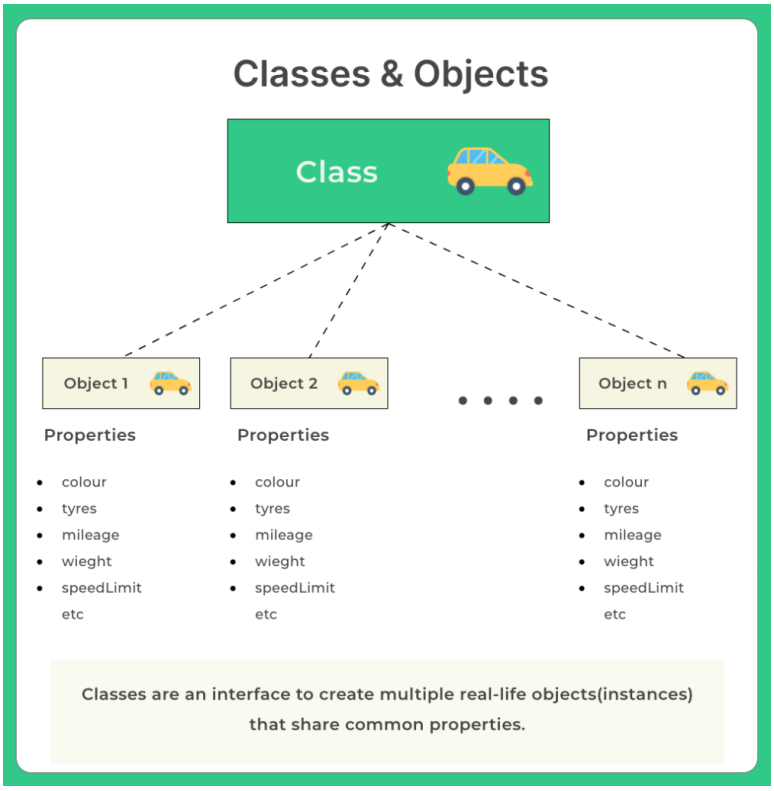
class in java

We know that java is an Object Oriented Programming Language. Whole java language functions on the concept of classes and objects. Classes can be defined as a blueprint for making an object. Let us understand what a class is through a real life example.

A car is a class. Swift DZire will be an object of the class car. It has attributes such as color, weight etc. and methods such as brake, drive, etc. It can be defined as a medium that allows the user to specify all attributes and methods that internally define the state and the behavior of the object.



Some features of class:

- Class can be imagined as a big data type, hence it's functionality is as good as a data type.
- Outer class should be either public or default so that it can be accessed by the JVM.
    - Public: public class
    - Default: class
- Class can have:
    - Methods
    - Variables
    - An object that can access these variables and methods.
- Class represents the common properties and actions of an object
- It has the following set of access modifiers that control the access or the visibility of a class:
    - **Public:** restricts the access to the class itself. Only methods that are part of the same class can access private members.
    - **Private:** restricts the access to the class itself. Only methods that are part of the same class can access private members.
    - **Protected:** allows the class itself and all its subclasses to access the member.

Java object :

A house, motorcycle, truck, notebook, sofa, or computer are examples of objects. An object is any item with state and activity. It could be mental or physical (tangible and intangible). An example of an intangible entity is the financial system.

More on Java Objects :

Everything in Java is built upon the objects and classes, as well as their traits and methods. A truck, for example, is an entity in the actual world. Along with features

like accelerate and break, the truck also has qualities like mass and colors.

Any kind of class can have an object as an instance.

Classes are logical constructs, whereas objects are the physical manifestations of a class, and these are the things that are actually taking up memory space.

Objects are categorized into three types:

1. **State:**

It simply means the value of the objects from its data type . In the real world every object holds a different set of values in some sort of manner. In other words .The states of a Java object are kept in fields that correspond to the various aspects of the object. For reference, a handgun with a seven clip in a third-person assault video game has eleven total states: one state for each bullet (e.g., nine, five, six, etc.), plus another state when the clip is empty.

2. **Identity :**

It is like whether one object is different from another or not. And to find the where the object is store in the memory or in other words we can say that The identity is just a feature used to identify that thing; examples include a random Id card or a memory address. Even while basic objects like a torch only have two values (power on and power off) and actions (power on, power off), they nonetheless have identities, such as the manufacture ID of that particular product.

3. **Behavior :**

It is most likely how our object behave on different set of input and aslo In order to enrich UIComponents with functionality not specifically described by the element implementation itself, behaviours are objects that are connected to UIComponents.

How to create a class :

o create a object of any class , we use the new keyword . This new keyword create the object at runtime.

Example :

```
Classname objectName = newKeyword classname();
```

```
class Boxers {
//.....variables...
//.....methods.....
}
public class Main
public static void main(String [] args){
//creating new object of class type Boxers
Boxers var = new Boxers();
}
}
```

Ways to Initialize an Object

1. Parent to parent reference.
2. Parent to child reference.
3. Child to child reference.

Code Example :

```
class Boxers {
//.....variables...
//.....methods.....
}
class RedBoxer extends Boxers{
//This class contains all the features of class Boxers.
}
public class Main{
public static void main(String [] args){
//parent to parent
Boxers var = new Boxers();
//parent to child
Boxers var1 = new RedBoxer();
```

```
//child to child
RedBoxer var2 = new RedBoxer();
}
}
```