Top 18 Java Design Pattern Interview Questions Answers for Experienced

Difference between prototype and factory pattern

1. Factory method is used to delegate the responsibility of choosing which implementation or subclass you want to use like Car interface can be implemented by SportsCar and EconomicalCar and based upon budget factory will return appropriate object.  
  
2. Prototype is used in scenarios where construction of the object is costly affair. It could be in terms of memory or computation for example, you have complex objects like Trade so, you can create a basic default object and on runtime can just clone it and do some changes as per requirement.

**Design pattern interview question in Java**  
Both OOP and GOF design pattern interview questions are an integral part of any good list of core Java interview questions. Java is a popular Object oriented programming language and has lots of design pattern and design principles, contributed by many developers and open source framework. As a Java programmer its expected from you to know OOPS concepts like [Abstraction](http://javarevisited.blogspot.sg/2010/10/abstraction-in-java.html), [Encapsulation,](http://javarevisited.blogspot.sg/2012/03/what-is-encapsulation-in-java-and-oops.html)and [Polymorphism](http://javarevisited.blogspot.sg/2011/08/what-is-polymorphism-in-java-example.html), What is design pattern in Java, Some popular Java design pattern and most importantly when to use those design pattern in Java application. The purpose of asking design pattern interview question in Java is to check whether Java programmer is familiar with those essential design patterns or not.  
  
Design patterns in Java interviews are as important as[multi-threading](http://javarevisited.blogspot.sg/2011/07/java-multi-threading-interview.html), [collection,](http://javarevisited.blogspot.sg/2011/11/collection-interview-questions-answers.html)and [programming questions](http://javarevisited.blogspot.sg/2011/06/top-programming-interview-questions.html). If you are senior or experienced Java programmer than expecting more complex and tough design pattern in Java interview e.g. Chain of Responsibility design pattern and solving real-time [software design questions](http://javarevisited.blogspot.sg/2012/06/20-design-pattern-and-software-design.html).

**Top Java design pattern questions and answers**

Here is my list of t*op 10 design pattern interview question in Java*. I have also provided an answer to those Java design pattern question as a link. no matter which level of Java interview is you going e.g. programmer, software engineer, senior software engineer in Java, you can expect few question from Java design pattern.  
  
**1. When to use Strategy Design Pattern in Java?**  
Strategy pattern in quite useful for implementing set of related algorithms e.g. compression algorithms, filtering strategies etc. Strategy design pattern allows you to create Context classes, which uses Strategy implementation classes for applying business rules. This pattern follows open closed design principle and quite useful in Java.  
  
One of a good example of Strategy pattern from JDK itself is a Collections.sort() method and [Comparator interface](http://java67.blogspot.sg/2012/10/how-to-sort-object-in-java-comparator-comparable-example.html), which is a strategy interface and defines a strategy for comparing objects. Because of this pattern, we don't need to modify sort() method (closed for modification) to compare any object, at the same time we can implement Comparator interface to define new comparing strategy (open for extension).  
  
  
**2. What is Observer design pattern in Java? When do you use Observer pattern in Java?**  
This is one of the most common Java design pattern interview questions. Observer pattern is based upon notification, there are two kinds of object Subject and Observer. Whenever there is change on subject's state observer will receive notification. See [What is Observer design pattern in Java with real life example](http://javarevisited.blogspot.sg/2011/12/observer-design-pattern-java-example.html) for more details.

# [A simple Swing GUI example for Observer Design Pattern](https://www.programcreek.com/2009/01/the-steps-involved-in-building-a-swing-gui-application/)

This example shows how to create a Swing GUI example, and explain why it is an example usage of Observer Design Pattern.

|  |
| --- |
| JFrame frame = **new** JFrame("Frame Title"); |

Create some components such as panels, buttons, text areas etc.

|  |
| --- |
| **final** JTextArea comp = **new** JTextArea();  JButton btn = **new** JButton("click"); |

Add components to display area and arrange its layout using the LayoutManagers.

|  |
| --- |
| frame.getContentPane().add(comp,BorderLayout.CENTER);  frame.getContentPane().add(btn, BorderLayout.SOUTH); |

Attach a listener to the button component. Interacting with a Component causes an Event to occur. To associate a user action with a component, attach a listener to it.

Here addActionListener method is the subject's register observer method. For a complete example of Observer design pattern, go to [Observer](https://www.programcreek.com/2011/01/an-java-example-of-observer-pattern/)example.

|  |
| --- |
| btn.addActionListener(**new** ActionListener(){  **public** **void** actionPerformed(ActionEvent ae){  comp.setText("Button has been clicked");  }  }); |

*public interface ActionListener extends EventListener*

The listener interface is for receiving action events. The class (Main, in this case) that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked.

**3. Difference between Strategy and State design Pattern in Java?**  
This is an interesting Java design pattern interview questions as both Strategy and State pattern has the same structure. If you look at UML class diagram for both patterns they look exactly same, but their intent is totally different.  
  
State design pattern is used to define and manage the state of an object, while Strategy pattern is used to define a set of an interchangeable algorithm and let's client choose one of them. So [Strategy pattern](http://www.shareasale.com/m-pr.cfm?merchantID=53701&userID=880419&productID=546412275" \t "_blank) is a client driven pattern while Object can manage their state itself.  
  
  
  
**4. What is decorator pattern in Java? Can you give an example of Decorator pattern?**  
Decorator pattern is another popular Java design pattern question which is common because of its heavy usage in java.io package. BufferedReader and BufferedWriter are a good example of decorator pattern in Java. See [How to use Decorator pattern in Java](http://javarevisited.blogspot.com/2011/11/decorator-design-pattern-java-example.html) for more details.  
  
  
**5. When to use Composite design Pattern in Java? Have you used previously in your project?**  
This design pattern question is asked on Java interview not just to check familiarity with the Composite pattern but also, whether a candidate has the real life experience or not.  
The*Composite pattern* is also a core Java design pattern, which allows you to treat both whole and part object to treat in a similar way. Client code, which deals with a Composite or individual object doesn't differentiate between them, it is possible because Composite class also implement the same interface as their individual part.  
  
One of the good examples of the Composite pattern from JDK is JPanel class, which is both Component and Container.  When the paint() method is called on JPanel, it internally called paint() method of individual components and let them draw themselves.  
  
On the second part of this design pattern interview question, be truthful, if you have used then say yes, otherwise say that you are familiar with the concept and used it by your own. By the way, always remember, giving an example from your project creates a better impression.  
  
  
  
**6. What is Singleton pattern in Java?**  
Singleton pattern in Java is a pattern which allows only one instance of Singleton class available in the whole application. java.lang.Runtime is a good example of Singleton pattern in Java. There are lot's of follow up questions on Singleton pattern see [10 Java singleton interview question answers](http://javarevisited.blogspot.com/2011/03/10-interview-questions-on-singleton.html) for those followups  
  
  
**7. Can you write thread-safe Singleton in Java?**  
There are multiple ways to write thread-safe singleton in Java e.g by writing singleton using double checked locking, by using static Singleton instance initialized during [class loading.](http://javarevisited.blogspot.sg/2012/07/when-class-loading-initialization-java-example.html) By the way using Java enum to create thread-safe singleton is the most simple way. See [Why Enum singleton is better in Java](http://javarevisited.blogspot.gr/2012/07/why-enum-singleton-are-better-in-java.html) for more details.  
  
  
**8. When to use Template method design Pattern in Java?**  
The Template pattern is another popular core Java design pattern interview question. I have seen it appear many times in real life project itself. Template pattern outlines an algorithm in form of template method and lets subclass implement individual steps.  
  
The key point to mention, while answering this question is that template method should be final, so that subclass can not override and change steps of the algorithm, but same time individual step should be abstract, so that child classes can implement them.  
  
  
**9. What is Factory pattern in Java? What is the advantage of using a static factory method to create an object?**  
Factory pattern in Java is a creation Java design pattern and favorite on many Java interviews.Factory pattern used to create an object by providing static factory methods. There are many advantages of providing factory methods e.g. caching immutable objects, easy to introduce new objects etc. See [What is Factory pattern in Java and benefits](http://javarevisited.blogspot.sg/2011/12/factory-design-pattern-java-example.html) for more details.

1. Factory design pattern provides approach to code for interface rather than implementation.
2. Factory pattern removes the instantiation of actual implementation classes from client code. Factory pattern makes our code more robust, less coupled and easy to extend. For example, we can easily change PC class implementation because client program is unaware of this.
3. Factory pattern provides abstraction between implementation and client classes through inheritance.

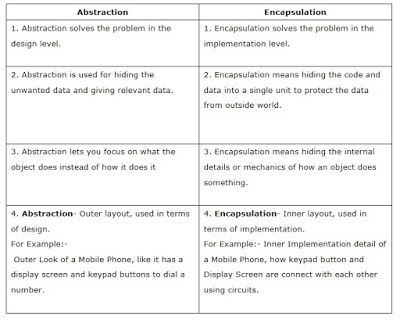
### Factory Design Pattern Examples in JDK

1. java.util.Calendar, ResourceBundle and NumberFormat getInstance() methods uses Factory pattern.
2. valueOf() method in wrapper classes like Boolean, Integer etc.

**10. What is the difference between Decorator and Proxy pattern in Java?**  
Another tricky Java design pattern question and trick here is that both Decorator and Proxy implements the interface of the object they decorate or encapsulate. As I said, many Java design pattern can have similar or exactly same structure but they differ in their intent.  
  
Decorator pattern is used to implement functionality on an already created object, while a Proxy pattern is used for controlling access to an object.  
  
One more difference between Decorator and the Proxy design pattern is that Decorator doesn't create an object, instead, it get the object in its constructor, while Proxy actually creates objects. You can also read [Head First Analysis and Design](http://aax-us-east.amazon-adsystem.com/x/c/QiDtfUpIWr7_ZfWCT8bbpPEAAAFiBnrmgAEAAAFKAfAZsrI/https:/assoc-redirect.amazon.com/g/r/http:/www.amazon.com/dp/0596008678/ref=as_at?creativeASIN=0596008678&linkCode=w61&imprToken=UvLq3Hf4Mi18x8nWlR.Czw&slotNum=0&tag=javamysqlanta-20" \t "_blank) to understand the difference between them.  
  
  
**11. When to use Setter and Constructor Injection in Dependency Injection pattern?**  
Use Setter injection to provide optional dependencies of an object, while use Constructor iInjection to provide a mandatory dependency of an object, without which it can not work. This question is related to [Dependency Injection design pattern](http://javarevisited.blogspot.com/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html) and mostly asked in the context of Spring framework, which is now become a standard for developing Java application.  
  
Since Spring provides IOC container, it also gives you a way to specify dependencies either by using setter methods or constructors. You can also take a look my [previous post](http://javarevisited.blogspot.com/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html) on the same topic.  
  
  
**12. What is difference between Factory and Abstract Factory in Java**  
I have already answered this question in detail with my article with the same title. The main difference is that Abstract Factory creates factory while Factory pattern creates objects. So both abstract the creation logic but one abstract is for factory and other for items. You can see [here](http://javarevisited.blogspot.sg/2013/01/difference-between-factory-and-abstract-factory-design-pattern-java.html" \t "_blank) to answer this Java design pattern interview question.  
  
  
**13. When to use Adapter pattern in Java? Have you used it before in your project?**  
Use Adapter pattern when you need to make two class work with incompatible interfaces. Adapter pattern can also be used to encapsulate third party code so that your application only depends upon Adapter, which can adapt itself when third party code changes or you moved to a different third party library.  
  
By the way, this Java design pattern question can also be asked by providing an actual scenario. You can further read [Head First Design Pattern](http://aax-us-east.amazon-adsystem.com/x/c/QiDtfUpIWr7_ZfWCT8bbpPEAAAFiBnrmgAEAAAFKAfAZsrI/https:/assoc-redirect.amazon.com/g/r/http:/www.amazon.com/dp/0596007124/ref=as_at?creativeASIN=0596007124&linkCode=w61&imprToken=UvLq3Hf4Mi18x8nWlR.Czw&slotNum=1&tag=javamysqlanta-20" \t "_blank) to learn more about Adapter pattern and its real world usage. The book is updated for Java 8 as well so you will learn new, Java 8 way to implement these old design patterns.

**14. Can you write code to implement producer consumer design pattern in Java?**  
The Producer-consumer design pattern is a concurrency design pattern in Java which can be implemented using multiple ways. If you are working in Java 5 then its better to use Concurrency util to implement producer-consumer pattern instead of plain old [wait and notify in Java](http://javarevisited.blogspot.sg/2011/05/wait-notify-and-notifyall-in-java.html).  Here is a good example of implementing [producer consumer problem using BlockingQueue in Java](http://javarevisited.blogspot.sg/2012/02/producer-consumer-design-pattern-with.html).  
  
  
**15. What is Open closed design principle in Java?**  
The Open closed design principle is one of the SOLID principle defined by Robert C. Martin, popularly known as Uncle Bob in his most popular book, [Clean Code](http://aax-us-east.amazon-adsystem.com/x/c/QiDtfUpIWr7_ZfWCT8bbpPEAAAFiBnrmgAEAAAFKAfAZsrI/https:/assoc-redirect.amazon.com/g/r/http:/www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882/ref=as_at?creativeASIN=0132350882&linkCode=w61&imprToken=UvLq3Hf4Mi18x8nWlR.Czw&slotNum=3&tag=javamysqlanta-20" \t "_blank). This principle advises that a code should be open for extension but closed for modification.

At first, this may look conflicting but once you explore the power of polymorphism, you will start finding patterns which can provide stability and flexibility of this principle.  
  
One of the key examples of this is State and Strategy design pattern, where Context class is closed for modification and new functionality is provided by writing new code by implementing a new state of strategy. See [this](http://javarevisited.blogspot.com/2011/11/great-example-of-open-closed-design.html)article to know more about Open closed principle.  
  
  
**16. What is Builder design pattern in Java? When do you use Builder pattern?**  
Builder pattern in Java is another creational design pattern in Java and often asked in Java interviews because of its specific use when you need to build an object which requires multiple properties some optional and some mandatory. See [When to use Builder pattern in Java](http://javarevisited.blogspot.sg/2012/06/builder-design-pattern-in-java-example.html) for more details  
  
  
**17. Can you give an example of  SOLID design principles in Java?**  
There are lots of SOLID design pattern which forms acronym SOLID e.g.  
1. Single Responsibility Principle or SRP  
3. Open Closed Design Principle or OCD  
3. Liskov Substitution Principle  
4. Interface Segregation Principle  
5. Dependency Inversion Principle.  
  
You can further read this [list of SOLID design principles](http://www.shareasale.com/m-pr.cfm?merchantID=53701&userID=880419&productID=546412145" \t "_blank) for Java programmer to answer this Java interview question.  
 **18. What is the difference between Abstraction and Encapsulation in Java?**  
Even though both Abstraction and Encapsulation looks similar because both hide complexity and make the external interface simpler there is a subtle difference between them. Abstraction hides logical complexity while Encapsulation hides Physical Complexity.  
  
Btw, I have already covered answer of this Java interview question in my previous post as [Difference between encapsulation and abstraction in Java](http://java67.blogspot.sg/2012/08/difference-between-abstraction-and-encapsulation-java-oops.html), here are some more difference from that post.

[](http://www.shareasale.com/m-pr.cfm?merchantID=53701&userID=880419&productID=546412145)

This was my list of **10 popular design pattern interview question in Jav**a. I have not included MVC (Model View Controller) design pattern because that is more specific to J2EE and [Servlet JSP interview](http://javarevisited.blogspot.sg/2011/09/servlet-interview-questions-answers.html), but if you are going for any Java interview which demands experience in J2EE then you must prepare MVC design pattern. That's all on Java design pattern interview question and answers. Please let us know if you have any other interesting question on Java design pattern.