UNIT-III (Class and Objects)

Class fundamentals: A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

1. **Modifiers** : A class can be public or has default access .
2. **Class name:** The name should begin with a initial letter (capitalized by convention).
3. **Superclass(if any):** The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
4. **Interfaces(if any):** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
5. **Body:** The class body surrounded by braces, { }.

Class fundamentals: एक वर्ग एक उपयोगकर्ता परिभाषित ब्लूप्रिंट या प्रोटोटाइप है जिसमें से ऑब्जेक्ट बनाए जाते हैं। यह उन गुणों या विधियों के समूह का प्रतिनिधित्व करता है जो एक प्रकार की सभी object के लिए सामान्य हैं।

सामान्य तौर पर, वर्ग घोषणाओं में ये घटक शामिल हो सकते हैं:

Modifiers: एक वर्ग public हो सकता है या उसकी डिफ़ॉल्ट पहुंच हो सकती है ।

Class का नाम: नाम एक प्रारंभिक पत्र (सम्मेलन द्वारा पूंजीकृत) से शुरू होना चाहिए।

SuperClass (यदि कोई हो): कक्षा के Parent (सुपरक्लास) का नाम, यदि कोई हो, तो कीवर्ड द्वारा पूर्ववर्ती। एक class केवल एक Parent का (उपवर्ग) विस्तार कर सकता है।

Interface (यदि कोई हो): कीवर्ड लागू होने से पहले वर्ग द्वारा लागू किए गए इंटरफेस की अल्पविराम से अलग सूची। एक वर्ग एक से अधिक इंटरफ़ेस लागू कर सकता है।

Body: Class body ब्रेसिज़ {} से घिरा ।

Syntax: <modifier> class <class name>

```
{
// Body of class
}
```

Object

It is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :

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

Example of an object : Dog

ऑब्जेक्ट

यह ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग की एक बुनियादी इकाई है और वास्तविक जीवन संस्थाओं का प्रतिनिधित्व करता है। एक सामान्य जावा प्रोग्राम कई ऑब्जेक्ट्स बनाता है, जो कि जैसा कि आप जानते हैं, इनवोकिंग methods द्वारा इंटरैक्ट करते हैं। एक object में निम्न शामिल हैं:

State: यह किसी object की विशेषताओं द्वारा दर्शाया जाता है। यह किसी object के गुणों को भी दर्शाता है।
Behavior: यह एक object के तरीकों से दर्शाया जाता है। यह अन्य objects के साथ किसी object की प्रतिक्रिया को भी दर्शाता है।
Identity: यह एक object को एक अनूठा नाम देता है और एक object को अन्य objects के साथ बातचीत करने में सक्षम बनाता है।
एक Object का उदाहरण: Dog

Declaring Objects (Also called instantiating a class)

When an object of a class is created, the class is said to be **instantiated**. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.

जब किसी वर्ग का ऑब्जेक्ट बनाया जाता है, तो क्लास को **instantiated** कहा जाता है। सभी instances कक्षा की विशेषताओं और व्यवहार को साझा करते हैं। लेकिन उन attributes के values, अर्थात् state प्रत्येक object के लिए unique हैं। एक single class में किसी भी संख्या के instances हो सकते हैं।

Example :

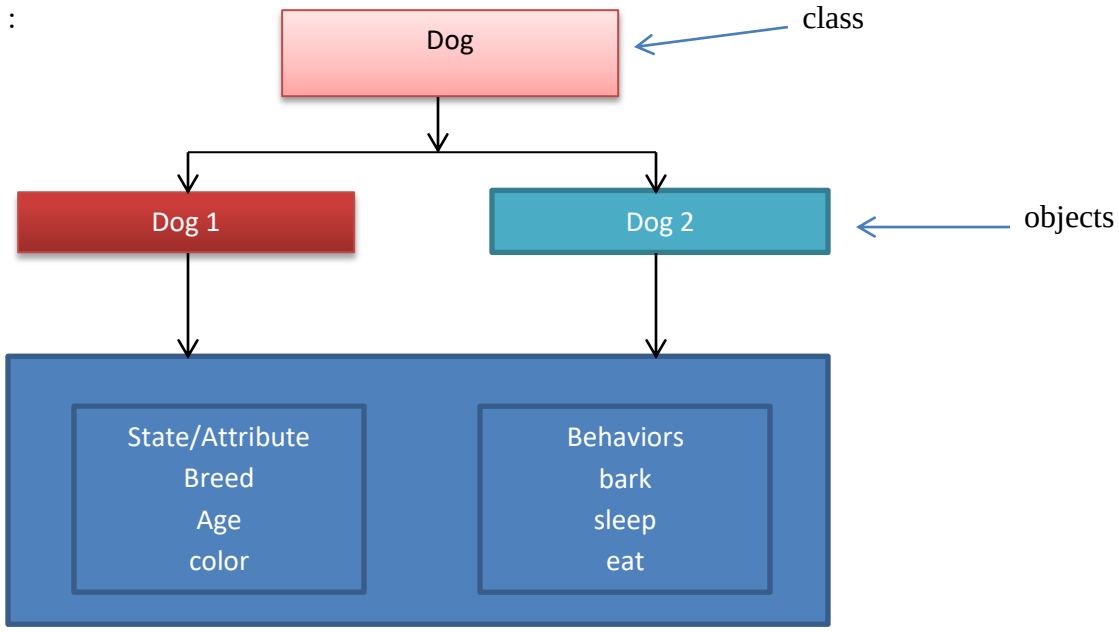
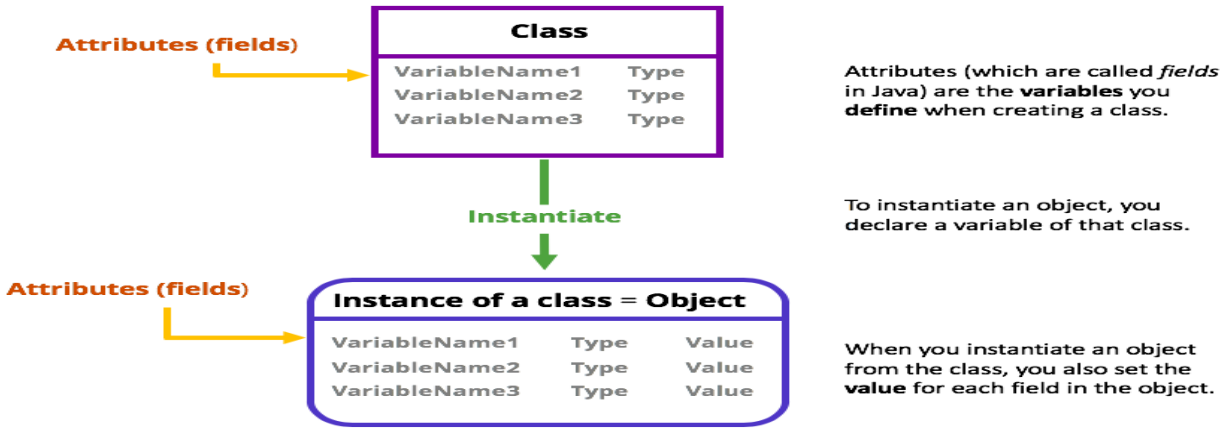


fig.: Shows the relaionship b/w class and object



For Example:

```
public class Dog
 {
   String breed;
   String size;
   int age;                    Instance Variables    //A variable declared inside the class but outside the body
   String color;                                      of the method, is called instance variable.

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


   public static void main(String[] args)
  {
      Dog d = new Dog();  <-------- Object Creation
      d.breed="Maltese";
      d.size="Small";
      d.age=2;                    Assignment of values
      d.color="white";
      System.out.println(d.getInfo());
   }
}                                                    Method Calling
```

Output:

```
Breed is: Maltese Size is:Small Age is:2 color is: white
```