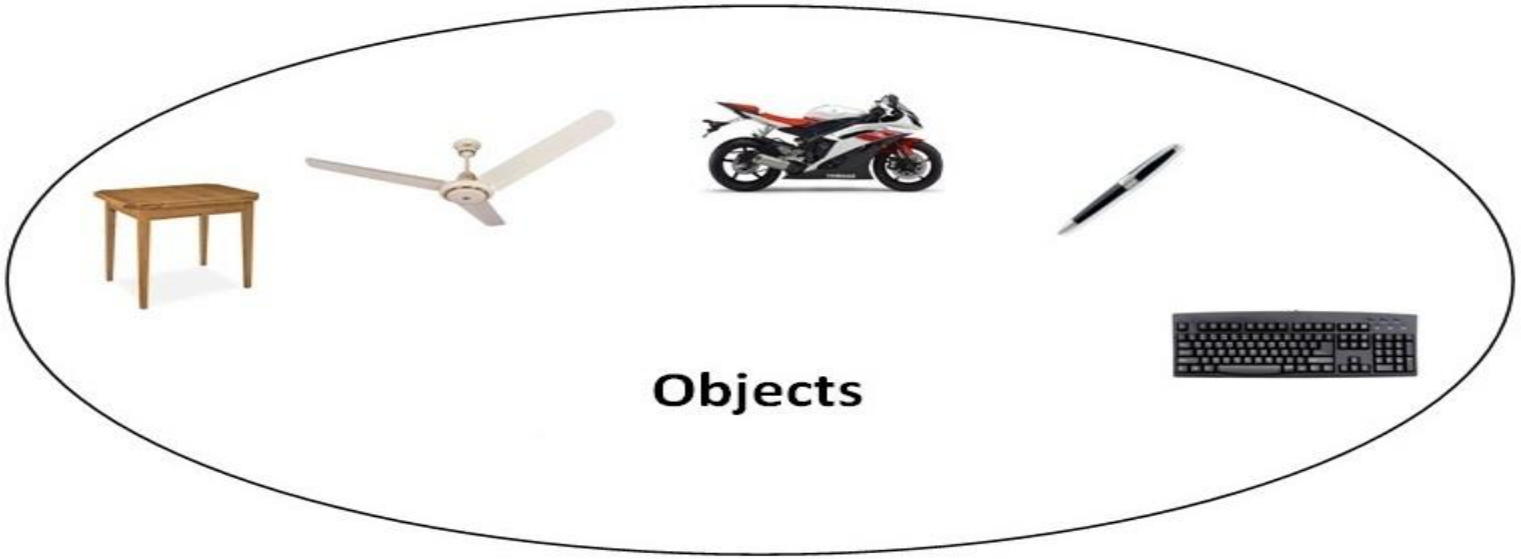
Java OOPs Concepts

- **Object** means a real word entity such as pen, chair, table etc.



- **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects.

OOPs Concepts

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

- Object
- Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc.
- It can be physical and logical.
- Class
- Collection of objects is called class.
- It is a logical entity.
- Inheritance
- When one object acquires all the properties and behaviors of parent object i.e. known as inheritance.
- It provides code reusability. It is used to achieve runtime polymorphism.

- Polymorphism
- When one task is performed by different ways i.e. known as polymorphism.
- For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.
- In java, we use method overloading and method overriding to achieve polymorphism.

- Abstraction
- Hiding internal details and showing functionality is known as abstraction.
- For example: phone call, we don't know the internal processing.
- In java, we use abstract class and interface to achieve abstraction.

- Encapsulation
- Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.


Capsule

- A java class is the example of encapsulation.

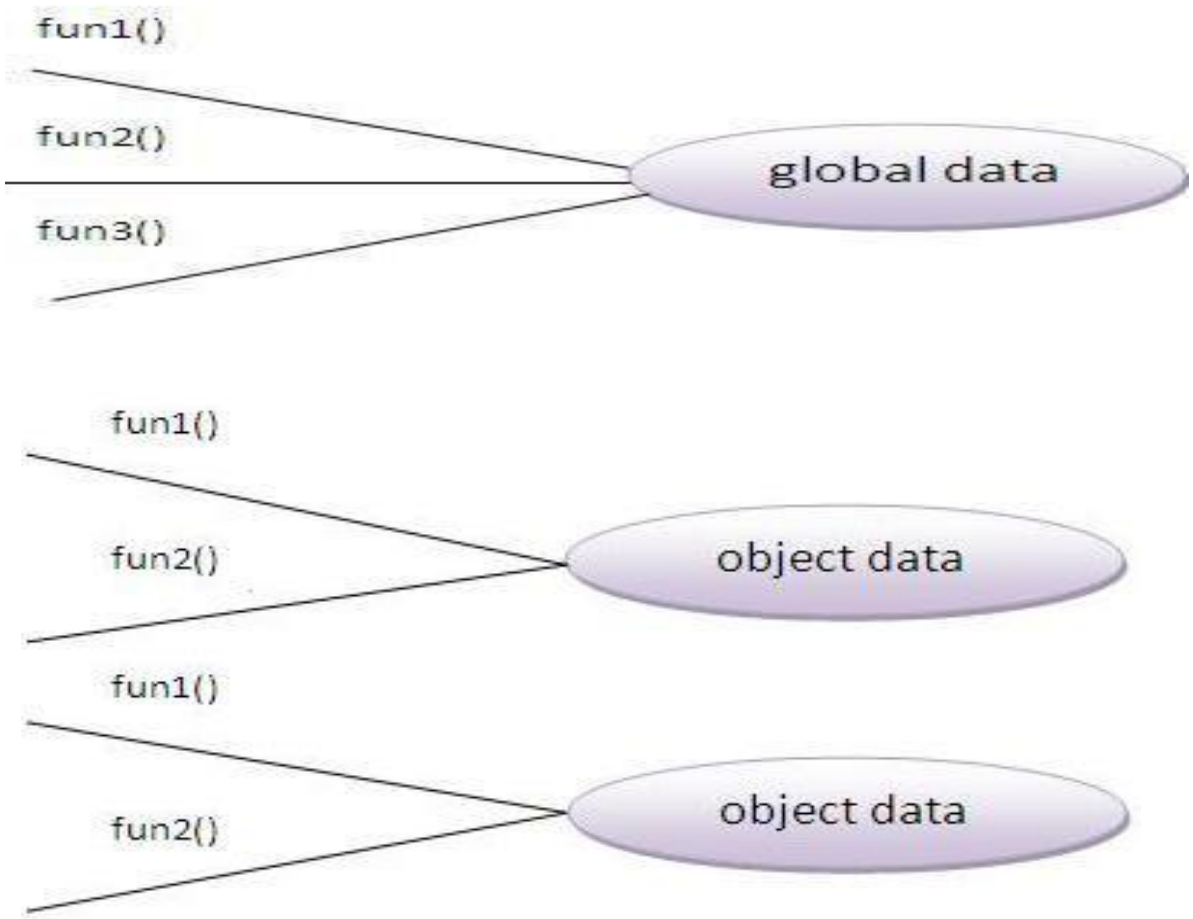
Advantages of OOPs over Procedure-oriented programming language

1)OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.

2)OOPs provides data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.

3)OOPs provides ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

Advantage of OOPs over Procedure-oriented programming language



- What is difference between object-oriented programming language and object-based programming language?

- Object based programming language follows all the features of OOPs except Inheritance.
- JavaScript and VBScript are examples of object based programming languages.
- Java is a pure object oriented programming language.

Java Naming conventions

- Java naming convention is a rule to follow as you decide what to name your identifiers such as class, package, variable, constant, method etc.
- But, it is not forced to follow. So, it is known as convention not rule.
- All the classes, interfaces, packages, methods and fields of java programming language are given according to java naming convention.
- Advantage of naming conventions in java
- By using standard Java naming conventions, you make your code easier to read for yourself and for other programmers.
- It indicates that less time is spent to figure out what the code does.

Naming conventions

If name is combined with two words, second word will start with uppercase letter always e.g. actionPerformed(), firstName, ActionEvent, ActionListener etc.

Name	Convention
class name	should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
interface name	should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc.
method name	should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
variable name	should start with lowercase letter e.g. firstName, orderNumber etc.
package name	should be in lowercase letter e.g. java, lang, sql, util etc.
constants name	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.

Object and Class in Java

- Object is the physical as well as logical entity whereas class is the logical entity only.
- An object has two characteristics:
- state: represents data (value) of an object.
- behavior: represents the behavior (functionality) of an object such as deposit, withdraw etc.
- For Example: Pen is an object. Its name is Reynolds, color is white etc. known as its state.
- It is used to write, so writing is its behavior.

- **Object is an instance of a class.**
- Class is a template or blueprint from which objects are created. So object is the instance(result) of a class.
- **Object Definitions:**
- Object is *a real world entity*.
- Object is *a run time entity*.
- Object is *an entity which has state and behavior*.
- Object is *an instance of a class*.

Class in Java

- 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>
{
    fields;(Instance Variables)
    methods;
}
```

Instance variable in Java

- A variable which is created inside the class but outside the method, is known as instance variable.
- Instance variable doesn't get memory at compile time.
- It gets memory at run time when object(instance) is created. That is why, it is known as instance variable.

Method in Java

- In java, a method is like function i.e. used to expose behavior of an object.

Advantage of Method

- Code Reusability
- Code Optimization

new keyword in Java

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

Object and Class Example: main within class

Student.java

```
class Student
{
 int id;//field or data member or instance variable
 String name;
 public static void main(String args[])
{
  Student s1=new Student();//creating an object of Student

  System.out.println(s1.id);//accessing member through ref
erence variable
  System.out.println(s1.name);
 }
}
```

Output
0
null

Object and Class Example: main outside class

TestStudent1.java

```
class Student
{
 int id;
 String name;
}
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

1) Object and Class Example: Initialization through reference

- Initializing object simply means storing data into object.

```
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

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

2) Object and Class Example: Initialization through method

```
class Student
{
 int rollno;
 String name;
 void insertRecord(int r, String n)
{
  rollno=r;
  name=n;
 }
 void displayInformation()
{
System.out.println(rollno+" "+name);
}
}
```

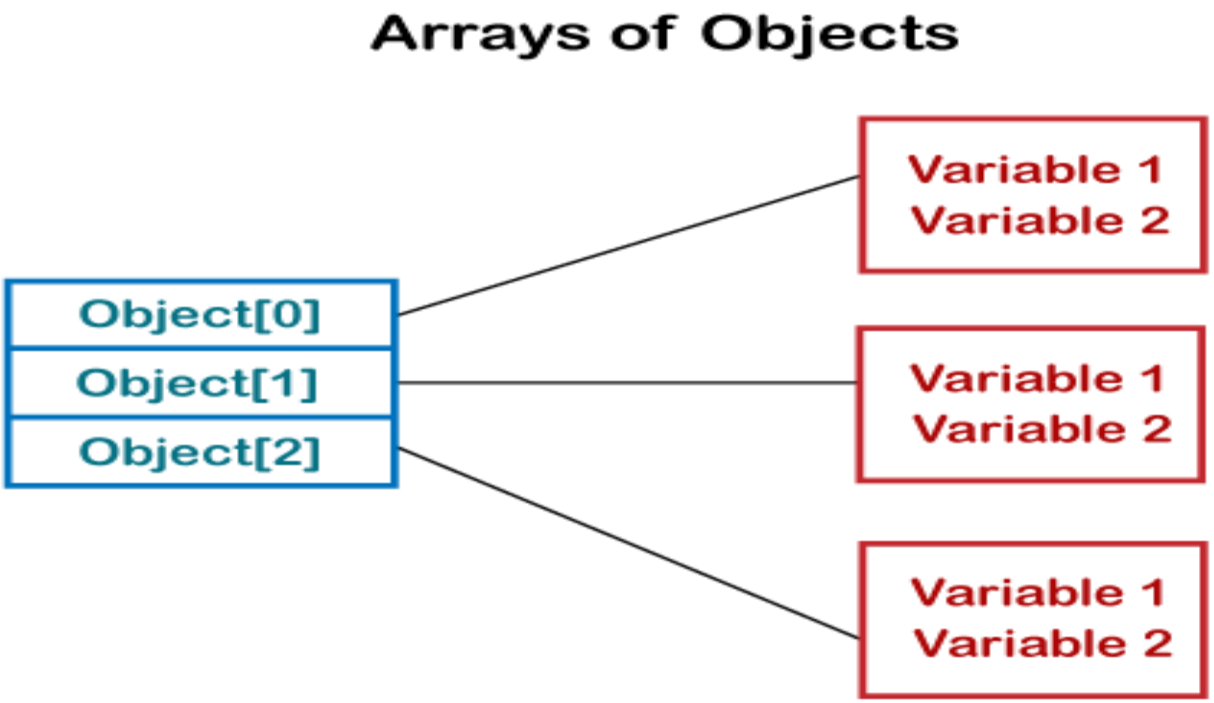
```
class TestStudent4
{
 public static void main(String args[])
{
  Student s1=new Student();
  Student s2=new Student();
  s1.insertRecord(1,"Kiran");
  s2.insertRecord(2,"Arya");
  s1.displayInformation();
  s2.displayInformation();
 }
}
```

1 Kiran

2 Arya

Array of Objects in Java

- Java allows us to store objects in an array.
- In Java, the class is also a user-defined data type.
- An array that contains **class type elements** are known as an **array of objects**. It stores the reference variable of the object.



Example

```
// Creating a Student clas with
// id and name as a attributes
class Student {

public int id;
public String name;

public void setData(int id, String name)
{
this.id = id;
this.name = name;
}

public void display()
{
System.out.println("Student id is: " + id + " "+
"and Student name is: "+ name);
System.out.println();
}
}
```

```
class ArrayObj {
        public static void main(String args[])
        {

        Student[] arr;
        arr = new Student[2];
        arr[0] = new Student();
        arr[1] = new Student();
        arr[0].setData(1, "SaiPavan");
        arr[1].setData(2, "Madhav");
        System.out.println("Student data in
        student arr 0: ");
        arr[0].display();


        System.out.println("Student data in
        student arr 1: ");
        arr[1].display();
        }
}
```

output

C:\WINDOWS\system32\cmd.exe

```
C:\Users\nhrao\Desktop\JAVAEG\Arrays>javac ArrayObj.java

C:\Users\nhrao\Desktop\JAVAEG\Arrays>java ArrayObj
Student data in student arr 0:
Student id is: 1 and Student name is: SaiPavan

Student data in student arr 1:
Student id is: 2 and Student name is: Madhav
```