Java Notes

### OOPs (Object-Oriented Programming System)

**Object** means a real-world entity such as a pen, chair, table, computer, watch, etc.

**Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

* [Object](https://www.javatpoint.com/object-and-class-in-java)
* Class
* [Inheritance](https://www.javatpoint.com/inheritance-in-java)
* [Polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java)
* [Abstraction](https://www.javatpoint.com/abstract-class-in-java)
* [Encapsulation](https://www.javatpoint.com/encapsulation)

### Object

Any entity that has state and behavior is known as an object. For example a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

An Object can be defined as an instance of a class.

An object contains an address and takes up some space in memory.

Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

**Example:** A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

### Class

## A class is defined as a blueprint from which you can create an individual object.

## Class doesn't consume any space.

### Inheritance

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

### Polymorphism

If one task is performed by different ways, it is known as polymorphism. For example: to convince the customer differently,

## 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.

### Encapsulation

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

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

## Advantage of OOPs over Procedure-oriented programming language

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

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

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

## What is the difference between an 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 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. These conventions are suggested by several Java communities such as Sun Microsystems and Netscape.

All the classes, interfaces, packages, methods and fields of Java programming language are given according to the Java naming convention. If you fail to follow these conventions, it may generate confusion or erroneous code.

## Advantage of naming conventions in java

By using standard Java naming conventions, you make your code easier to read for yourself and other programmers. Readability of Java program is very important. It indicates that less time is spent to figure out what the code does.

The following are the key rules that must be followed by every identifier:

* The name must not contain any white spaces.
* The name should not start with special characters like & (ampersand), $ (dollar), \_ (underscore).

Let's see some other rules that should be followed by identifiers.

### Class

* It should start with the uppercase letter.
* It should be a noun such as Color, Button, System, Thread, etc.
* Use appropriate words, instead of acronyms.

**Example:**

**public** **class** Employee

{

//code snippet

}

### Interface

* It should start with the uppercase letter.
* It should be an adjective such as Runnable, Remote, ActionListener.
* Use appropriate words, instead of acronyms.

**interface** Printable

{

//code snippet

}

### Method

* It should start with lowercase letter.
* It should be a verb such as main(), print(), println().
* If the name contains multiple words, start it with a lowercase letter followed by an uppercase letter such as actionPerformed().

**Example:-**

**class** Employee

{

//method

**void** draw()

{

//code snippet

}

}

### Variable

It should start with a lowercase letter such as id, name.

* It should not start with the special characters like & (ampersand), $ (dollar), \_ (underscore).
* If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter such as firstName, lastName.
* Avoid using one-character variables such as x, y, z.

**Example :-**

1. **class** Employee
2. {
3. //variable
4. **int** id;
5. //code snippet
6. }

### Package

* It should be a lowercase letter such as java, lang.
* If the name contains multiple words, it should be separated by dots (.) such as java.util, java.lang.

**Example :-**

**package** com.javatpoint; //package

**class** Employee

{

//code snippet

}

### Constant

* It should be in uppercase letters such as RED, YELLOW.
* If the name contains multiple words, it should be separated by an underscore(\_) such as MAX\_PRIORITY.
* It may contain digits but not as the first letter.

**Example :-**

**class** Employee

{

//constant

**static** **final** **int** MIN\_AGE = 18;

//code snippet

}