Register | Login | Logout | Contact Us

Java-Success.com

Industrial strength Java/JEE Career Companion for those who want to go places

search here … Go

Home | Java FAQs | 600+ Java Q&As | Career | Tutorials | Member | Why?

Can u Debug? | Java 8 ready? | Top X | Productivity Tools | Judging Experience?

Home › Tech Key Areas › 13 Technical Key Areas Interview Q&A › Design Concepts › Understanding Open/Closed Principle (OCP) from the SOLID OO principles with a Java example

Understanding Open/Closed Principle (OCP) from the SOLID OO principles with a Java example

Posted on November 22, 2014 by Arulkumaran Kumaraswamipillai — No Comments ↓

0 Like Share 0 G+1 Share

Q. Is there anything wrong with the following class design? If yes, can the design be improved?

```
 package com.ocp;

import javax.management.RuntimeErrorException;

import org.apache.commons.lang.StringUtils;

public class MathOperation {

 public int operate(int input1, int input2, Stri

  if(StringUtils.isEmpty(operator)){
   throw new IllegalArgumentException("Invalid o
  }

  if(operator.equalsIgnoreCase("+")){
   return input1 + input2;
  }
  else if(operator.equalsIgnoreCase("*")){
   return input1 * input2;
  } else {
   throw new RuntimeException("unsupported opera
  }
 }

}

```

JUnit test class.

```
 package com.ocp;

import junit.framework.Assert;

import org.junit.Before;
import org.junit.Test;

public class MathOperationTest {

 MathOperation operation;

 @Before
 public void init(){
  operation = new MathOperation();
 }

 @Test
 public void testAddition() {
  Assert.assertEquals(8, operation.operate(5, 3,
 }

 @Test
 public void testMultiplication() {
  Assert.assertEquals(15, operation.operate(5, 3
 }

}

```

A. It's not a good idea to try to anticipate changes in requirements ahead of time, but you should focus on writing

As a Java Architect

Java architecture & design concepts

code that is well written enough so that it's easy to change. This means, you should strive to **write code that doesn't have to be changed every time the requirements change**. This is what the **Open/Closed principle** is. According to **GoF design pattern** authors "software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification". Spring framework promotes this principle.

In the above example, you can anticipate more operators like "-" (subtraction) and division (/) to be supported in the future and the class "***MathOperation***" is not closed for modification. When you need to support operators "-" and "%" you need to add 2 more "else if" statements. Whenever you see large if/else or switch statements, you need to think if "**Open/Closed**" design principle is more suited.

Let's open for extension and close for modifications

In the rewritten example below, the classes ***AddOperation*** and ***MultiplyOperation*** are closed for modification, but open for extension by allowing you to add new classes like ***SubtractOperation*** and ***DivisionOperation*** by implementing the ***Operation*** interface.

Define the **interface** Operation.

```
 package com.ocp;

public interface Operation {
       abstract int operate(int input1, int input
}

```

Define the **implementations**

```
 package com.ocp;


public class AddOperation implements Operation {

  @Override
  public int operate(int input1, int input2) {
   return input1 + input2;
  }

 }

```

```
  package com.ocp;

 public class MultiplyOperation implements Operat

  @Override
  public int operate(int input1, int input2) {
   return input1 * input2;
  }

 }

```

Finally, the **JUnit** test class

```
  package com.ocp;

 import junit.framework.Assert;

 import org.junit.Before;
 import org.junit.Test;

 public class MathOperation2Test {

  Operation operation;

  @Test
  public void testAddition() {
   operation = new AddOperation();
   Assert.assertEquals(8, operation.operate(5, 3)
  }

  @Test
  public void testMultiplication() {
   operation = new MultiplyOperation();
   Assert.assertEquals(15, operation.operate(5, 3
  }

 }

```

This is only a trivial example, but in real life applications, **wherever you have large if/else statements, you need to think if OCP can be applied**. Spring framework promotes this principle.

Popular Posts

- JEE Tutorials (19)
- Scala Tutorials (1)
- Spring & HIbernate Tu
- Tools Tutorials (19)
- Other Tutorials (45)

Preparing for Java written & coding tests

open all | close all

- ♦ Complete the given
- Can you write code? (
- Converting from A to I
- Designing your classe
- Java Data Structures
- Passing the unit tests
- What is wrong with th
- Writing Code Home A
- Written Test Core Jav
- Written Test JEE (1)

How good are your…to go places?

open all | close all

- Career Making Know-
- Job Hunting & Resum

♦ 11 Spring boot interview questions & answers

861 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

829 views

18 Java scenarios based interview Questions and Answers

448 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

407 views

♦ 7 Java debugging interview questions & answers

311 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

303 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

294 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

288 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

263 views

8 Git Source control system interview questions & answers

215 views

Bio | **Latest Posts**

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.

‹ 3 scenarios to get handle on Java generics

♦ 17 Java Coding Tips for job interviews and pre-interview coding tests ›

Posted in Design Concepts, member-paid, OOP

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.

© 2016 Java-Success.com ↑ Responsive Theme powered by WordPress