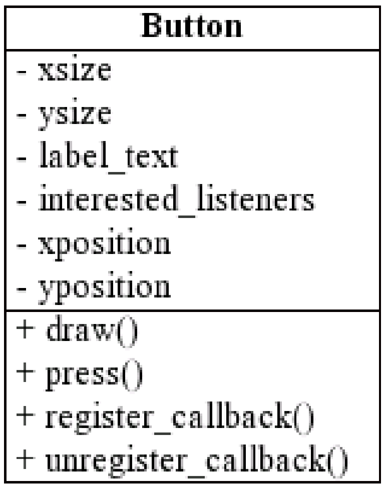
LEC2 OOP 2018-2019

<u>The diffrences between structured programming and OOP:</u>

Structured Programming Language
1) Follow top-down approach to program design.
2) Data and Functions don’t tide with each other.
3) Large programs are divided into smaller self contained program segment known as functions.
4) Data moves openly around the system from function to function.
5) Functions are dependent so reusability is not possible

Object Oriented Programming Language
1) Follow bottom-up approach in program design.
2) Functions and data are tied together.
3) Programs are divided into entity called Objects.
4) Data is hidden and can’t be accessed by the external world
5) Functions are not dependent so reusability is possible

Class (notation) Example



State and behavior are the basic properties of an Object.

State tells us about the type or the value of that object where as behavior tells us about the operations or things that the object can perform. For example, let’s say we have an Object called car, so car object will have color, engine type, wheels etc. as it’s state, this car object can run at 180kmph, it can turn right and left, it can go back and forth, it can carry 4 people etc. These are it’s **behaviors.**

In **object**-oriented programming, a **class** is a template **definition** of the method s and variable s in a particular kind of **object** . Thus, an **object** is a specific instance of a **class**; it contains real values instead of variables. The **class** is one of the **defining** ideas of **object**-oriented programming.

Classes and Objects

Classes and objects are the fundamental components of OOP's. Often there is a confusion between classes and objects. In this tutorial, we try to tell you the difference between class and object.

1- What is Class?

A class is an entity that determines how an object will behave and what the object will contain. In other words, it is a blueprint or a set of instruction to build a specific type of object.

Syntax

```
class  class_name{
   field;
   method;
 }
```

2- What is an Object?

An object is nothing but a self-contained component which consists of methods and properties to make a particular type of data useful. Object determines the behavior of the class. When you send a **message** (function call)to an object, you are asking the object to invoke or execute one of its methods. From a programming point of view, an object can be a data structure, a variable or a function. It has a memory location allocated.

Syntax : Student s = new Student();

What is the Difference Between Object & Class?

A **class** is a **blueprint or prototype** that defines the variables and the methods (functions) common to all objects of a certain kind. An **object** is a specimen of a class. Software objects are often used to model real-world objects you find in everyday life.

The differences between object and class:

Object	Class
Object is an instance of a class.	**Class is a blueprint or template from which objects are created.**
Object is a real world entity such as pen, laptop, mobile, bed, keyboard, mouse, chair etc.	**Class is a group of similar objects.**
Object is a physical entity.	**Class is a logical entity.**
Object is created through new keyword mainly e.g. Student s1=new Student();	**Class is declared using class keyword e.g. class Student{}**
Object is created many times as per requirement.	**Class is declared once.**
Object allocates memory when it is created.	**Class doesn't allocated memory when it is created.**
There are many ways to create object in java such as new keyword, newInstance() method, clone() method, factory method and deserialization.	**There is only one way to define class in java using class keyword.**

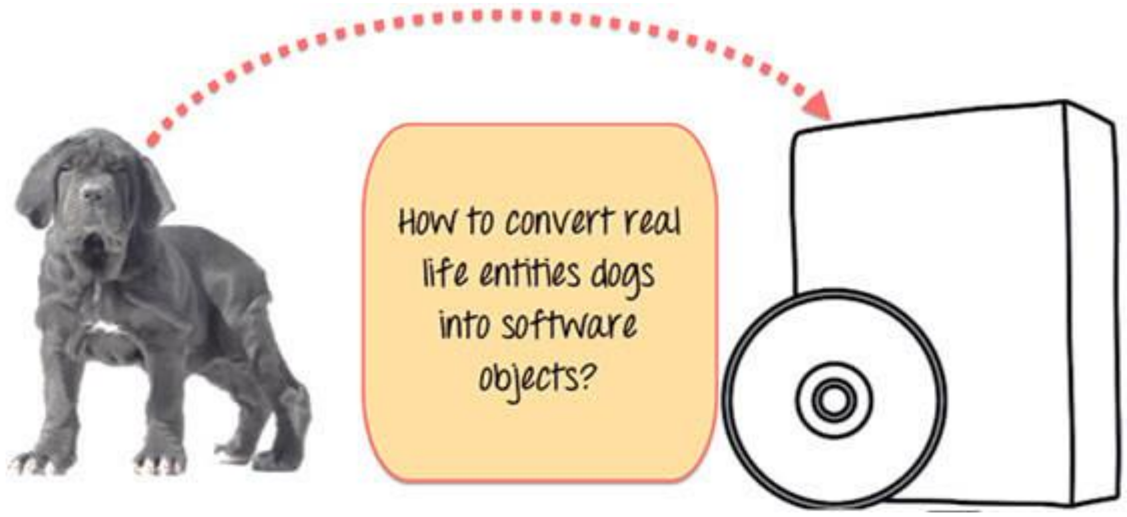
Let's see some real life example of class and object in java to understand the difference well:

Class: Human Object: Man, Woman
Class: Fruit Object: Apple, Banana, Mango, Guava wtc.
Class: Mobile phone Object: iPhone, Samsung, Moto
Class: Food Object: Pizza, Burger, Samosa

Understand the concept of Java Classes and Objects with an example.

Let's take an example of developing a pet management system, specially meant for dogs. You will need various information about the dogs like different breeds of the dogs, the age, size, etc.

You need to model real-life beings, i.e., dogs into software entities.


How to convert real life entities dogs into software objects?

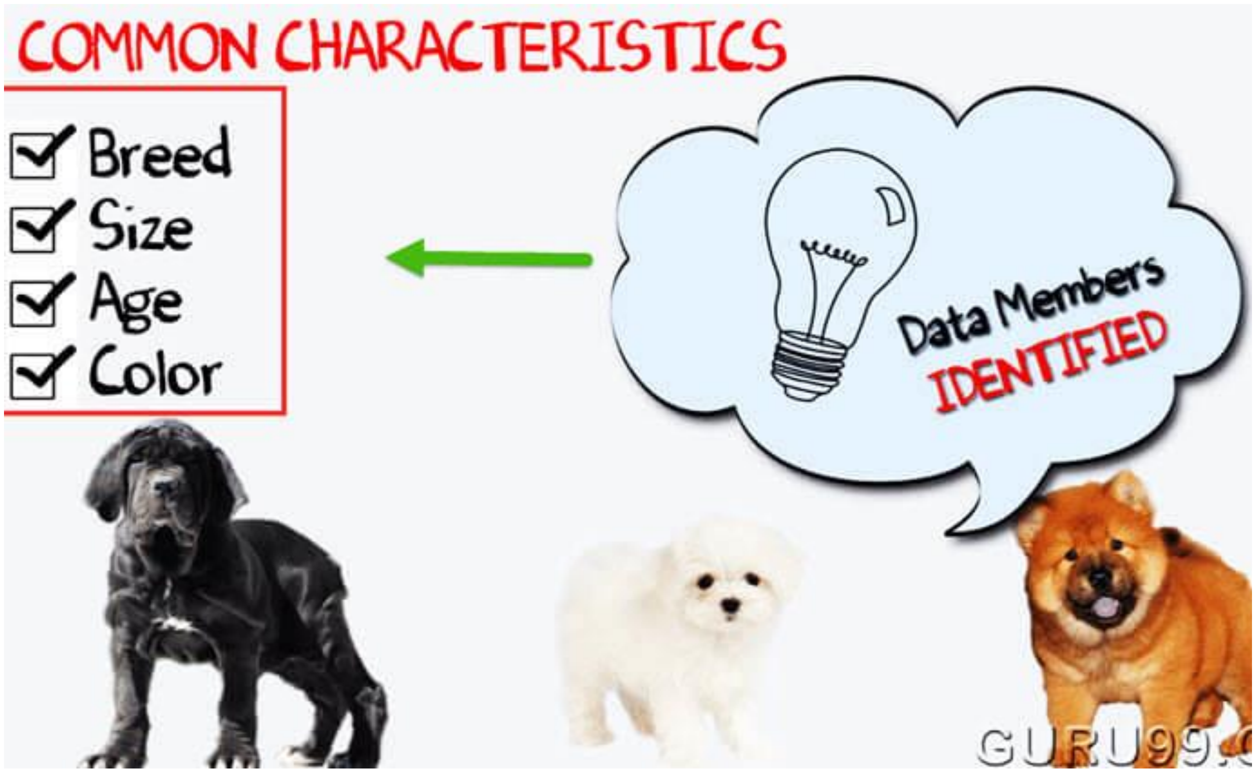
Moreover, the million dollar question is, how you design such software? **Here is the solution-**

First, let's do an exercise.

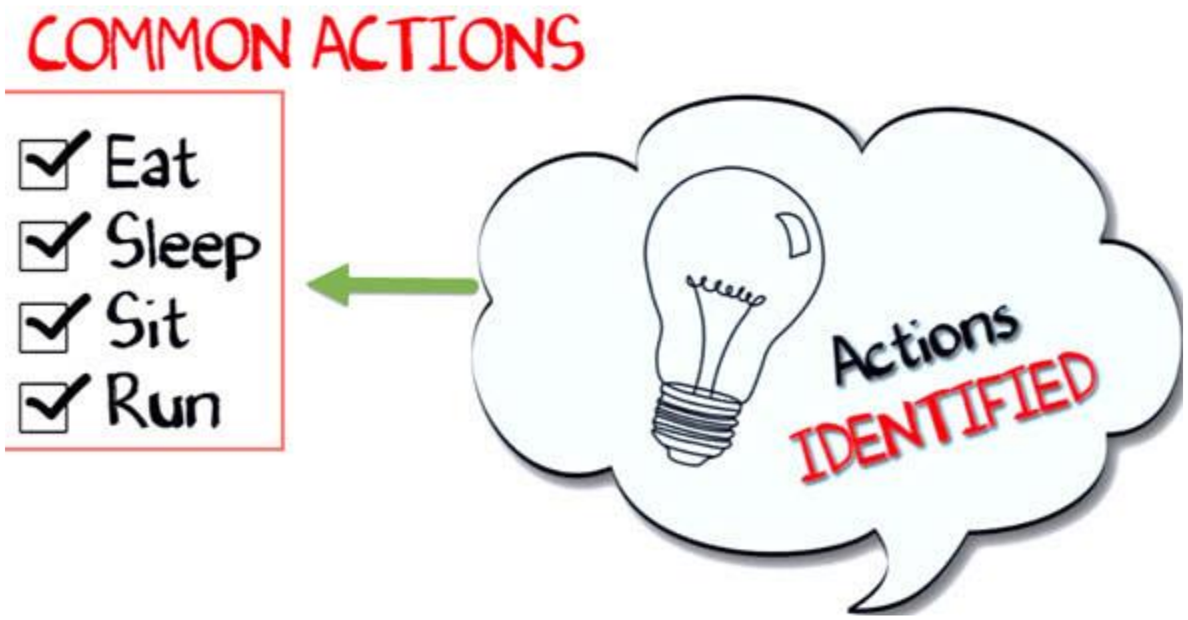
You can see the picture of three different breeds of dogs below.

Stop here right now! List down the differences between them.

Some of the differences you might have listed out maybe breed, age, size, color, etc. If you think for a minute, these differences are also some common characteristics shared by these dogs. These characteristics (breed, age, size, color) can form a data members for your object.

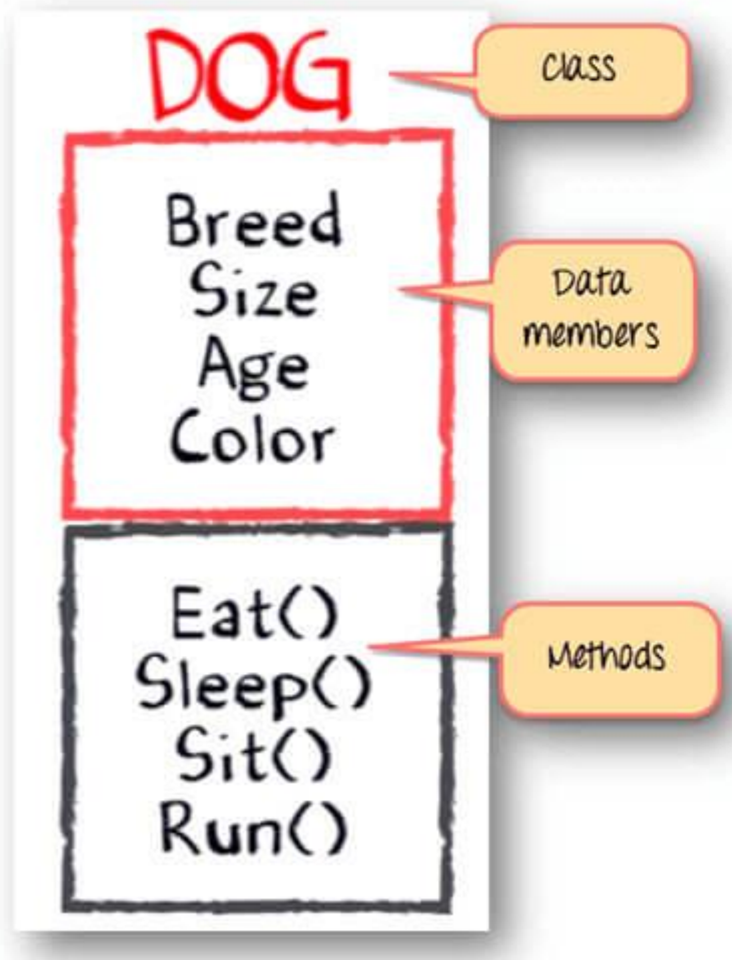


Next, list out the common behaviors of these dogs like sleep, sit, eat, etc. So these will be the actions of our software objects.

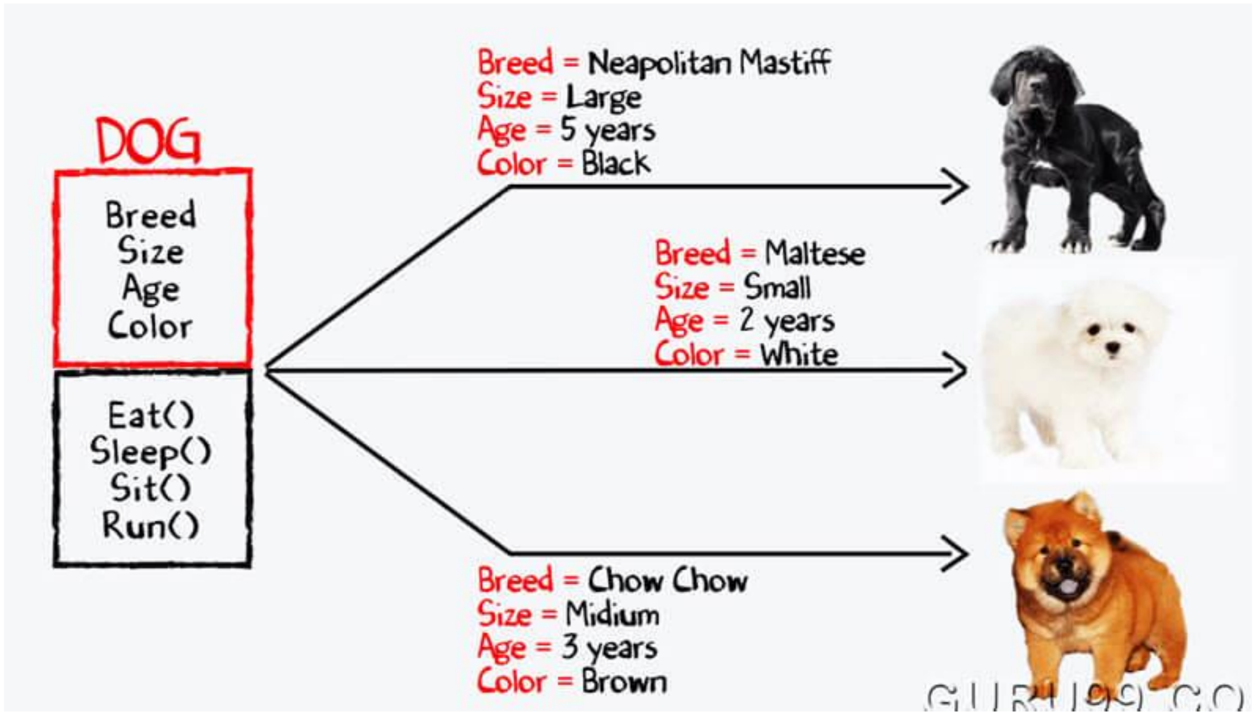


So far we have defined following things,

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



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



List of OOP Concepts in Java

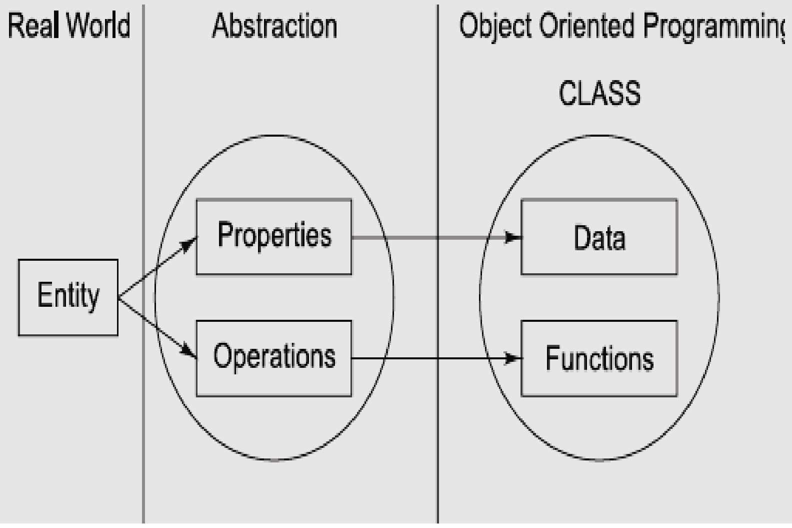
OOP concepts in Java are the main ideas behind Java's Object Oriented Programming. They are an **abstraction, encapsulation, inheritance, and polymorphism**.

- **Abstraction.** Abstraction means using simple things to represent complexity. We all know how to turn the TV on, but we don't need to know how it works to enjoy it. In Java, abstraction means simple things like **objects**, **classes**, and **variables** represent more complex underlying code and data. This is important because it lets avoid repeating the same work multiple times. The class is an **A**bstract **D**ata **T**ype (**ADT**).
- **Encapsulation.** This is the practice of keeping fields within a class private, then providing access to them via public methods. It's a protective barrier that keeps the data and code safe within the class itself. This way, we can re-use objects like code components or variables without allowing open access to the data system-wide.

  In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as **<u>data hiding</u>**. Declare the variables of a class as private.

- **Inheritance.** This is a special feature of Object Oriented Programming. It lets programmers create new classes that share some of the attributes of existing classes. This lets us build on previous work without reinventing the wheel. **<u>Reusability</u>** is the concept of inheritance provide the idea of reusability. This means that we can add additional features to an existing class without modifying it
- **Polymorphism.** This Java OOP concept lets programmers use the same word to mean different things in different contexts. One form of polymorphism in Java is **method overloading**. That's when different meanings are implied by the code itself. The other form is **method overriding**. That's when the different meanings are implied by the values of the supplied variables. See more on this below.

The concepts of object-oriented technology must be represented in object-oriented programming languages. Only then, complex problems can be solved in the same manner as they are solved in real-world situations. OOP languages use classes and objects for representing the concepts of abstraction and encapsulation. The mapping of abstraction to a program is shown in Fig. 1.18.


Real World
Abstraction
Object Oriented Programmin
CLASS
Entity
Properties
Operations
Data
Functions