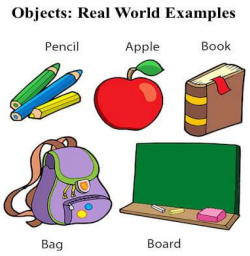
## UNIT II – Control Statements and Constructors

Control structures – Arrays - **Objects and classes: Classes** – Access Specifiers – methods and attributes - constructors: Default Constructor – Parameterized Constructor – Copy Constructor- Garbage collection.

### Objects and classes: Classes

**What is an object?**

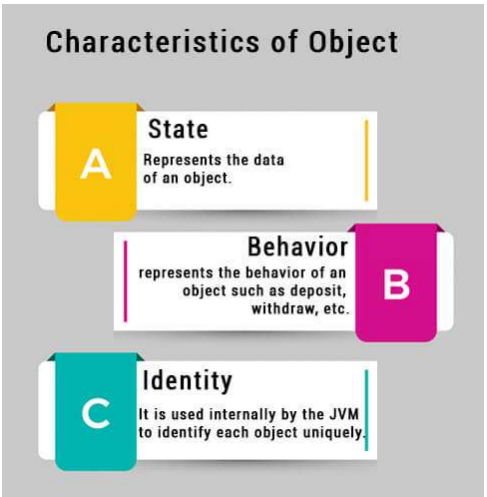
An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical **(tangible and intangible)**. The example of an intangible object is the banking system.




An object has 3 characteristics

- **State:** represents the data (value) of an object.
- **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
- **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

Pen is an object.
name is Reynolds;
color is white, known as its **state.**
It is used to write, so writing is its **behavior**



**An object is an instance of a class.** A class is a template or blueprint from which objects are created. So, an object is the instance of a class.

**Object Definitions:**

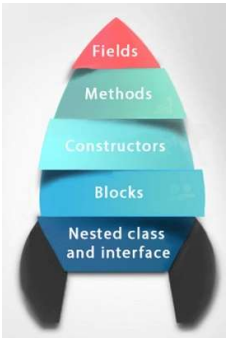
- An object is *a real-world entity*.
- An object is *a runtime entity*.
- The object is *an entity which has state and behavior*.
- The object is *an instance of a class*.

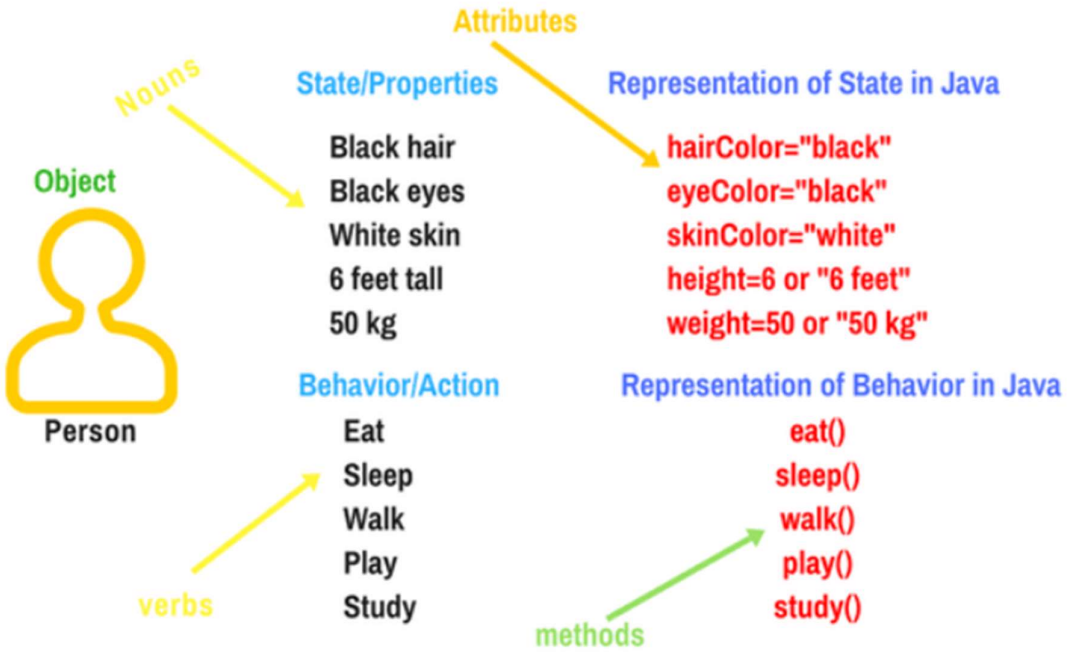
**What is a class?**

A class is a **group of objects which have common properties**. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

- **Fields**
- **Methods**
- **Constructors**
- **Blocks**
- **Nested class and interface**




**Syntax to declare a class:**

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

**Instance variable**

A variable which is created inside the class but outside the method is known as an instance variable. Instance variable doesn't get memory at compile time. It gets memory at runtime when an object or instance is created. That is why it is known as an instance variable.

**Method**

A method is like a function which is used to expose the behavior of an object.

Advantage of Method

- Code Reusability
- Code Optimization

**new keyword**

The new keyword is used to allocate memory at runtime. All objects get memory in Heap memory area.

## Object and Class Example: main within the class

***Write a java program to create a student class which has two data members id and name. And also create object of the Student class by new keyword and printing the object's value.***

```
class Student{
 //defining fields
 int id;//field or data member or instance variable
 String name;
 //creating main method inside the Student class
 public static void main(String args[]){
  //Creating an object or instance
  Student s1=new Student();//creating an object of Student
  //Printing values of the object
  System.out.println(s1.id); //accessing member through reference variable
  System.out.println(s1.name);
 }
}
```

**Output**

```
0
null
```

## Object and Class Example: main outside the class

```
class Student{
 int id;
 String name;
}
//Creating another class TestStudent1 which contains the main method
class TestStudent1{
 public static void main(String args[]){
  Student s1=new Student();
  System.out.println(s1.id);
  System.out.println(s1.name);
 }
}
```

**Output**

```
0
null
```

**3 Ways to initialize object**

There are 3 ways to initialize object in Java.

1. By reference variable
2. By method
3. By constructor

**Object and Class: Initialization through reference**

Initializing an object means storing data into the object. Let's see a simple example where we are going to initialize the object through a reference variable.

**Example**

```
class Student{
 int id;
 String name;
}
class TestStudent2{
 public static void main(String args[]){
  Student s1=new Student();
  s1.id=101;
  s1.name="Sonoo";
```

```
 System.out.println(s1.id+" "+s1.name) ;//printing members with a white space
 }
}
```

**Output**

*101 Sonoo*

Create multiple objects and store information in it through reference variable.

```
class Student{
 int id;
 String name;
}
class TestStudent3{
 public static void main(String args[]){
  //Creating objects
  Student s1=new Student();
  Student s2=new Student();
  //Initializing objects
  s1.id=101;
  s1.name="Sonoo";
  s2.id=102;
  s2.name="Amit";
  //Printing data
  System.out.println(s1.id+" "+s1.name);
  System.out.println(s2.id+" "+s2.name);
 }
}
```

**Output**

*101 Sonoo*

*102 Amit*

**Object and Class: Initialization through method**

creating the two objects of Student class and initializing the value to these objects by invoking the insertRecord method. Displaying the state (data) of the objects by invoking the displayInformation() method.

**Example**

```
class Student{
```

```
 int rollno;
 String name;
 void insertRecord(int r, String n){
  rollno=r;
  name=n;
 }
 void displayInformation(){System.out.println(rollno+" "+name);}
}
class TestStudent4{
 public static void main(String args[]){
  Student s1=new Student();
  Student s2=new Student();
  s1.insertRecord(111,"Karan");
  s2.insertRecord(222,"Aryan");
  s1.displayInformation();
  s2.displayInformation();
 }
}
```

**Output:**

111 Karan

222 Aryan