instance factory methods Vs Static factory methods

Ask Question

Can't all factory methods be static ? Does something that produces a product need state ? When is it appropriate to go for a instance factory or static factory method ? Can you provide me examples differentiating the two ?

java factory-pattern

edited Nov 16 '11 at 5:39

asked Nov 16 '11 at 5:32
user2434
2,835 11 47 73

3 "a instance static method" is self-contradictory. Please revise your question. Perhaps you meant "a instance factory method". As to your concrete question, you're basically asking about "Abstract Factory" versus "Factory Method". That must give you new keywords. Wikipedia for example has 2 separate articles on those design patterns. – BalusC Nov 16 '11 at 5:36

edited the question – user2434 Nov 16 '11 at 5:39

2 Static factory methods are essentially named constructors, so yes, they must be static. I'm not familiar with instance factory methods, unless you are referring to the Abstract Factory pattern which has instance methods that are factories, but that is a different concept. IMHO Static factory method is a bad name and causes confusion, it should have been named something like named constructor method, clearly expressing that it's a more powerful alternative to using the default Java constructor syntax. – Daniel Canas Nov 16 '11 at 6:33

4 Answers

Assuming that by "instance factory method" you're actually saying about the GoF "factory method", the term "static factory method" is described by Joshua Bloch in his book "Effective Java". Googling around I reached at these sites:

- Factory Method: http://sourcemaking.com/design_patterns/factory_method

- Static Factory Method: http://www.informit.com/articles/article.aspx?p=1216151

Hope that it helped to make the difference a little bit clearer.

Following Marvo's advice:

- Factory Method as stated in GoF:

  > define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.

Example (Java code):

```
public abstract class Product { ... }

public class ConcreteProduct extends Product { ... }

public abstract class Creator {
  public void anOperation() { ... product = factoryMethod(); ... }
  public abstract Product factoryMethod();
}

public class ConcreteCreator extends Creator {
  public Product factoryMethod() { return new ConcreteProduct(); }
}
```

- Static Factory Method as stated in Effective Java:

  > A class can provide a public static factory method, which is simply a static method that returns an instance of the class. (...) One advantage of static factory methods is that, unlike constructors, they have names. (...) A second advantage of static factory methods is that, unlike constructors, they are not required to create a new object each time they're invoked. (...) A third advantage of static factory methods is that, unlike constructors, they can return an object of any subtype of their return type. (...) The main disadvantage of providing only static factory methods is that classes without public or protected constructors cannot be subclassed.

Example (Java code):

```
public class Boolean {
  ...
  public static Boolean valueOf(boolean b) {
    return b ? Boolean.TRUE : Boolean.FALSE;
```

```
        }
        ...
    }
```

edited Mar 19 at 12:10
Bless
1,548 2 14 23

answered Dec 8 '12 at 23:20
Ramon Chiara
376 2 6

My current preference is to make factory methods not static for the sake of easier testing. You can't change a static factory method call at runtime, whereas if I could provide a factory implementation to the object then I could test it more thoroughly as I'm in more control of the context and object graph.

answered Nov 16 '11 at 5:58
Nate W.
7,334 6 30 59

---

4 I did not understand what you are saying. Can you please give an example to make your point clear ? – user2434 Nov 16 '11 at 6:03

@user2434: This now depends on what exactly we mean by static factory vs instance factory. Could you provide an example of what you're thinking a static factory and instance factory would look like? – Nate W. Nov 16 '11 at 6:54

@NateW. I did not quite what you mean by saying "You can't change a static factory method call at runtime" can explaing it in more details pls – M.T Jun 25 '17 at 7:15

If you are returning a Singleton from your factory then you are going to need make sure you only have one instance, if you are going to create a new instance every time you call the factory then make it static.

answered Nov 16 '11 at 5:42

Bob The Janitor
9,653 7 37 69

why not use just a static factory to return new instances all the time ? is it not possible ? – user2434 Nov 16 '11 at 6:02

2 there is nothing wrong with that at all, that is how I generally do it. – Bob The Janitor Nov 16 '11 at 14:49

It depends, for instance you can have a factory that will only produce a certain amount of objects in which case there will be an associated state. For the most part factory methods can be static as long as they don't rely on any non-static variables (such as non-static globals and such) for their creation. It also tends to differentiate different factories for different parts of an application so depending on whether there is a state or not in your factory is what you would go with it's not set in stone that all factory methods should be static, check what situation applies to you and write it appropriately.

Another consideration is the static keyword, this causes whatever is static to be instantiated only once in memory but a drawback is that it resides in the process memory always (which increases working set size). This may be something that you don't want as your factory might have very high locality in certain areas and it would otherwise just be using up memory somewhere else but this is usually an optimization issue that should be looked at only if the issue of memory pressure in your application arises.

answered Nov 16 '11 at 5:35
Jesus Ramos
19.4k 8 45 76