***Java OOPs Concepts***

## OOPs (Object Oriented Programming System)

**“Simula”** is considered as the first object-oriented programming language.

The programming paradigm where everything is represented as an object is known as truly **Object-oriented programming language**.

**“Smalltalk”** is considered as the first truly object-oriented programming language.

**If we want to develop any application we can develop by following any one of the following systems.**

1. **Procedure-oriented/Structured programming systems**

2. **Object-oriented programming systems**

**1. Procedure-oriented programming systems**

If we want to develop an application based on these procedure-oriented programming systems, we take the help of **functions or procedures**.

**Example:**  PASCAL, COBOL, FORTRAN, ADA, Visual Basic, Perl, 8086, C, Developer-2000...

In real time, we use this for developing software’s related to System Software and projects related to application software’s, but not projects related to internet software’s (distributed applications).

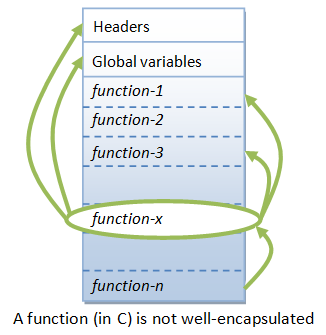
If we choose procedural oriented programming language for development of distributed applications then we get the following limitations.

1. Procedural oriented programming language possesses **platform dependency**.
2. In Procedural oriented programming language the data is visiting between client and server side applications in the form of plain text but not in the form of **Cipher** text.  
   **Note: Cipher** means “**a way of changing a message to keep it secret**”.
3. Procedural oriented programming language gives **poor security**.
4. In Procedural oriented programming language the data is visiting between client and server side applications in the form of byte by byte and results in poor communication.
5. In Procedural oriented programming language the data is available around the functions and leads to redundancy (repetition) problems.

**Drawbacks**

1. In this system we create multiple functions which will increase the code side so that maintenance will be complicated and debugging will become complicated.
2. In this system, we mainly concentrate on operations or functions but not on data which is very important for our application.
3. In this system data is open and globally available.
4. There is no data security in this system.
5. Functions cannot be expanded.

##### **Traditional Procedural-Oriented languages**



Traditional procedural-oriented programming languages (such as C, Fortran, Cobol and Pascal) suffer some notable drawbacks in creating reusable software components:

1. The procedural-oriented programs are made up of functions. Functions are less reusable. It is very difficult to copy a function from one program and reuse in another program because the function is likely to reference the global variables and other functions. In other words, functions are not well-encapsulated as a self-contained reusable unit.
2. The procedural languages are not suitable of high-level abstraction for solving real life problems. For example, C programs uses constructs such as if-else, for-loop, array, method, pointer, which are low-level and hard to abstract real problems such as a **Customer Relationship Management** ***(CRM)*** system or a computer soccer game.

The traditional procedural-languages separate the data structures (variables) and algorithms (functions).

In the early 1970s, the US Department of Defense (DoD) commissioned a task force to investigate why its IT budget always went out of control; but without much to show for. The findings are:

1. 80% of the budget went to the software (with the remaining 20% to the hardware).
2. More than 80% of the software budget went to maintenance (only the remaining 20% for new software development).
3. Hardware components could be applied to various products, and their integrity normally did not affect other products. (Hardware can share and reuse! Hardware faults are isolated!)
4. Software procedures were often non-sharable and not reusable. Software faults could affect other programs running in computers.

The task force proposed to make software behave like hardware OBJECT. Subsequently, DoD replaces over 450 computer languages, which were then used to build DoD systems, with an object-oriented language called Ada.

**2. Object-oriented programming systems**

If we want to develop an application based on this object-oriented programming systems we take the help of **objects and classes.**

This system can resolve the problems of previous system that is procedure oriented system.

The concepts of oops are very strong and suitable for any large and complicated projects.

Oops concepts are developed based on human life so that we can easily understand and implement in the application.

**Example**:

Java, c++, .net, small talk, Object PASCAL Object COBOL, Simula...

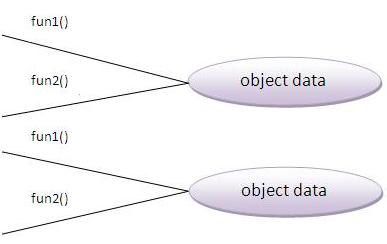
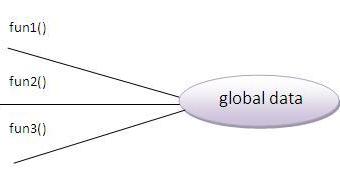
### Advantages of Object oriented programming language in development of distributed applications (Java)

### Object oriented programming language gives platform independency.

1. In Object oriented programming language applications the data is visiting between client and server side applications in the form of **Cipher text/ encrypted text**.
2. In Object oriented programming language we get **full security**.
3. In Object oriented programming language applications data can be visiting between client and server side applications all at once and results in effective communications
4. In Object oriented programming language application development is around the objects and solves the redundancy problems.

### Advantage 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. |
| 1. Oops provides data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere. |
| 1. 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. |



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

|  |
| --- |
| Ans. Object based programming language follows all the features of OOPs except **Inheritance**.  JavaScript and VBScript are examples of object based programming languages. **Object-Oriented Programming Languages** OOP_Objects.png  **Object-oriented programming (OOP) languages are designed to overcome these problems.**   1. The basic unit of OOP is a class, which encapsulates both the static properties and dynamic operations within a "box", and specifies the public interface for using these boxes. Since classes are well-encapsulated, it is easier to reuse these classes. In other words, OOP combines the data structures and algorithms of a software entity inside the same box. 2. OOP languages permit higher level of abstraction for solving real-life problems. The traditional procedural language (such as C and Pascal) forces you to think in terms of the structure of the computer (e.g. memory bits and bytes, array, decision, loop) rather than thinking in terms of the problem you are trying to solve. The OOP languages (such as Java, C++ and C#) let you think in the problem space, and use software objects to represent and abstract entities of the problem space to solve the problem.   OOP_SoccerGame.png  As an example, suppose you wish to write a computer soccer games (which I consider as a complex application). It is quite difficult to model the game in procedural-oriented languages. But using OOP languages, you can easily model the program accordingly to the "real things" appear in the soccer games.   * Player: attributes include name, number, location in the field, and etc; operations include run, jump, kick-the-ball, and etc. * Ball: * Reference: * Field: * Audience: * Weather:   Most importantly, some of these classes (such as Ball and Audience) can be reused in another application, e.g., computer basketball game, with little or no modification. Benefits of OOP The procedural-oriented languages focus on procedures, with function as the basic unit. You need to first figure out all the functions and then think about how to represent data.  The object-oriented languages focus on components that the user perceives, with objects as the basic unit. You figure out all the objects by putting all the data and operations that describe the user's interaction with the data.  **Object-Oriented technology has many benefits:**   * **Ease in software design** as you could think in the problem space rather than the machine's bits and bytes. You are dealing with high-level concepts and abstractions. Ease in design leads to more productive software development. * **Ease in software maintenance**: object-oriented software are easier to understand, therefore easier to test, debug, and maintain. * **Reusable software**: you don't need to keep re-inventing the wheels and re-write the same functions for different situations. The fastest and safest way of developing a new application is to reuse existing codes - fully tested and proven codes   **General Explanation:** |

**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. It simplifies the software development and maintenance by providing some concepts:

* **Object**
* **Class**
* **Inheritance**
* **Polymorphism**
* **Abstraction**
* **Encapsulation**

#### Object

Any entity that has state and behavior is known as an **object**.

**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.

**Example**:

To converse the customer differently, to draw something e.g. shape or rectangle etc.

In java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc.



#### Abstraction

**Hiding internal details and showing functionality** is known as abstraction.

**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.



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

# 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.

Java follows "**Hungarian notation**" in naming conventions.

**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. Readability of Java program is very important. It indicates that **less time** is spent to figure out what the code does.

|  |  |
| --- | --- |
| **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. |

###### **1. Convention for class Names**

The starting letter of each word, in a [class](http://way2java.com/oops-concepts/your-first-java-program/) name, should be of uppercase.

|  |  |
| --- | --- |
| **IDENTIFIER ACCEPTS** | **CONVENTION SAYS** |
| myfirsthouse | MyFirstHouse |
| Firsthouse | FirstHouse |
| House | House |
| Wishes | Wishes |
| numberformatexception | NumberFormatException |

###### **2. Convention for Variable Names**

The starting letter of the first word should be of lowercase and the starting letter of all the remaining words should be of uppercase.

|  |  |
| --- | --- |
| **IDENTIFIER ACCEPTS** | **CONVENTION SAYS** |
| Myfirsthouse | myFirstHouse |
| Firsthouse | firstHouse |
| House | house |
| Wishes | wishes |
| Lightgray | lightGray |
| Price | price |
| X | x |

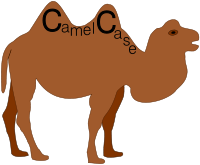
###### **3. Convention for Method Names**

Convention for [method](http://way2java.com/oops-concepts/using-variables-from-methods/) is simply that of variable only but followed by parenthesis.

|  |  |
| --- | --- |
| IDENTIFIER ACCEPTS | CONVENTION SAYS |
| myfirsthouse() | myFirstHouse() |
|  |  |
| firsthouse() | firstHouse() |
|  |  |
| house() | house() |
|  |  |
| display() | display() |
|  |  |
| indexof() | indexOf() |
|  |  |
| lastindexof() | lastIndexOf() |

**CamelCase in java naming conventions**

Java follows camel case syntax for naming the class, interface, method and variable.



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

###### **Three basic concepts of OOPS**

1. **Abstraction**
2. **Encapsulation**
3. **Polymorphism**

###### **1. Abstraction**

Abstraction means using the equipment (or code) without knowing the details of working.

**Example:** You are using your mobile phone without knowing how a mobile operates internally. Just a click on a button connects to your friend. This is **abstraction**. That is, the details of mobile mechanism are abstracted.

Similarly we use the code of somebody to get the functionality without knowing how the code is written or working.

**Example**, you are using printf() function to write to the DOS prompt without the awareness of what the code of printf(). One of the ways of achieving abstraction is [inheritance](http://way2java.com/oops-concepts/inheritance/). Through inheritance a subclass can use the super class methods as if they belong to it without caring their implementation details.

###### **2. Encapsulation**

Object-oriented concepts borrowed many terms from other technologies like **encapsulation** (from pharmaceuticals), **inheritance** (from biology), [**cloning**](http://way2java.com/java-lang/cloning-%e2%80%93-duplicating-an-object-marker-interface/) (from genetics) and [**polymorphism**](http://way2java.com/oops-concepts/using-methods-and-method-overloading/)(from biology) etc.

Placing a powdered drug in a gelatin capsule and sealing it is known as **encapsulation**. With encapsulation, a Pharmacist hides the properties of a drug like its taste and color from the patient.

Similar meaning is in OOPs also. Encapsulation hides the implementation details of coding; other way it is abstraction. With abstraction, implementation of information is hidden.

In a programming language, variables represent the properties and methods are used to change the properties.

**For example**, the **speed** variable represent the property of a motor car and the method **accelerator()** is used to change the speed.

Objects are used to call the methods. There may be multiple motor cars and every car has its own speed. Here, motor car represents an object. Every object encapsulates its own data. This [encapsulation](http://way2java.com/oops-concepts/three-great-principles-%e2%80%93-data-binding-data-hiding-encapsulation/) concept takes OOP languages a lead over traditional procedure-oriented languages. **Binding data with objects (generally through method calls) is known as encapsulation**.

In encapsulation, to have control over the manipulation of data (not to feed wrong data, for example, the speed cannot be negative) by other classes, a programmer declares **variables as private** and **methods as public**. Other classes can **access the private variables through public methods**. With encapsulation, every object maintains its own data and this data is entirely private to that object. Other objects cannot access or modify the data.

###### **3. Polymorphism**

**Polymorphism** is a Greek term and means many forms of the same ("poly" means many and "morphism" means forms).

It is an OOP paradigm where **one method can be made to give different outputs (functionalities) when called at different times**.

Polymorphism is two ways –[**static polymorphism**](http://way2java.com/oops-concepts/using-methods-and-method-overloading/) where methods are binded at compile time and [**dynamic polymorphism**](http://way2java.com/oops-concepts/dynamic-polymorphism/)where methods are binded dynamically at runtime. The same person is called as an officer (in office), husband (in house) and player (in cricket team). The person can be treated as base class. Extra subclasses can be added by [hierarchical inheritance](http://way2java.com/oops-concepts/types-of-inheritance/) like son etc.

In real life, a woman is polymorphic. For a son, she is mother, for a husband she is wife, for a father she is daughter etc. The same woman does different jobs with different people. This we called as polymorphism. Same way, the same method can do different jobs like calculating triangle area, rectangle area etc,

###### **What is a class in Java?**

**The basic entity of Java coding is a "class".**   
A class is delimited by a pair of braces, { and }. All the code resides within a class. The name of the class is the name with which the whole code is known. An [object](http://way2java.com/oops-concepts/reference-variables-anonymous-objects/) is a handle to manipulate the whole code of the class. A class can comprise of [constructors](http://way2java.com/oops-concepts/constructors-and-constructor-overloading/), [variables](http://way2java.com/oops-concepts/usning-local-and-instance-variables/), [methods](http://way2java.com/oops-concepts/using-variables-from-methods/) and some blocks of code (like [static blocks](http://way2java.com/java-introduction/static-blocks-%e2%80%93-static-initialization/)) etc.

|  |  |
| --- | --- |
|  | *import java.lang.\*;*  *public class Wishes{*  *public static void main(String args[]){*  *System.out.println("Hello World");*  *System.out.println("Best wishes");*  *}*  *}* |

**Let us discuss some coding details of the above simple program.**

*import java.lang.\*;*

"**import**" is a keyword of Java equivalent to "include" of C/C++. Here, a small difference exists. "**java.lang**" is known as package. A [package](http://way2java.com/java-lang/introduction-%e2%80%93-java-lang-package/) is equivalent to a header file of C-lang. A package contains classes and methods exist in the classes. But, a header file contains directly functions. Java classes (containing methods) are distributed through packages and sub packages. "lang" is the sub package of package "java". Distributing classes through packages has got an advantage over traditional header files.

**Note:** "java.lang" package is implicitly imported if the programmer does not import himself.

*public class Wishes*

In the above statement, "[**public**](http://way2java.com/packages/access-specifiers-accessibility-permissions-restrictions/)" is a keyword and denotes an access specifier. "Public" means anybody can access the code without any restrictions. Java includes other [access Specifiers](http://way2java.com/oops-concepts/access-specifiers-access-modifiers/) also. The whole program is known by the name "class Wishes". Again "**class**" is a keyword. Observe, the class is delimited by two braces, {and}. Java demands that all the code you write, may spread thousands of lines, and should be within the class declaration only.

[*public static void main(String args[])*](http://way2java.com/oops-concepts/public-static-void-mainstring-args/)

The **main()** method comes with many decorative paraphernalia and each one conveys some meaning to the compiler. The "[**static**](http://way2java.com/oops-concepts/static-keyword-%e2%80%93-philosophy/)" means the main() method can be called without the help of an object. "**void**" keyword means the main() method does not return a value. The main() takes a String array as parameter and is used later to access [command-line](http://way2java.com/arrays/command-line-arguments/) arguments.

The **[System.out.println()](http://way2java.com/io/system-out-println/)** prints the data at the DOS prompt (equivalent to printf()). [println()](http://way2java.com/io/printstream/)method gives an implicit "\n" character. "**in**" in println() stands for new line. The output comes in two different lines.

**When main method terminates it return control to whom?**

1. To JVM.

**Below statement will work smoothly, it means in the above statement, int is a return type that returning 0 value, that means main method can have return type.**

*public static int main(string[] args){  
 return 0;  
}*

1. This type of main method is treated as your own method like you write display() method. But doesn’t work as a callback method. If you write this main without args[] main, your program does not execute.

**How it is possible in the bellow statement.  
public static void main(String args[])  
in the above statement void is a return type, we can also declare int as return type. It will work smoothly.**

1. void is not return type. A method/function should have three constraints to be satisfied.

1. There should be a return type. void indicates that the method does not have return a value. The return types can be objects or primitive data types.  
2. There should be parenthesis; atleast empty parenthesis.  
3. There should body; atleast empty body.

**In the command prompt when we execute the java code with the help of java interpreter we only specify the name of the class within which the “main” method is defined and do not explicitly call the “main” method with (.) operator. Then how the main method is invoked automatically?**

1. Suppose if you execute like this:

c:\> java Demo

automatically it becomes

c:\> java Demo.class

then Demo.class is loaded.

Instead, if you type yourself as

c:\> java Demo.class

then Java adds automatically

c:\> java Demo.class.class

then such file does not exist in you directory, thereby it gives error message.

Java designer makes many things simple to use. Purposefully they avoided .class to put in the execution command, just to make your typing simple

**The main function takes argument of String type of array, but at the time of specifying the class file to “java” interpreter we do not pass the any kind value and still it runs smoothly. How is it possible that the program executes without passing any argument, as in case of normal methods we need to pass the arguments (if defined).**

1. Along with your execution command, as no arguments are passed, the String array size becomes zero by default. It is not error, why, to program your execution simple as many times we do not pass arguments, especially in real-time applications.