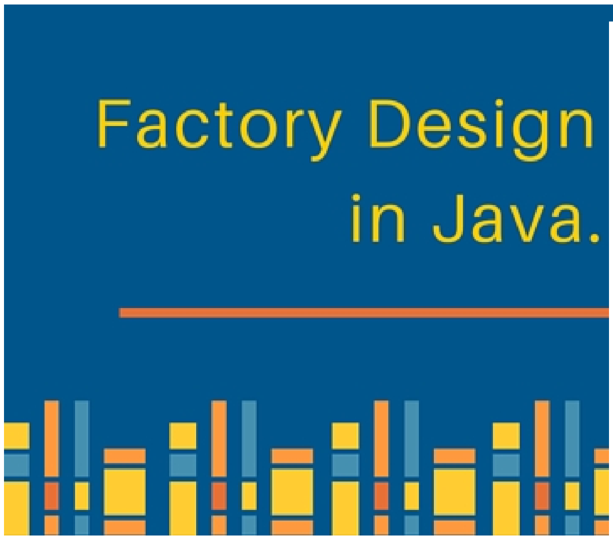
Factory Design Pattern in Ja

PANKAJ — 33 COMMENTS

Welcome to the Factory Design Pattern in Java tutorial. **Factory P**
pattern and it's widely used in JDK as well as frameworks like Sp

Table of Contents [hide]

Factory Design Pattern



Factory design pattern is used when we have a super class with r
we need to return one of the sub-class. This pattern take out the
from client program to the factory class.

Let's first learn how to implement factory design pattern in java ar
advantages. We will see some of factory design pattern usage in
Factory Method Design Pattern.

Factory Design Pattern Super Class

Super class in factory design pattern can be an interface, **abstrac**
factory design pattern example, we have abstract super class wit
testing purpose.

```
package com.journaldev.design.model;

public abstract class Computer {

        public abstract String getRAM();
        public abstract String getHDD();
        public abstract String getCPU();

        @Override
        public String toString(){
                return "RAM= "+this.getRAM()+", HDD="+
CPU="+this.getCPU();
        }
}
```

Factory Design Pattern Sub Classes

Let's say we have two sub-classes PC and Server with below imp

```
package com.journaldev.design.model;

public class PC extends Computer {

	private String ram;
	private String hdd;
	private String cpu;

	public PC(String ram, String hdd, String cpu){
		this.ram=ram;
		this.hdd=hdd;
		this.cpu=cpu;
	}
	@Override
	public String getRAM() {
		return this.ram;
	}

	@Override
	public String getHDD() {
		return this.hdd;
	}
```

Notice that both the classes are extending `Computer` super class

```
package com.journaldev.design.model;

public class Server extends Computer {

	private String ram;
```

```
        private String hdd;
        private String cpu;

        public Server(String ram, String hdd, String c
                this.ram=ram;
                this.hdd=hdd;
                this.cpu=cpu;
        }
        @Override
        public String getRAM() {
                return this.ram;
        }

        @Override
        public String getHDD() {
                return this.hdd;
```

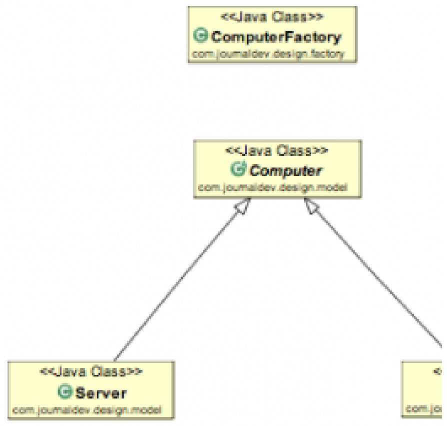
Factory Class

Now that we have super classes and sub-classes ready, we can v
implementation.

```
package com.journaldev.design.factory;

import com.journaldev.design.model.Computer;
import com.journaldev.design.model.PC;
import com.journaldev.design.model.Server;

public class ComputerFactory {

        public static Computer getComputer(String type
cpu){
                if("PC".equalsIgnoreCase(type)) return
                else if("Server".equalsIgnoreCase(type
cpu);

                return null;
        }
}
```

Some important points about Factory Design Pattern method are

1. We can keep Factory class Singleton or we can keep the me
2. Notice that based on the input parameter, different subclass
   the factory method.



<<Java Class>>
ComputerFactory
com.journaldev.design.factory
<<Java Class>>
Computer
com.journaldev.design.model
<<Java Class>>
Server
com.journaldev.design.model

Here is a simple test client program that uses above factory desig

```
package com.journaldev.design.test;

import com.journaldev.design.factory.ComputerFactory;
import com.journaldev.design.model.Computer;

public class TestFactory {

	public static void main(String[] args) {
		Computer pc = ComputerFactory.getComputer("pc","2 GB","500 GB","2.4 
GHz");
		Computer server = ComputerFactory.getComputer("server","16 GB","1 
TB","2.9 GHz");
		System.out.println("Factory PC Config:
		System.out.println("Factory Server Con
	}

}
```

Output of above program is:

```
Factory PC Config::RAM= 2 GB, HDD=500 GB, CPU=2.4 GHz
Factory Server Config::RAM= 16 GB, HDD=1 TB, CPU=2.9 G
```

Factory Design Pattern Advantages

1. Factory design pattern provides approach to code for interfa
2. Factory pattern removes the instantiation of actual impleme pattern makes our code more robust, less coupled and eas change PC class implementation because client program is
3. Factory pattern provides abstraction between implementati

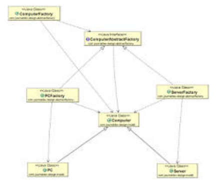
Factory Design Pattern Examples in JDK

1. java.util.Calendar, ResourceBundle and NumberFormat `get`
2. `valueOf()` method in wrapper classes like Boolean, Intege

Factory Design Pattern YouTube Video Tut

I recently uploaded a video on YouTube for Factory Design patter the video and subscribe to my YouTube channel.

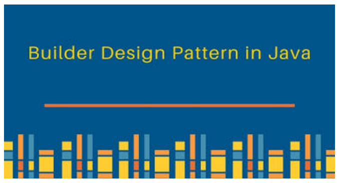
You can download the example code from my GitHub Project.



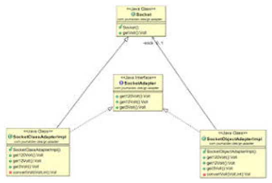
Abstract Factory Design Pattern in Java

Java Singleton Design Pattern Example Best Practices

Java Desig Example T



Builder Design Pattern in Java



Adapter Design Pattern in Java

Thread Sa Singleton Example C

« **PREVIOUS**

Java Singleton Design Pattern Best Practices with Examples

About Pankaj

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: DESIGN PATTERNS

X

Let's Stay Connected

5 Free Programming eBooks, Interview Tips, Exclusive Posts and much more...

Enter your name here...

Enter your email here...

SUBSCRIBE NOW

We promise not to spam you. You can unsubscribe at any time.

Comments

Justin says
FEBRUARY 27, 2019 AT 12:23 AM
how do i implement a factory to store my data and call it insi
Reply

7hills says
JANUARY 2, 2019 AT 5:52 AM
Thanks for sharing
Reply

supriya says
AUGUST 21, 2018 AT 3:01 AM
Very helpful artical. Thank you.
Reply

Jaleel says
JULY 24, 2018 AT 2:30 PM
Very weel explained! Thank you!
Reply

wade says
JUNE 3, 2018 AT 4:43 AM
very nice tutorial, easy to understand!
Reply

Nirav Khandhedia says
APRIL 4, 2018 AT 9:59 AM
I understand that there's always a benefit to get the instance
implementation.

How can I force users to compulsorily use the factory implementation to get the instance, instead of getting away creating an instance directly using new operato
i.e. How can I force people to use ServerFactory.getInstance(
Reply

Geeks says
APRIL 10, 2018 AT 2:31 AM
You can make Server Class as an abstract class.
Reply

abhi says
APRIL 19, 2018 AT 3:47 AM
But in that case Your Factory wont be able to create
Reply

Muthu Vignesh says
MAY 5, 2018 AT 11:56 PM
If you can make the constructor private as per singl
the instance using new Server(); and only enforces t
getInstance(). Also as an additional rule getInstance
implemented in the sub-classes extending comput
be present in Computer class which is already an abstract class. Hope m right, others correct me if m wrong
Reply

Srinivasa….. says
MARCH 30, 2018 AT 2:58 AM
It's Nice Explanation Learnt new things more it's best practic
Reply

Chris says
FEBRUARY 21, 2018 AT 3:13 AM
Quick and easy tutorial, thanks.

X
Let's Stay Connected
5 Free Programming eBooks, Interview Tips, Exclusive Posts and much more...
Enter your name here...
Enter your email here...
SUBSCRIBE NOW
We promise not to spam you. You can unsubscribe at any time.

Reply

Daniel says
FEBRUARY 16, 2018 AT 7:35 AM
Learned some new stuff with very detailed information.
Reply

BkOfc says
JANUARY 23, 2018 AT 5:39 PM
Where below are implemented
import com.journaldev.design.abstractfactory.PCFactory;
import com.journaldev.design.abstractfactory.ServerFactory;
Reply

Pankaj says
JANUARY 23, 2018 AT 8:33 PM
Sorry, that came out while copying the code from my Ec
useless, I have removed them from above code.
Reply

Prashanth says
NOVEMBER 15, 2017 AT 8:05 PM
It's a very good tutorial. But I have a doubt,
You mentioned Calendar#getInstance() as factory pattern implementation. But in this there is a small difference right?
There is no separate factory class. The super class Calendar itself is acting as the factory class.
Does an implementation like this have any advantage or disa
Reply

Luis Cunha says
SEPTEMBER 8, 2017 AT 8:02 AM
Hi, is there a place in which I can download all the source co

These are great examples, but I have to copy-paste each sin

cumbersome.

Thank you very much, and congratulations for such good ma

Reply

Vishal says

AUGUST 23, 2017 AT 1:47 AM

Yes. Its very nice article about simple factory covers basic co

Reply

Stephen Ubogu says

MAY 7, 2017 AT 4:09 AM

I am relatively new to design patterns but I need to ask this q subclass of computer say Laptop to the application.? Does th computer factory class? This looks like violating the OO princ modification but open to extension.

Reply

JocelynL says

OCTOBER 6, 2016 AT 3:28 AM

It seems to me that you're showing what is called a simple factory with ComputerFactory; It is not the Factory Method Pattern.

The client TestFactory delegates the creation to another class which it is composed with.

If you want to implement the Factory Method Pattern,:

1. ComputerFactory should define an asbtract method getComputer(String ram, String hdd, String cpu)
2. ComputerFactory should have two subclasses PCFactory and ServerFactory.which implements the superclass abstract method to return either a PC or a server
3. The client should be given one of the two concrete factories and call the factory method to get PC or servers, depending which one was instanciated

Reply

catherine says

FEBRUARY 25, 2017 AT 6:24 PM

yes , i agree. the article is about simple factory not facto

But still a nice article.

Reply

Vijay Kambala says
MAY 3, 2018 AT 5:18 AM
Could you please reply back with actual factory pat
in your own explanatroy words…
Reply

Vinod Kumar says
MARCH 23, 2017 AT 4:52 AM
yes absolutely you are right. It is not factory method patt
Reply

ravi says
JUNE 15, 2017 AT 1:18 AM
This is a factory (factory method) pattern, if you make fa
factory pattern
Reply

Gani Victory says
AUGUST 11, 2016 AT 3:37 AM
Nice article !!!!!!!!
Reply

panky031 says
JUNE 3, 2016 AT 7:27 AM
Now i found the perfect article for Design pattern.
Thanks Pankaj
Reply

vamshi says
FEBRUARY 23, 2015 AT 6:20 PM

Thanks for the clear explanation.
I have one doubt here in Factory pattern. We have two concr
interface/Abstract class whose instances are created inside
Computer pc = ComputerFactory.getComputer("pc","2 GB","50
we can also use
Computer pc=new PC("pc","2 GB","500 GB","2.4 GHz"); to get ne
advantage of using ComputerFactory.getComputer() method
Reply

Ofer Yuval says
APRIL 22, 2015 AT 12:03 AM
See here
http://stackoverflow.com/questions/14575457/factory-
Reply

Ajay says
APRIL 5, 2016 AT 11:29 AM
That's why it's called an creation all design pattern caus
keeping the instance creation all logic here and there ca
which is just doing it for us you just name it..name the ob
Reply

Avinash Nayak says
SEPTEMBER 14, 2017 AT 10:31 AM
Thats because u will be bound to the object, ie if you create Computer pc=new PC("pc","2 GB","500 GB","2.4 GHz") u will always get the instance of PC and it would be hardcoding, So if you use factory you wil not worry of the implementation u will always get the object of reference Computer.
Reply

Siva says
OCTOBER 10, 2014 AT 2:26 PM
Some body asked me that Why do we have to implement si
And here in your post you say that either we can use static m
please detail on this?
Reply

JavaBee says
MAY 16, 2016 AT 7:00 AM
May be its a late reply, but worth share thought here.
Factory classes(In general design patterns) are meant fo
pattern states that, the objective of this pattern is to dec
program. And this can be achieved either by static meth
Why we use singleton pattern when we have static?
we use "static" when a piece of code/data same across
of code that needs to be executed even when class is n
required.
Question here is, how to make outer class itself static? tl
The "singleton pattern" is a mechanism, which gives the
class(avoid multiple instantiation of a class when it is not
functionality only once) OR have the single instance to a
This can be achieved by using static + additional checks
Ex: Thread pool.
Reply

robothy says
OCTOBER 11, 2016 AT 2:26 AM
Good answer!!!
Reply

RazorEdge says
FEBRUARY 26, 2017 AT 8:14 AM
Very good answer… Thank You
Reply

Comment Policy: Please submit comments to add value to the post. Comments like "Tha
want to post code then wrap them inside <pre> tags. For example **<pre>class Foo { }</p**
If you want to post XML content, then please escape < with &lt; and > with &gt; otherwise

Let's Stay Connected

5 Free Programming eBooks, Interview Tips, Exclusive Posts and much more...

Enter your name here...

Enter your email here...

SUBSCRIBE NOW

We promise not to spam you. You can unsubscribe at any time.

X

Leave a Reply

Your email address will not be published. Required fields are mar
Comment

Name *

Email *

POST COMMENT

Instantly Search Tutorials...

DESIGN PATTERNS TUTORIAL

Java Design Patterns

Let's Stay Connected

5 Free Programming eBooks, Interview Tips, Exclusive Posts and much more...

Enter your name here...

Enter your email here...

SUBSCRIBE NOW

We promise not to spam you. You can unsubscribe at any time.