### HowToDoInJava

Java 8

Regex

Concurrency

**Best Practices** 

**Spring Boot** 

JUnit5

**Interview Questions** 



## Object Oriented Principles

OOPs - Introduction

OOPs - Access Modifiers

OOPs - Constructors

OOPs - Instance Initializers

OOPs - Abstraction

OOPs - Encapsulation

OOPs – Inheritance

OOPs - Polymorphism

OOPs – Overloading vs

Overriding

OOPs - Interface vs.

**Abstract Class** 

OOPs - extends vs

implements

OOPs – instanceof operator

OOPs - Multiple

Inheritance

Association, Aggregation

and Composition

#### **Popular Tutorials**

Java 8 Tutorial

Core Java Tutorial

# Java OOPs Concepts – Object Oriented Principles

By Lokesh Gupta | Filed Under: Object Oriented Principles

In this Java OOPs concepts tutorial, we will learn four major object oriented principles – abstraction, encapsulation, inheritance, and polymorphism. They are also known as four pillars of the object oriented programming paradigm.

- 1. **Abstraction** is the process of exposing the essential details of an entity, while ignoring the irrelevant details, to reduce the complexity for the users.
- 2. *Encapsulation* is the process of bundling data and operations on the data together in an entity.
- 3. *Inheritance* is used to derive a new type from an existing type, thereby establishing a parent-child relationship.
- 4. *Polymorphism* lets an entity take on different meanings in different contexts.

#### **Search Tutorials**

Type and Press ENTER...

Collections in Java
Java Concurrency
Spring Boot Tutorial
Spring AOP Tutorial
Spring MVC Tutorial
Spring Security Tutorial
Hibernate Tutorial
Python Tutorial
Jersey Tutorial
Maven Tutorial
Log4j Tutorial

Regex Tutorial

#### **Table of Contents**

- 1. Abstraction
- 2. Encapsulation
- 3. Inheritance
- 4. Polymorphism

## 1. Abstraction

Abstraction in OOPs is very easy to understand when you relate it to the real time example. For example, when you drive your car you do not have to be concerned with the exact internal working of your car. What you are concerned with is interacting with your car via its interfaces like steering wheel, brake pedal, accelerator pedal etc. Here the knowledge you have of your car is abstract.

In computer science, abstraction is the process by which data and programs are defined with a representation similar in form to its meaning (*semantics*) while hiding away the implementation details.

In more simple terms, abstraction is to hide information that is not relevant to context or rather show only relevant information and to simplify it by comparing it to something similar in the real world.



Abstraction captures only those details about an object that is relevant to the current perspective.

Typically abstraction can be seen in two ways:

#### 1. Data abstraction

Data abstraction is the way to create complex data types from multiple smaller data types – which is more close to real life entity. e.g. An <a href="Employee">Employee</a> class can be a complex object of having various small associations.

```
public class Employee
{
    private Department department;
    private Address address;
    private Education education;
    //So on...
}
```

So, if you want to fetch information of a employee, you ask it from Employee object – as you do in real life, ask the person itself.

#### 2. Control abstraction

Control abstraction is achieved by hiding the sequence of actions for a complex task – inside a simple method call, so logic to perform the task can be hidden from the client and could be changed in future without impacting the client code.

```
public class EmployeeManager
{
    public Address getPrefferedAddress(Employee e)
    {
        //Get all addresses from database
        //Apply logic to determine which address is preferred
        //Return address
    }
}
```

In above example, tomorrow if you want to change the logic so that everytime domestic address is always the preferred address, you will change the logic inside <code>getPrefferedAddress()</code> method, and client will be unaffected.

Read More: Understanding abstraction in java

## 2. Encapsulation

Wrapping data and methods within classes in combination with implementation hiding (through access control) is often called encapsulation in OOPs. The result is a data type with characteristics and behaviors. Encapsulation essentially has both i.e. information hiding and implementation hiding.

"Whatever changes, encapsulate it" – A famous design principle

**Information hiding** is done through using access control modifiers (public, private, protected) and implementation hiding is achieved through creation of interface for a class.

**Implementation hiding** gives the designer the freedom to modify how the responsibility is fulfilled by an object. This is especially valuable at points where the design (or even the requirements) are likely to change.

Let's take an example to make it more clear.

#### 2.1. Information hiding

```
class InformationHiding
{
    //Restrict direct access to inward data
    private ArrayList items = new ArrayList();

    //Provide a way to access data - internal logic can safely be chapublic ArrayList getItems(){
        return items;
    }
}
```

#### 2.2. Implementation hiding

```
interface ImplementationHiding {
   Integer sumAllItems(ArrayList items);
class InformationHiding implements ImplemenatationHiding
    //Restrict direct access to inward data
    private ArrayList items = new ArrayList();
   //Provide a way to access data - internal logic can safely be cha
   public ArrayList getItems(){
        return items;
    public Integer sumAllItems(ArrayList items) {
       //Here you may do N number of things in any sequence
       //Which you do not want your clients to know
       //You can change the sequence or even whole logic
       //without affecting the client
```

Read More: Encapsulation In Java

## 3. Inheritance

Inheritance is another important concept in object oriented programming. Inheritance in Java is a mechanism by which one object acquires the properties and behaviors of the parent object. It's essentially creating a parent-child relationship between classes. In Java, you will use inheritance mainly for code re-usability and maintainability.

Keyword "extends" is used to inherit a class in java. The "extends" keyword indicates that you are making a new class that derives from an existing class. In the terminology of Java, a class that is inherited is called a super class. The new class is called a subclass.

A subclass inherits all the non-private members (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.

e.g.

```
class Employee
{
    private Department department;
    private Address address;
    private Education education;
    //So on...
}
class Manager extends Employee {
    private List<Employee> reportees;
}
```

In above example, Manager is specialized version of Employee and reuses department, address and education from Employee class as well as define it's own reportees list.

## 4. Polymorphism

**Polymorphism** is the ability by which, we can create functions or reference variables which behaves differently in different programmatic context.

In java language, polymorphism is essentially considered into two versions:

- Compile time polymorphism (static binding or **method overloading**)
- Runtime polymorphism (dynamic binding or **method overriding**)

Read More: Polymorphism in java

Above are four Java OOPs concepts, and I will suggest you to develop a good understanding of each one of it.

Happy Learning!!

References:

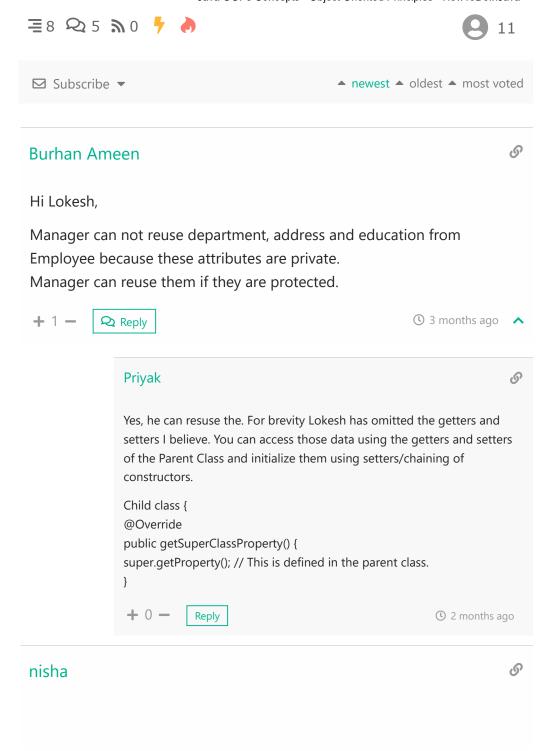
https://docs.oracle.com/javase/tutorial/java/concepts/ http://c2.com/cgi/wiki?InformationHiding

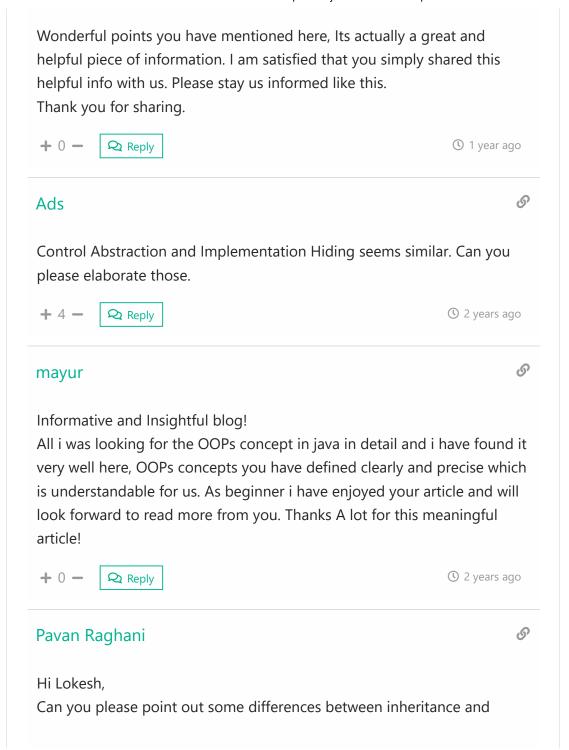
## **About Lokesh Gupta**

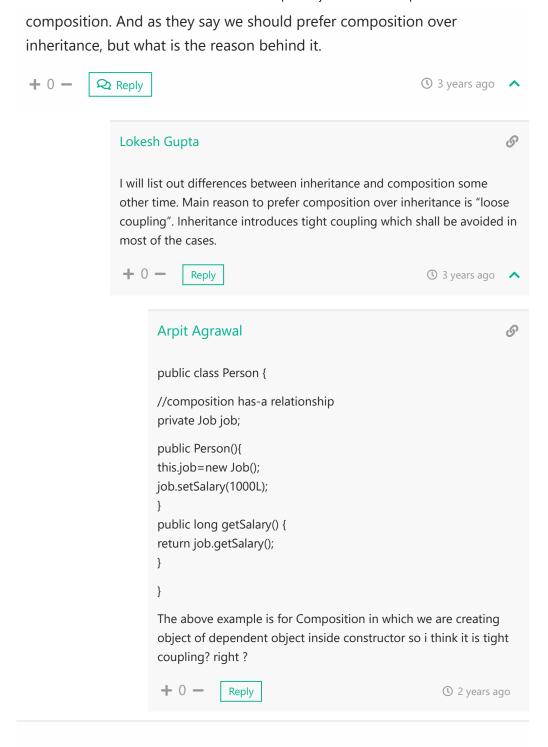
A family guy with fun loving nature. Love computers, programming and solving everyday problems. Find me on Facebook and Twitter.

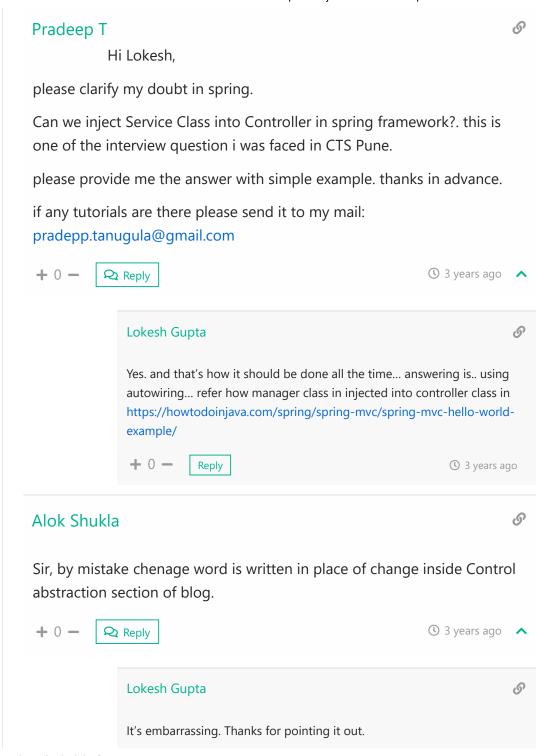
Leave a Reply

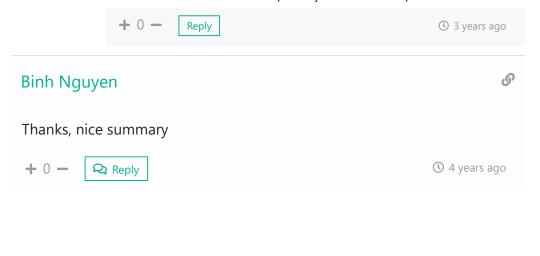
Join the discussion...

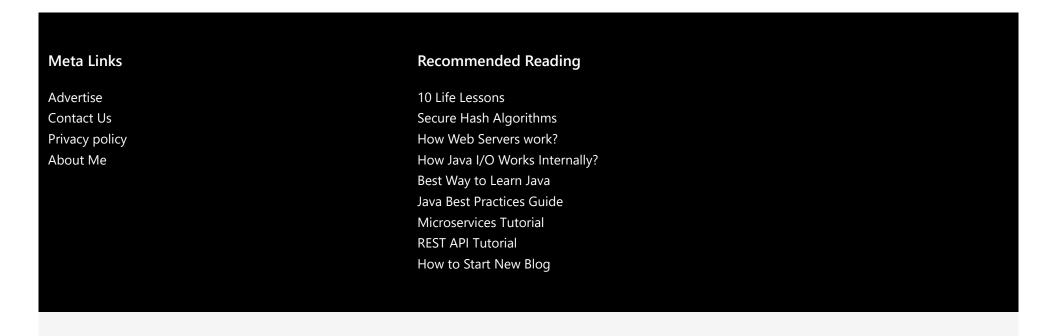












Copyright © 2016 · HowToDolnjava.com · All Rights Reserved. | Sitemap