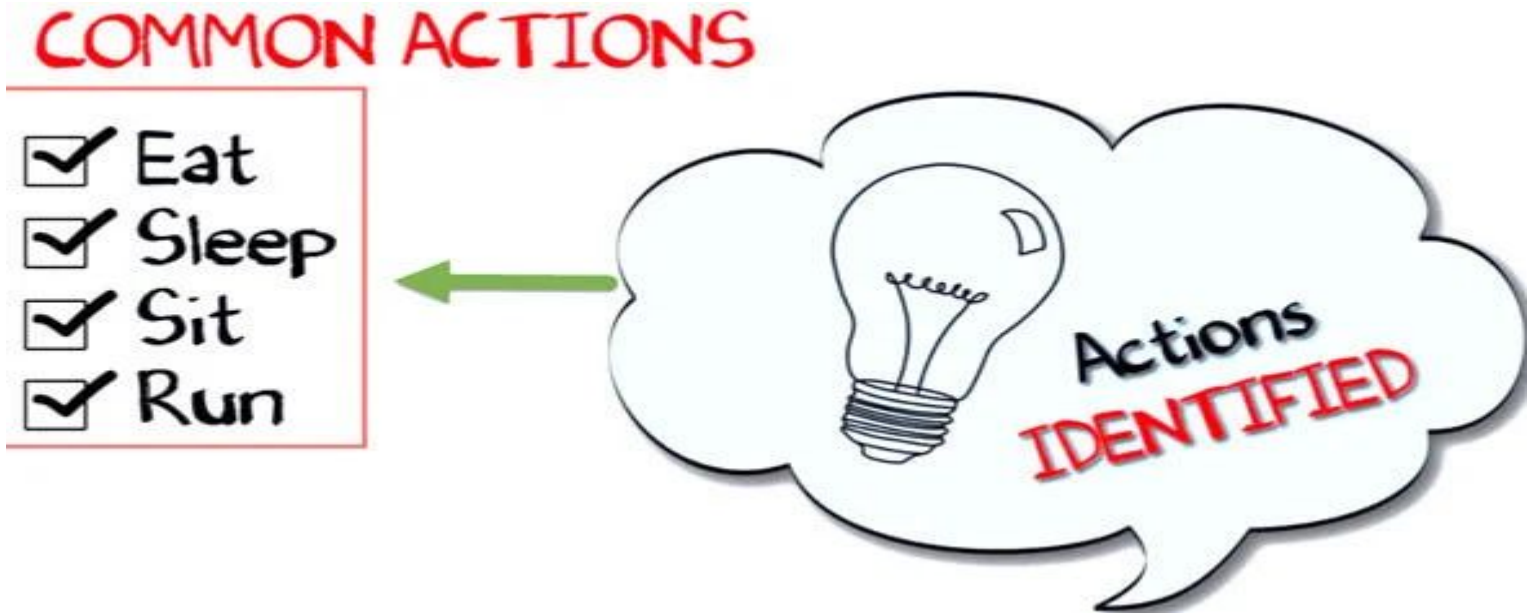
(Ex- Pet Management System)

- Some of the differences that might be listed out maybe breed, age, size, color, etc. These differences are also some common characteristics shared by these dogs. These characteristics (breed, age, size, color) can form a data members for our object.

Understanding the concept of Object and Class

(Ex- Pet Management System)

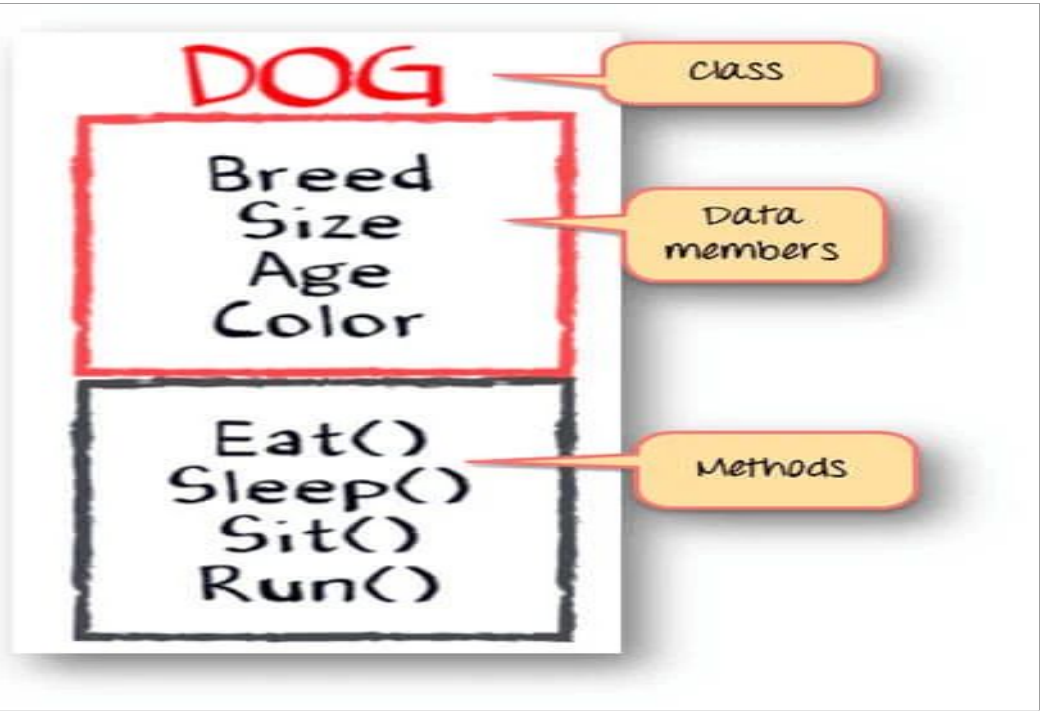
- Next, let's list out the common behaviors of these dogs like sleep, sit, eat, etc. So these will be the actions of our software objects.



Understanding the concept of Object and Class (Ex- Pet Management System)

- So far we have defined following things:

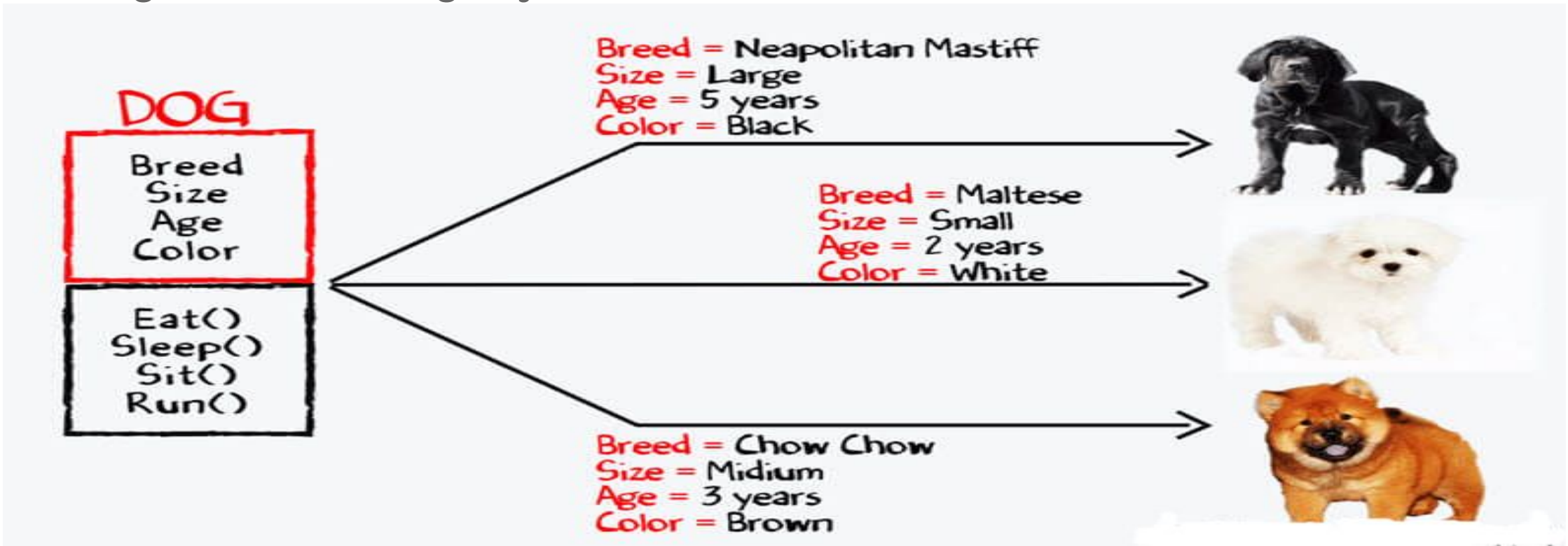
- **Class** : Dogs
- **Data members** : size, age, color, breed, etc.
- **Methods** : eat, sleep, sit and run.



Understanding the concept of Object and Class

(Ex- Pet Management System)

- Now, for different values of data members (breed size, age, and color) in a class, we will get different dog objects.



Understanding the concept of Object and Class

(Ex- Pet Management System)
Java Source Code Demo

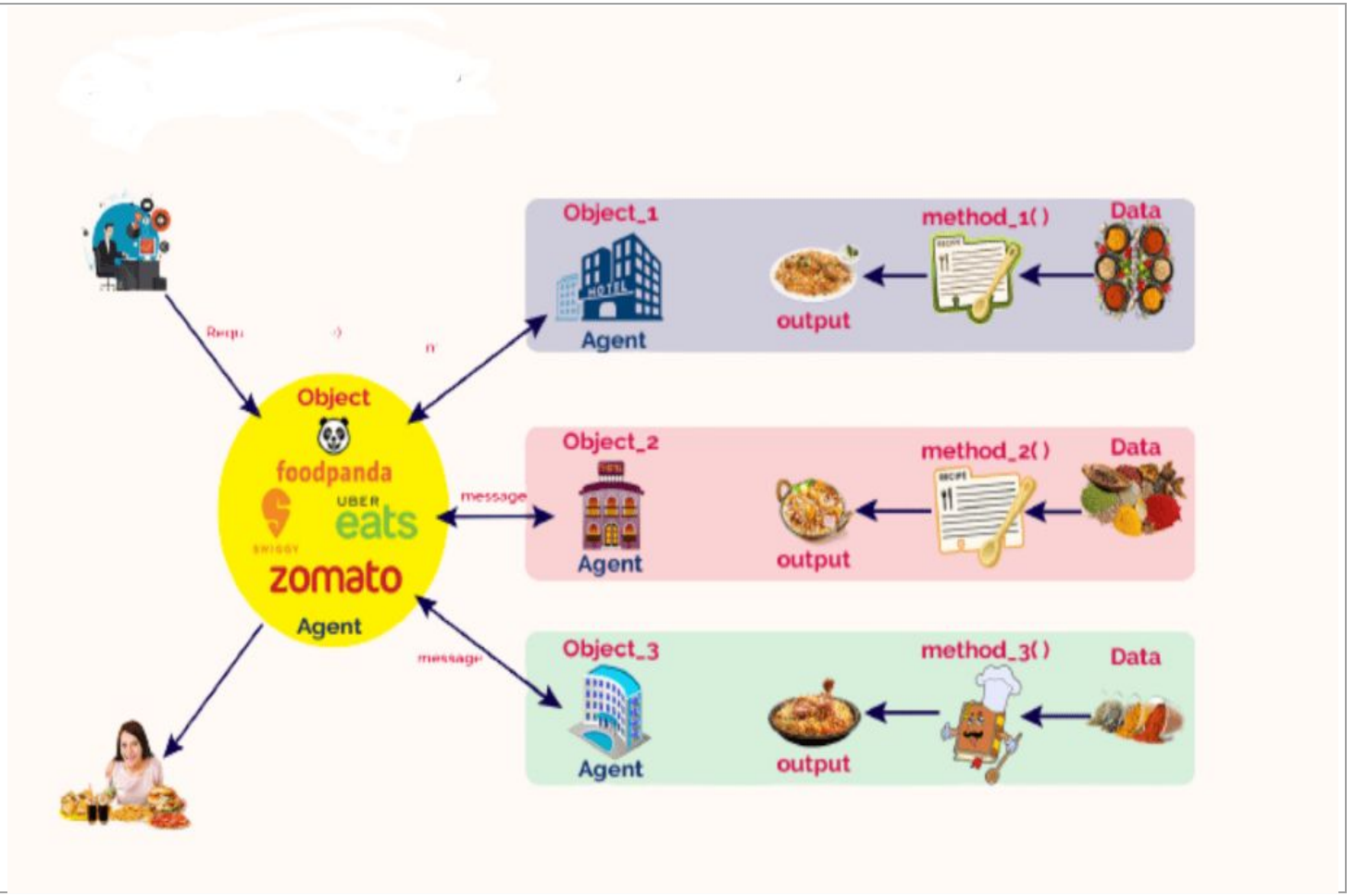
```
// Class Declaration
public class Dog {
    // Instance Variables
    String breed;
    String size;
    int age;
    String color;

    // method 1
    public String getInfo() {
        return ("Breed is: "+breed+" Size is:"+size+" Age
is:"+age+" color is: "+color);
    }
```

```
public static void main(String[] args) {
        Dog maltese = new Dog();
        maltese.breed="Maltese";
        maltese.size="Small";
        maltese.age=2;
        maltese.color="white";
        System.out.println(maltese.getInfo());
    }
}
```

Some More Examples of Class and Objects

I. Online Food Delivery

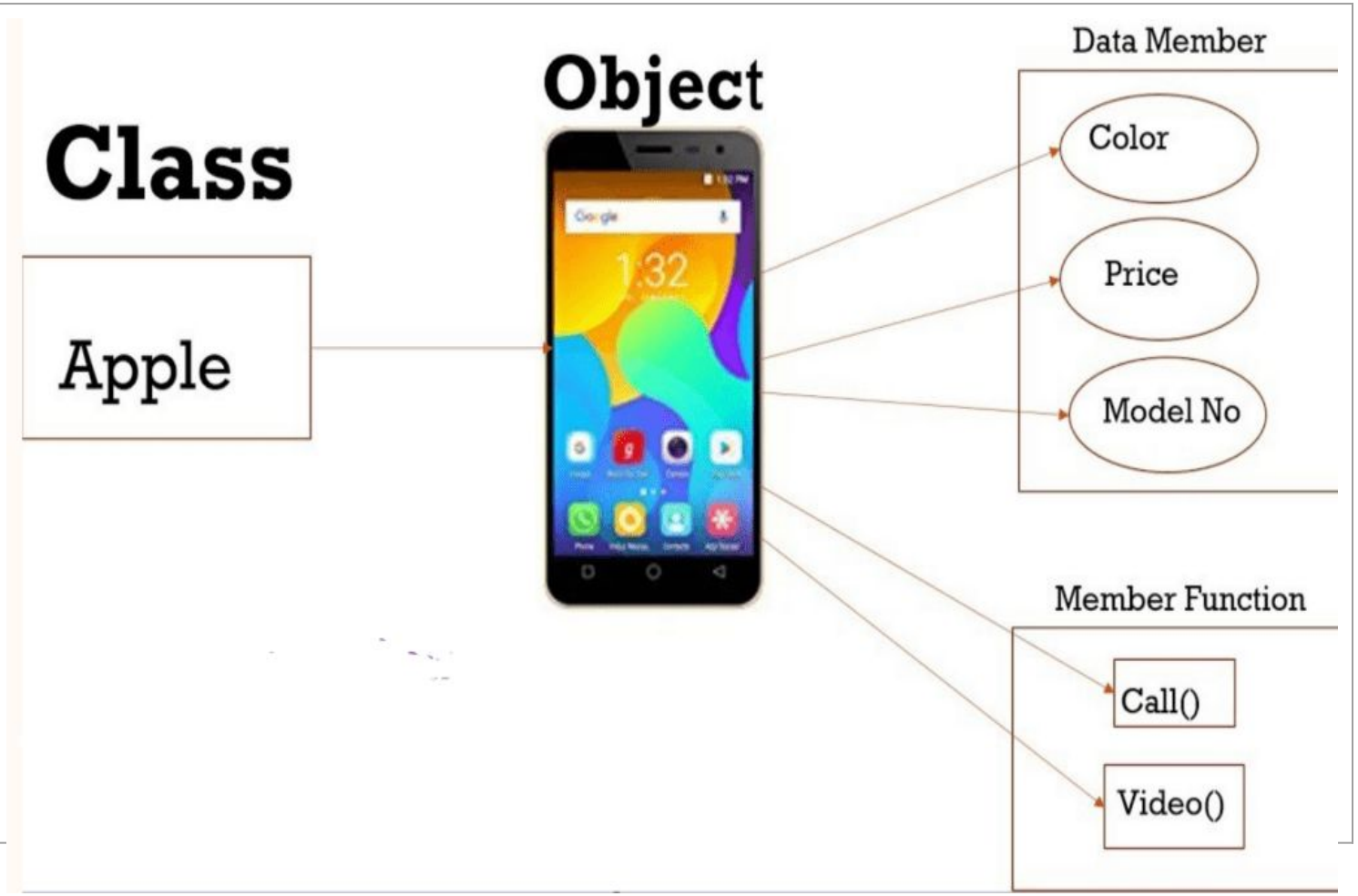
- We will place an order for fast food;
- the agent will pick and deliver the order to customers' locations when it comes to the hotel.
- Class of **order** will be called, and 3 objects and methods of class order will be called



Some More Examples of Class and Objects

II. Electronic items

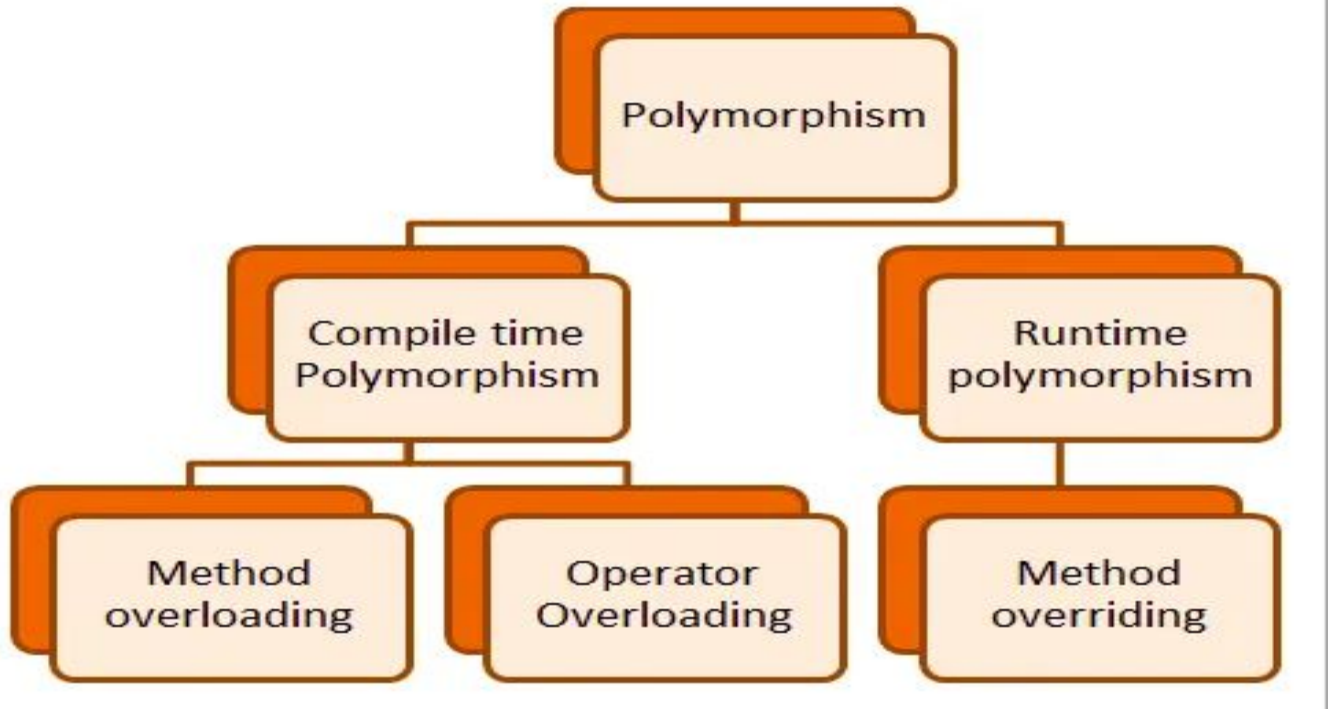
- Let's take the example of mobile phones as an object which belongs to any company say 'Apple'.
- Mobiles can contains be associated many Data members (like Color, Price, Model No) and Member functions (like Call() and Video)



OOP Concepts: Polymorphism

- Refers to the ability to exist in many forms.
- Offers great scalability and boosts the code's readability.
- Generally, polymorphism can be obtained using inheritance which implies that the class must belong to the same hierarchy tree.
- Programmatically, you can perform different operations using only one method or function.

Java Polymorphism Types

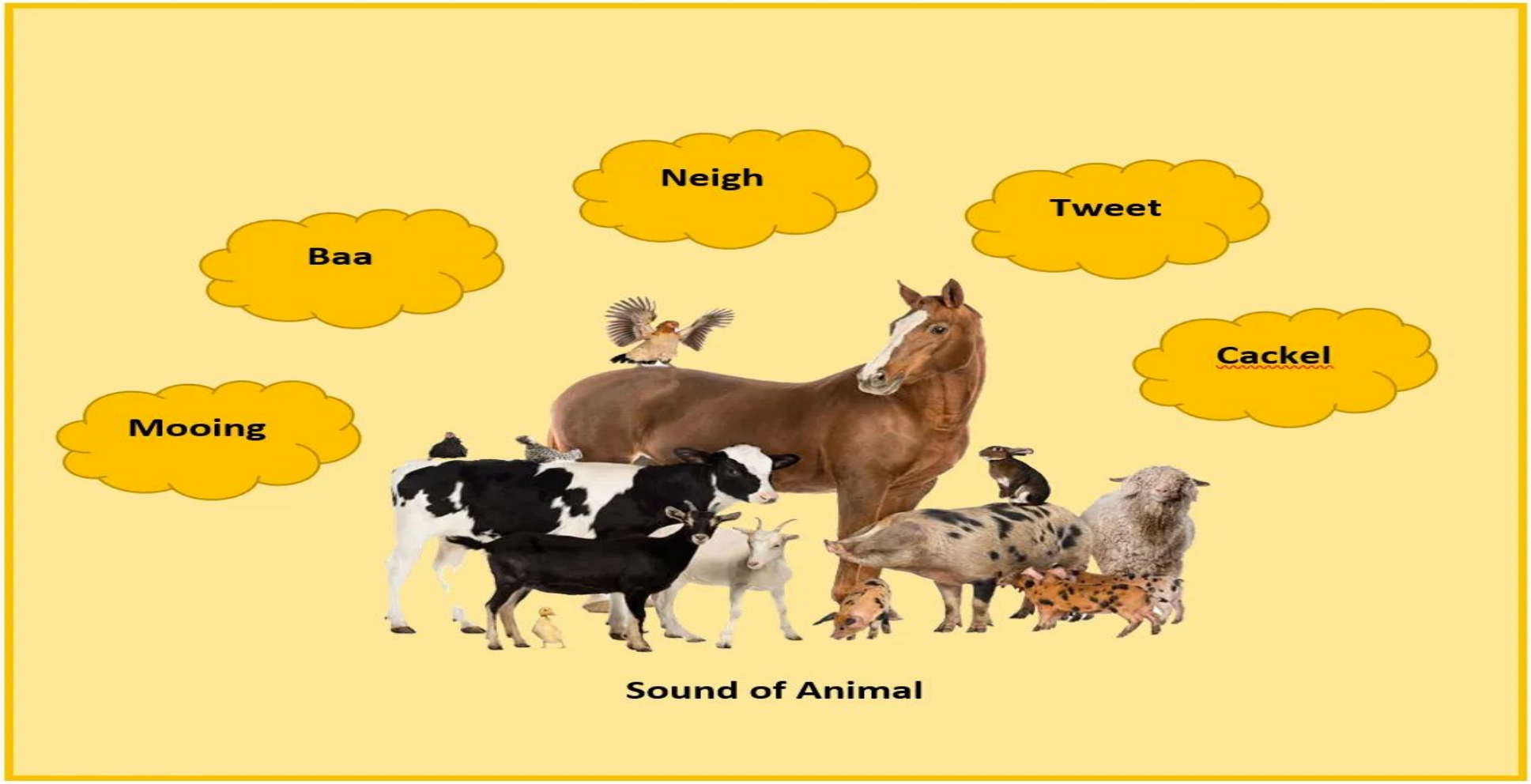


OOP Concepts: Polymorphism Examples

- We can take a boy as a real-world example. This boy can be a student, a player, and a writer. So that this boy can exist in different ways in different situations.

OOP Concepts: Polymorphism Examples

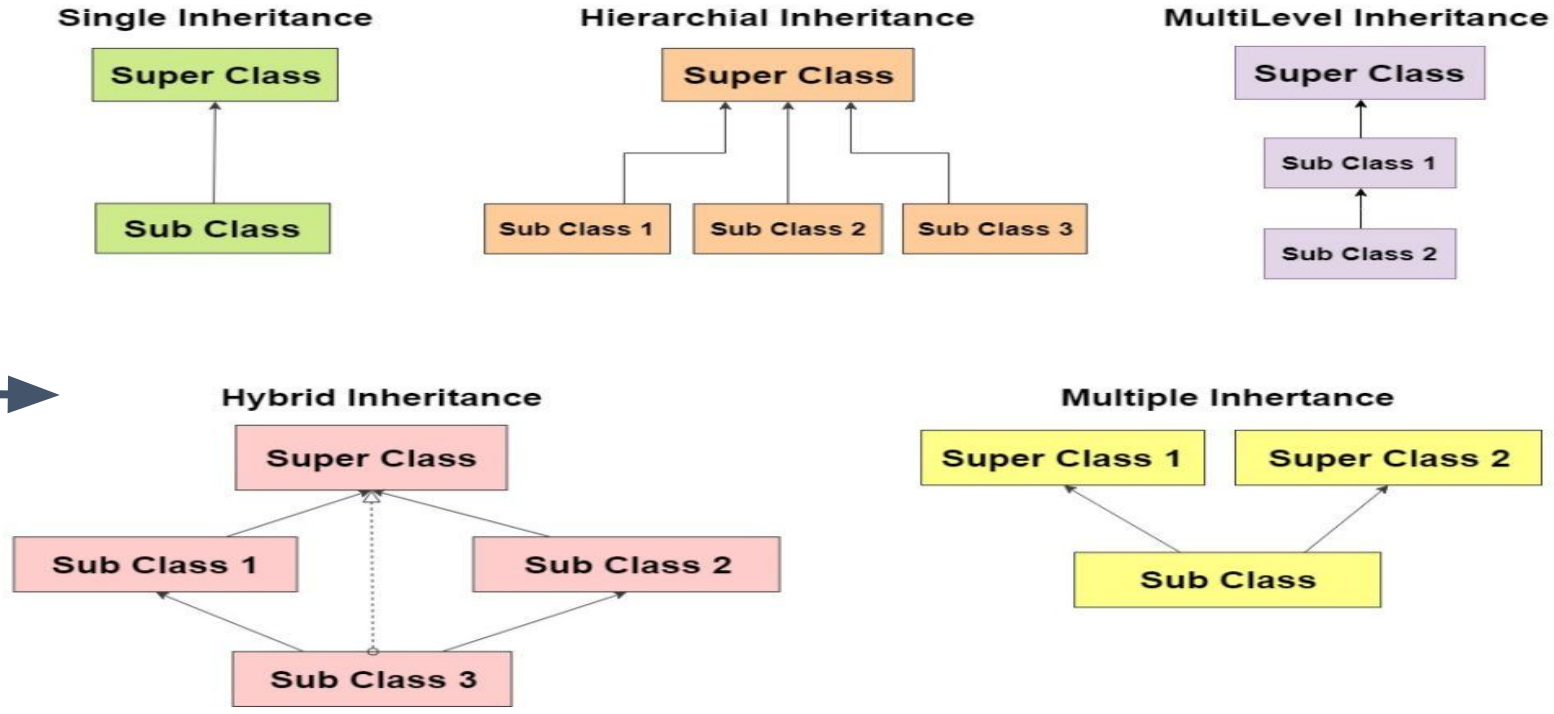
- **Sound of animals.** People have the same sound but different animals make different sounds. The following diagram shows few different sounds make by animals.



OOP Concepts: Inheritance

- Inheritance means it allows classes to inherit common properties from the parent class.
- Inheritance in Java is typically used to create a new class that adds functionality to an existing class. The class which inherits the properties of other is known as subclass (derived class / child class) and the class whose properties are inherited is known as superclass (base class / parent class).

Java Inheritance Types



OOP Concepts: Inheritance Examples

- Let's assume that there is a class as Vehicle. All vehicles are not the same. We can inherit common properties like color, size, type from the parent vehicle class and create classes like Car, Bus, Bike.

OOP Concepts: Inheritance Examples

- Let's take another parent class as Animals. Here also we can inherit common properties like name, sound, color, breed from Animal class and create classes like Dog, Cat, Horse and etc.

OOP Concepts: Encapsulation

- Encapsulation means it binds data and code together into one unit, i.e., integrating data (variables) and code (methods) into a single unit.
- In encapsulation, a class's variables are hidden from other classes and can only be accessed by the methods of the class in which they are found.

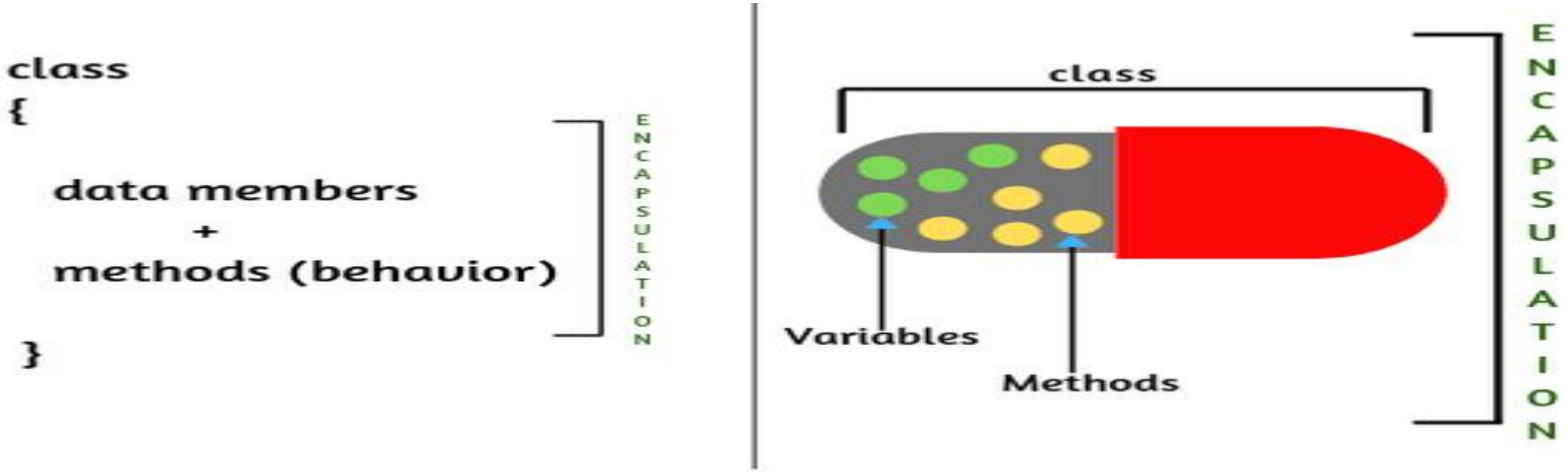
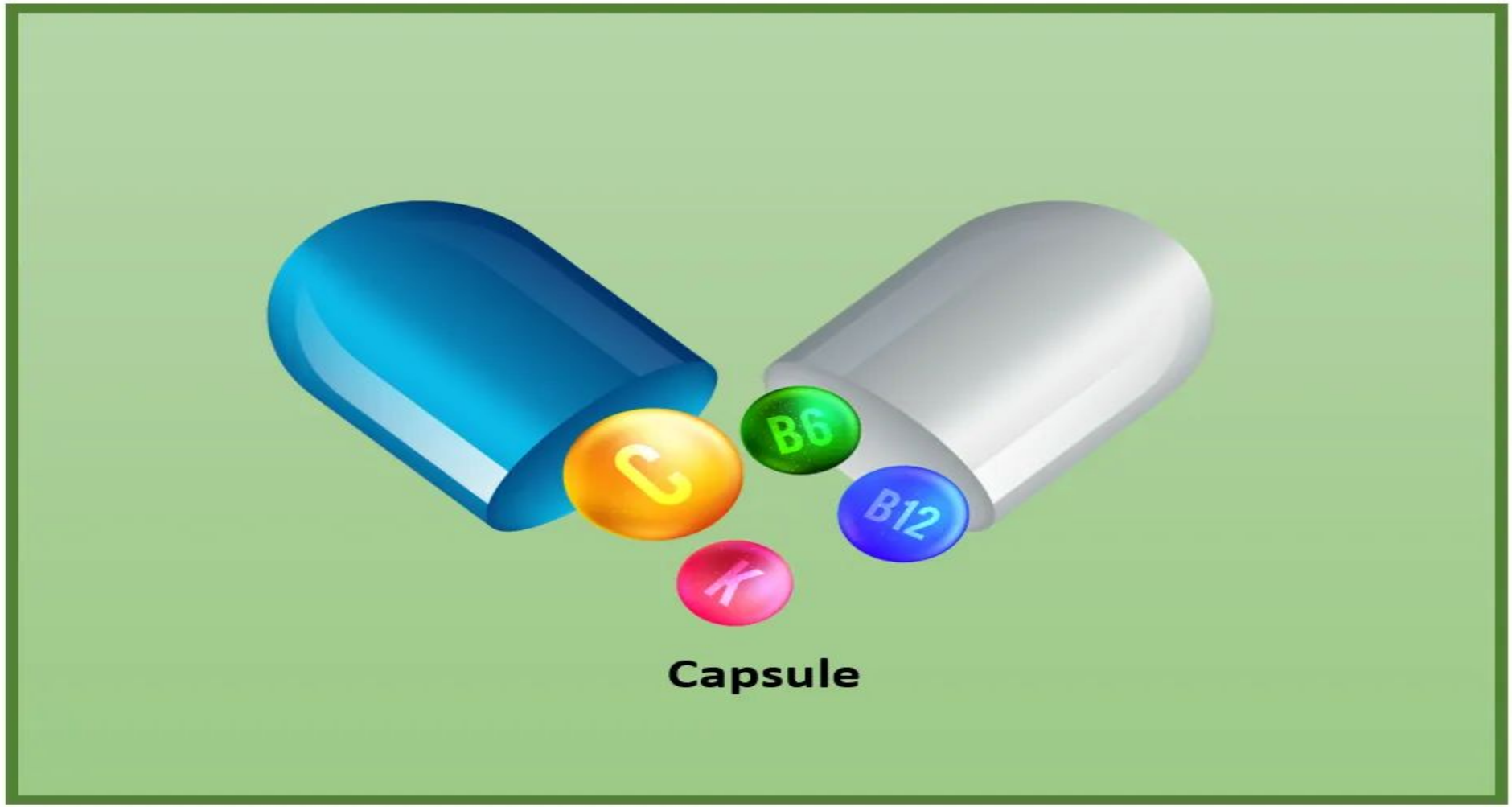


Fig: Encapsulation

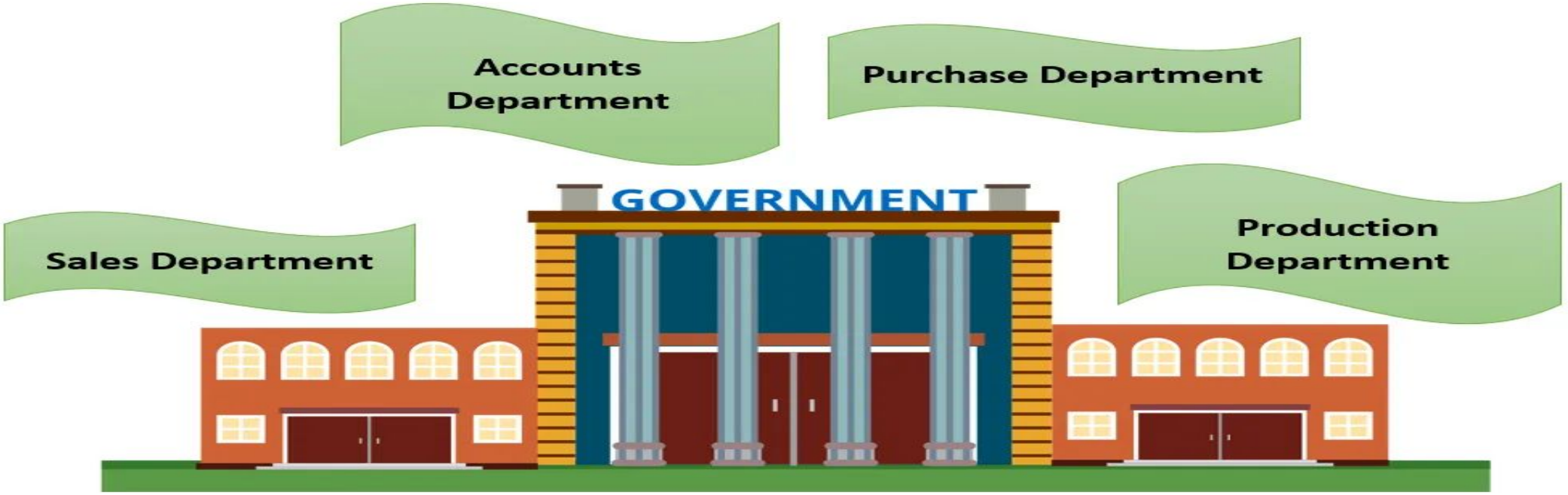
OOP Concepts: Encapsulation Examples

- The most commonly used example is the medical capsule. This capsule mixes few types of medicines and stored in one capsule.



OOP Concepts: Encapsulation Examples

- Another example for encapsulation is a large organization. An organization is consists of several departments like the production department, purchase department, sales department, and Accounts department. It combines all these departments together and had formed the organization.



OOP Concepts: Abstraction

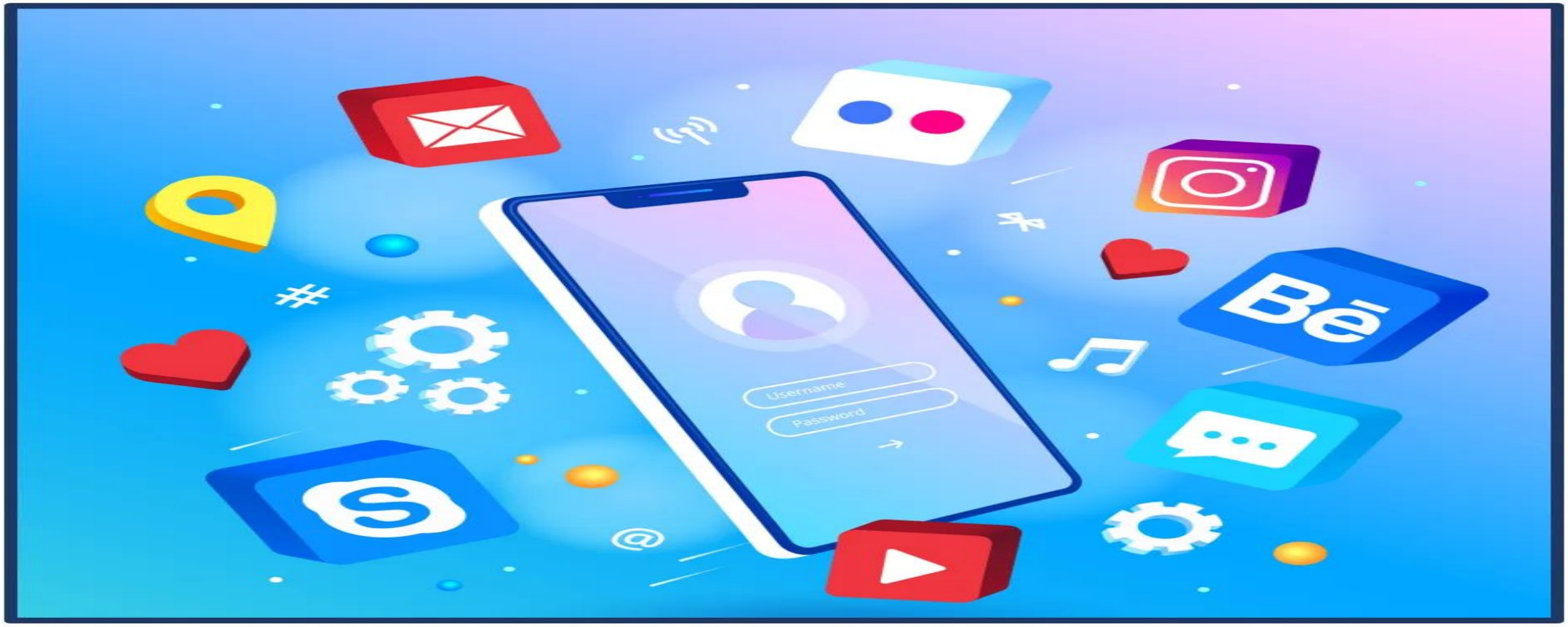
- In abstraction, it displays only the important information by hiding the implementation part.
- Abstraction refers to the practice of hiding implementation details and providing a simplified view of a system to its users.
  - It is used to simplify complex systems by exposing only the necessary features and behaviors, while hiding the underlying complexity.
- Abstraction is important because it helps to simplify code, making it easier to use, maintain, and extend.
  - By providing a simplified view of a system, abstraction helps to encapsulate complexity, making it easier to manage and work with.

OOP Concepts: Abstraction Examples

- Let's take the ATM machine. In an ATM machine, we can perform functions like withdraw cash, deposit cash, check balance, print bills, and so on. Even though it performs a lot of actions it doesn't show us the process. It has hidden its process by showing only the main things like getting inputs and giving the output.

OOP Concepts: Abstraction Examples

- **Mobile phones**. On a mobile phone, we can perform so many actions like making a call, sending messages, take pictures, download software and etc. We perform a lot of things but here also we don't know the inside process of these things. Which means the implementation parts are hidden.



References

1. https://www.guru99.com/java-oops-class-objects.html
2. https://techoreview.com/real-life-examples-to-learn-object-oriented-programming/
3. https://jeemariyana.medium.com/oop-concepts-with-real-world-examples-cda1cd277f4f
4. https://panditaarchit98.medium.com/know-everything-about-polymorphism-in-java-67ba0dd2804b
5. https://simplesnippets.tech/inheritance-in-java-types-of-inheritance/
6. https://www.scientecheasy.com/2020/07/encapsulation-in-java.html/
7.