Table of Contents

* [Object Oriented Programming (OOP)](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#1)
* [General Questions about Java](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#2)
* [Java Threads](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#3)
* [Java Collections](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#4)
* [Garbage Collectors](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#5)
* [Exception Handling](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#6)
* [Java Applets](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#7)
* [Swing](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#8)
* [JDBC](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#9)
* [Remote Method Invocation (RMI)](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#10)
* [Servlets](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#11)
* [JSP](https://www.javacodegeeks.com/2014/04/java-interview-questions-and-answers.html#12)

Object Oriented Programming (OOP)

Java is a computer programming language that is concurrent, class-based and object-oriented. The advantages of object oriented software development are shown below:

* Modular development of code, which leads to easy maintenance and modification.
* Reusability of code.
* Improved reliability and flexibility of code.
* Increased understanding of code.

Object-oriented programming contains many significant features, such as **encapsulation**, **inheritance**, **polymorphism**and [**abstraction**](http://www.javacodegeeks.com/2014/07/abstraction-in-java.html). We analyze each feature separately in the following sections.

Encapsulation

Encapsulation provides objects with the ability to hide their internal characteristics and behavior. Each object provides a number of methods, which can be accessed by other objects and change its internal data. In Java, there are three access modifiers: public, private and protected. Each modifier imposes different access rights to other classes, either in the same or in external packages. Some of the advantages of using encapsulation are listed below:

* The internal state of every objected is protected by hiding its attributes.
* It increases usability and maintenance of code, because the behavior of an object can be independently changed or extended.
* It improves modularity by preventing objects to interact with each other, in an undesired way.

You can refer to our tutorial [here](http://examples.javacodegeeks.com/java-basics/encapsulation-in-java/) for more details and examples on encapsulation.

Polymorphism

Polymorphism is the ability of programming languages to present the same interface for differing underlying data types. A polymorphic type is a type whose operations can also be applied to values of some other type.

Inheritance

Inheritance provides an object with the ability to acquire the fields and methods of another class, called base class. Inheritance provides re-usability of code and can be used to add additional features to an existing class, without modifying it.

Abstraction

[Abstraction](http://www.javacodegeeks.com/2014/04/why-abstraction-is-really-important.html) is the process of separating ideas from specific instances and thus, develop classes in terms of their own functionality, instead of their implementation details. Java supports the creation and existence of abstract classes that expose interfaces, without including the actual implementation of all methods. The abstraction technique aims to separate the implementation details of a class from its behavior.

Differences between Abstraction and Encapsulation

Abstraction and encapsulation are complementary concepts. On the one hand, abstraction focuses on the behavior of an object. On the other hand, encapsulation focuses on the implementation of an object’s behavior. Encapsulation is usually achieved by hiding information about the internal state of an object and thus, can be seen as a strategy used in order to provide abstraction.

General Questions about Java

**1. What is JVM ? Why is Java called the “Platform Independent Programming Language” ?** A Java virtual machine (JVM) is a process [virtual machine](http://www.javacodegeeks.com/2013/12/part-1-of-3-synopsis-of-articles-videos-on-performance-tuning-jvm-gc-in-java-mechanical-sympathy-et-al.html) that can execute Java [bytecode](http://www.javacodegeeks.com/2013/12/mastering-java-bytecode.html). Each Java source file is compiled into a bytecode file, which is executed by the JVM. Java was designed to allow application programs to be built that could be run on any platform, without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible, because it is aware of the specific instruction lengths and other particularities of the underlying hardware platform.

**2. What is the Difference between JDK and JRE ?** The Java Runtime Environment (JRE) is basically the Java Virtual Machine (JVM) where your Java programs are being executed. It also includes browser plugins for applet execution. The Java Development Kit (JDK) is the full featured Software Development Kit for Java, including the JRE, the compilers and tools (like [JavaDoc](http://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html), and [Java Debugger](http://docs.oracle.com/javase/7/docs/technotes/tools/windows/jdb.html)), in order for a user to develop, compile and execute Java applications.

**3. What does the “static” keyword mean ? Can you override private or static method in Java ?** The static keyword denotes that a member variable or method can be accessed, without requiring an instantiation of the class to which it belongs. A user cannot override [static methods in Java](http://www.javacodegeeks.com/2012/05/java-static-methods-can-be-code-smell.html), because method overriding is based upon dynamic binding at runtime and static methods are statically binded at compile time. A static method is not associated with any instance of a class so the concept is not applicable.

**4. Can you access non static variable in static context ?** A static variable in Java belongs to its class and its value remains the same for all its instances. A static variable is initialized when the class is loaded by the JVM. If your code tries to access a non-static variable, without any instance, the compiler will complain, because those variables are not created yet and they are not associated with any instance.

**5. What are the Data Types supported by Java ? What is Autoboxing and Unboxing ?** The eight primitive data types supported by the Java programming language are:

* byte
* short
* int
* long
* float
* double
* boolean
* char

Autoboxing is the [automatic conversion made by the Java compiler](http://www.javacodegeeks.com/2013/07/java-generics-tutorial-example-class-interface-methods-wildcards-and-much-more.html) between the primitive types and their corresponding object wrapper classes. For example, the compiler converts an int to an [Integer](http://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html?is-external=true), a double to a [Double](http://docs.oracle.com/javase/7/docs/api/java/lang/Double.html), and so on. If the conversion goes the other way, this operation is called unboxing.

**6. What is Function Overriding and Overloading in Java ?** Method overloading in Java occurs when two or more methods in the same class have the exact same name, but different parameters. On the other hand, method overriding is defined as the case when a child class redefines the same method as a parent class. Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides.

**7. What is a Constructor, Constructor Overloading in Java and Copy-Constructor ?** A constructor gets invoked when a new object is created. Every class [has a constructor](http://www.javacodegeeks.com/2014/01/which-is-better-option-cloning-or-copy-constructors.html). In case the programmer does not provide a constructor for a class, the Java compiler (Javac) creates a default constructor for that class. The constructor overloading is similar to method overloading in Java. Different constructors can be created for a single class. Each constructor must have its own unique parameter list. Finally, Java does support copy constructors like C++, but the difference lies in the fact that Java doesn’t create a default copy constructor if you don’t write your own.

**8. Does Java support multiple inheritance ?** No, Java does not support multiple inheritance. Each class is able to extend only on one class, but is able to implement more than one interfaces.

**9. What is the difference between an Interface and an Abstract class ?** Java provides and supports the creation both of [abstract classes](http://examples.javacodegeeks.com/java-basics/java-abstract-class-example/) and interfaces. Both implementations share some common characteristics, but they differ in the following features:

* All methods in an interface are implicitly abstract. On the other hand, an abstract class may contain both abstract and non-abstract methods.
* A class may implement a number of Interfaces, but can extend only one abstract class.
* In order for a class to implement an interface, it must implement all its declared methods. However, a class may not implement all declared methods of an abstract class. Though, in this case, the sub-class must also be declared as abstract.
* Abstract classes can implement interfaces without even providing the implementation of interface methods.
* Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.
* Members of a Java interface are public by default. A member of an abstract class can either be private, protected or public.
* An interface is absolutely abstract and cannot be instantiated. An abstract class also cannot be instantiated, but can be invoked if it contains a main method.

Also check out the [Abstract class and Interface differences for JDK 8](http://www.javacodegeeks.com/2014/04/abstract-class-versus-interface-in-the-jdk-8-era.html).

**10. What are pass by reference and pass by value ?** When an object is passed by value, this means that a copy of the object is passed. Thus, even if changes are made to that object, it doesn’t affect the original value. When an object is passed by reference, this means that the actual object is not passed, rather a reference of the object is passed. Thus, any changes made by the external method, are also reflected in all places.

Java Threads

**11. What is the difference between processes and threads ?** A process is an execution of a program, while a [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html)is a single execution sequence within a process. A process can contain multiple threads. A [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) is sometimes called a lightweight process.

**12. Explain different ways of creating a thread. Which one would you prefer and why ?** There are three ways that can be used in order for a [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) to be created:

* A class may extend the [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) class.
* A class may implement the [Runnable](http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html) interface.
* An application can use the [Executor](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Executor.html) framework, in order to create a thread pool.

The [Runnable](http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html) interface is preferred, as it does not require an object to inherit the [Thread](http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) class. In case your application design requires multiple inheritance, only interfaces can help you. Also, the thread pool is very efficient and can be implemented and used very easily.

**13. Explain the available thread states in a high-level.** During its execution, a thread can reside in one of the following [states](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html):

1. [**NEW**](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#NEW): The thread becomes ready to run, but does not necessarily start running immediately.
2. [**RUNNABLE**](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#RUNNABLE): The Java Virtual Machine (JVM) is actively executing the thread’s code.
3. [**BLOCKED**](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#BLOCKED): The thread is in a blocked state while waiting for a monitor lock.

[**WAITING**](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#WAITING): The thread waits for another thread to perform a particular **action**.

**AqqqTIMED\_WAITING**: The thread waits for another thread to perform a particular qsa. AT6xaaction up to a specified waiting time.

1. [**TERMINATED**](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html#TERMINATED): The thread has finished its execution.

**14. What is the difference between a synchronized method and a synchronized block ?** In Java programming, each object has a lock. A thread can acquire the lock for an object by using the synchronized keyword. The synchronized keyword can be applied in a method level (coarse grained lock) or block level of code (fine grained lock).

**15. How does thread synchronization occurs inside a monitor ? What levels of synchronization can you apply ?**The JVM uses locks in conjunction with monitors. A monitor is basically a guardian that watches over a sequence of synchronized code and ensuring that only one thread at a time executes a synchronized piece of code. Each monitor is associated with an object reference. The thread is not allowed to execute the code until it obtains the lock.

**16. What’s a deadlock ?** A condition that occurs when [two processes are waiting for each other to complete](http://www.javacodegeeks.com/2013/01/java-deadlock-example-how-to-analyze-deadlock-situation.html), before proceeding. The result is that both processes wait endlessly.

**17. How do you ensure that N threads can access N resources without deadlock ?** A very simple way to avoid deadlock while using N threads is to impose an ordering on the locks and force each thread to follow that ordering. Thus, if all threads lock and unlock the mutexes in the same order, no deadlocks can arise.

Java Collections

**18. What are the basic interfaces of Java Collections Framework ?** [Java Collections Framework](http://docs.oracle.com/javase/7/docs/technotes/guides/collections/overview.html) provides a well designed set of interfaces and classes that support operations on a collections of objects. The most basic interfaces that reside in the Java Collections Framework are:

* [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html), which represents a group of objects known as its elements.
* [Set](http://docs.oracle.com/javase/7/docs/api/java/util/Set.html), which is a collection that cannot contain duplicate elements.
* [List](http://docs.oracle.com/javase/7/docs/api/java/util/List.html), which is an ordered collection and can contain duplicate elements.
* [Map](http://docs.oracle.com/javase/7/docs/api/java/util/Map.html), which is an object that maps keys to values and cannot contain duplicate keys.

**19. Why Collection doesn’t extend Cloneable and Serializable interfaces ?** The [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) interface specifies groups of objects known as elements. Each concrete implementation of a [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) can choose its own way of how to maintain and order its elements. Some collections allow duplicate keys, while some other collections don’t. The semantics and the implications of either cloning or serialization come into play when dealing with actual implementations. Thus, the concrete implementations of collections should decide how they can be cloned or serialized.

**20. What is an Iterator ?** The [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) interface provides a number of methods that are able to iterate over any [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html). Each Java [Collection](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) contains the [iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html" \l "iterator%28%29) method that returns an [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) instance. Iterators are [capable of removing elements from the underlying collection](http://www.javacodegeeks.com/2011/05/avoid-concurrentmodificationexception.html) during the iteration. **21. What differences exist between Iterator and ListIterator ?** The differences of these elements are listed below:

* An [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) can be used to traverse the [Set](http://docs.oracle.com/javase/7/docs/api/java/util/Set.html) and [List](http://docs.oracle.com/javase/7/docs/api/java/util/List.html) collections, while the [ListIterator](http://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html) can be used to iterate only over [Lists](http://docs.oracle.com/javase/7/docs/api/java/util/List.html).
* The [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) can traverse a collection only in forward direction, while the [ListIterator](http://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html) can traverse a [List](http://docs.oracle.com/javase/7/docs/api/java/util/List.html) in both directions.
* The [ListIterator](http://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html) implements the [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) interface and contains extra functionality, such as adding an element, replacing an element, getting the index position for previous and next elements, etc.

**22. What is difference between fail-fast and fail-safe ?** The [Iterator's](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) fail-safe property works with the clone of the underlying collection and thus, it is not affected by any modification in the collection. All the collection classes in java.util package are fail-fast, while the collection classes in java.util.concurrent are fail-safe. Fail-fast iterators throw a [ConcurrentModificationException](http://examples.javacodegeeks.com/java-basics/exceptions/java-util-concurrentmodificationexception-how-to-handle-concurrent-modification-exception/), while fail-safe iterator never throws such an exception.

**23. How HashMap works in Java ?** A [HashMap in Java stores key-value pairs](http://www.javacodegeeks.com/2014/03/how-hashmap-works-in-java.html). The [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) requires a hash function and uses [hashCode](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html" \l "hashCode%28%29) and equals methods, in order to put and retrieve elements to and from the collection respectively. When the put method is invoked, the [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) calculates the hash value of the key and stores the pair in the appropriate index inside the collection. If the key exists, its value is updated with the new value. Some important characteristics of a [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) are its capacity, its load factor and the threshold resizing.

**24. What is the importance of hashCode() and equals() methods ?** In Java, a HashMap uses the hashCode and [equals](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals%28java.lang.Object%29) methods to determine the index of the key-value pair and to detect duplicates. More specifically, the hashCodemethod is used in order to determine where the specified key will be stored. Since different keys may produce the same hash valu the [equals](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals%28java.lang.Object%29) method is used, in order to determine whether the specified key actually exists in the collection or not. Therefore, the implementation of both methods is crucial to the accuracy and efficiency of the HashMap.

**25. What differences exist between HashMap and Hashtable ?** Both the [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) and [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) classes implement the Map interface and thus, have very similar characteristics. However, they differ in the following features:

* A [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) allows the existence of null keys and values, while a [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) doesn’t allow neither null keys, nor null values.
* A [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) is synchronized, while a [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) is not. Thus, [HashMap](http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) is preferred in single-threaded environments, while a [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) is suitable for multi-threaded environments.
* A HashMap provides its set of keys
* L and a Java application can ite I'llate ccc
* over them. Thus, a HashMap is fail-fast. On the other hand, a Hashtable provides an [Enumeration](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html) of its keys.
* The [Hashtable](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html) class is considered to be a legacy class.

**26. What is difference between Array and ArrayList ? When will you use Array over ArrayList ?** The [Array](http://docs.oracle.com/javase/7/docs/api/java/lang/reflect/Array.html) and [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) classes differ on the following features:

* [Arrays](http://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html) can contain primitive or objects, while an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) can contain only objects.
* [Arrays](http://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html) have fixed size, while an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) is dynamic.
* An [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html)provides more methods and features, such as [addAll](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html" \l "addAll(java.util.Collection)), [removeAll](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html" \l "removeAll(java.util.Collection)), [iterator](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html" \l "iterator()), etc.
* For a list of primitive data types, the collections use autoboxing to reduce the coding effort. However, this approach makes them slower when working on fixed size primitive data types.

**27. What is difference between ArrayList and LinkedList ?** Both the [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) and [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) classes implement the List interface, but they differ on the following features:

* An [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html) is an index based data structure backed by an [Array](http://docs.oracle.com/javase/7/docs/api/java/lang/reflect/Array.html). It provides random access to its elements with a performance equal to O(1). On the other hand, a [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) stores its data as list of elements and every element is linked to its previous and next element. In this case, the search operation for an element has execution time equal to O(n).
* The Insertion, addition and removal operations of an element are faster in a [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) compared to an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html), because there is no need of resizing an array or updating the index when an element is added in some arbitrary position inside the collection.
* A [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) consumes more memory than an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html), because every node in a [LinkedList](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html) stores two references, one for its previous element and one for its next element.

Check also our article [ArrayList vs. LinkedList](http://www.javacodegeeks.com/2013/12/arraylist-vs-linkedlist.html).

**28. What is Comparable and Comparator interface ? List their differences.** Java provides the [Comparable](http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html)interface, which contains only one method, called [compareTo](http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html" \l "compareTo(T)). This method compares two objects, in order to impose an order between them. Specifically, it returns a negative integer, zero, or a positive integer to indicate that the input object is less than, equal or greater than the existing object. Java provides the [Comparator](http://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html) interface, which contains two methods, called [compare](http://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html#compare(T,%20T)) and [equals](http://docs.oracle.com/javase/7/docs/api/java/util/Comparator.html#equals(java.lang.Object)). The first method compares its two input arguments and imposes an order between them. It returns a negative integer, zero, or a positive integer to indicate that the first argument is less than, equal to, or greater than the second. The second method requires an object as a parameter and aims to decide whether the input object is equal to the comparator. The method returns true, only if the specified object is also a comparator and it imposes the same ordering as the comparator.

**29. What is Java Priority Queue ?** The [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html) is an unbounded queue, based on a priority heap and its elements are ordered in their natural order. At the time of its creation, we can provide a Comparator that is responsible for ordering the elements of the [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html). A [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html) doesn’t allow [null values](http://examples.javacodegeeks.com/java-basics/exceptions/java-lang-nullpointerexception-how-to-handle-null-pointer-exception/), those objects that doesn’t provide natural ordering, or those objects that don’t have any comparator associated with them. Finally, the Java [PriorityQueue](http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html) is not thread-safe and it requires O(log(n)) time for its enqueing and dequeing operations.

**30. What do you know about the big-O notation and can you give some examples with respect to different data structures ?** The [Big-O notation](http://www.javacodegeeks.com/2011/04/simple-big-o-notation-post.html) simply describes how well an algorithm scales or performs in the worst case scenario as the number of elements in a data structure increases. The Big-O notation can also be used to describe other behavior such as memory consumption. Since the collection classes are actually data structures, we usually use the Big-O notation to chose the best implementation to use, based on time, memory and performance. Big-O notation can give a good indication about performance for large amounts of data.

**31. What is the tradeoff between using an unordered array versus an ordered array ?** The major advantage of an ordered array is that the search times have time complexity of O(log n), compared to that of an unordered array, which is O (n). The disadvantage of an ordered array is that the insertion operation has a time complexity of O(n), because the elements with higher values must be moved to make room for the new element. Instead, the insertion operation for an unordered array takes constant time of O(1).

**32. What are some of the best practices relating to the Java Collection framework ?**

* Choosing the right type of the collection to use, based on the application’s needs, is very crucial for its performance. For example if the size of the elements is fixed and know a priori, we shall use an [Array](http://docs.oracle.com/javase/7/docs/api/java/lang/reflect/Array.html), instead of an [ArrayList](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html).
* Some collection classes allow us to specify their initial capacity. Thus, if we have an estimation on the number of elements that will be stored, we can use it to avoid rehashing or resizing.
* Always use Generics for type-safety, readability, and robustness. Also, by using Generics you avoid the [ClassCastException](http://docs.oracle.com/javase/7/docs/api/java/lang/ClassCastException.html) during runtime.
* Use immutable classes provided by the Java Development Kit (JDK) as a key in a Map, in order to avoid the implementation of the [hashCode](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html" \l "hashCode%28%29) and equals methods for our custom class.
* Program in terms of interface not implementation.
* Return zero-length collections or arrays as opposed to returning a null in case the underlying collection is actually empty.

**33. What’s the difference between Enumeration and Iterator interfaces ?** [Enumeration](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html) is twice as fast as compared to an Iterator and uses very less memory. However, the [Iterator](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html) is much safer compared to [Enumeration](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html), because other threads are not able to modify the collection object that is currently traversed by the iterator. Also, [Iterators](http://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html)allow the caller to remove elements from the underlying collection, something which is not possible with [Enumerations](http://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html).

**34. What is the difference between HashSet and TreeSet ?** The [HashSet](http://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html) is Implemented using a hash table and thus, its elements are not ordered. The add, remove, and contains methods of a [HashSet](http://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html) have constant time complexity O(1). On the other hand, a [TreeSet](http://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html) is implemented using a tree structure. The elements in a [TreeSet](http://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html) are sorted, and thus, the add, remove, and contains methods have time complexity of O(logn).

Garbage Collectors

**35. What is the purpose of garbage collection in Java, and when is it used ?** The purpose of garbage collection is to identify and discard those objects that are no longer needed by the application, in order for the resources to be reclaimed and reused.

**36. What does System.gc() and Runtime.gc() methods do ?** These methods can be used as a hint to the JVM, in order to start a garbage collection. However, this it is up to the Java Virtual Machine (JVM) to start the garbage collection immediately or later in time.

**37. When is the finalize() called ? What is the purpose of finalization ?** The finalize method is called by the garbage collector, just before releasing the object’s memory. It is normally advised to release resources held by the object inside the finalize method.

**38. If an object reference is set to null, will the Garbage Collector immediately free the memory held by that object ?** No, the object will be available for garbage collection in the next cycle of the garbage collector.

**39. What is structure of Java Heap ? What is Perm Gen space in Heap ?** The [JVM has a heap](http://www.javacodegeeks.com/2012/07/5-tips-for-proper-java-heap-size.html) that is the runtime data area from which memory for all class instances and arrays is allocated. It is created at the JVM start-up. Heap memory for objects is reclaimed by an automatic memory management system which is known as a garbage collector. Heap memory consists of live and dead objects. Live objects are accessible by the application and will not be a subject of garbage collection. Dead objects are those which will never be accessible by the application, but have not been collected by the garbage collector yet. Such objects occupy the heap memory space until they are eventually collected by the garbage collector.

**40. What is the difference between Serial and Throughput Garbage collector ?** The throughput garbage collector uses a parallel version of the young generation collector and is meant to be used with applications that have medium to large data sets. On the other hand, the serial collector is usually adequate for most small applications (those requiring heaps of up to approximately 100MB on modern processors).

**41. When does an Object becomes eligible for Garbage collection in Java ?** A Java object is subject to garbage collection when it becomes unreachable to the program in which it is currently used.

**42. Does Garbage collection occur in permanent generation space in JVM ?** Garbage Collection does occur in PermGen space and if PermGen space is full or cross a threshold, it can trigger a full garbage collection. If you look carefully at the output of the garbage collector, you will find that PermGen space is also garbage collected. This is the reason why correct sizing of PermGen space is important to avoid frequent full garbage collections. Also check our article [Java 8: PermGen to Metaspace](http://www.javacodegeeks.com/2013/02/java-8-from-permgen-to-metaspace.html).

Exception Handling

**43. What are the two types of Exceptions in Java ? Which are the differences between them ?** Java has two types of exceptions: checked exceptions and unchecked exceptions. Unchecked exceptions do not need to be declared in a method or a constructor’s throws clause, if they can be thrown by the execution of the method or the constructor, and propagate outside the method or constructor boundary. On the other hand, checked exceptions must be declared in a method or a constructor’s throws clause. See here for tips on [Java exception handling](http://www.javacodegeeks.com/2013/07/java-exception-handling-tutorial-with-examples-and-best-practices.html).

**44. What is the difference between Exception and Error in java ?** [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) and [Error](http://docs.oracle.com/javase/7/docs/api/java/lang/Error.html) classes are both subclasses of the [Throwable](http://docs.oracle.com/javase/7/docs/api/java/lang/Throwable.html) class. The [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) class is used for exceptional conditions that a user’s program should catch. The [Error](http://docs.oracle.com/javase/7/docs/api/java/lang/Error.html) class defines exceptions that are not excepted to be caught by the user program.

**45. What is the difference between throw and throws ?** The throw keyword is used to explicitly raise a exception within the program. On the contrary, the throws clause is used to indicate those exceptions that are not handled by a method. Each method must explicitly specify which exceptions does not handle, so the callers of that method can guard against possible exceptions. Finally, multiple exceptions are separated by a comma.

**45. What is the importance of finally block in exception handling ?** A finally block will always be executed, whether or not an exception is actually thrown. Even in the case where the catch statement is missing and an exception is thrown, the finally block will still be executed. Last thing to mention is that the finally block is used to release resources like I/O buffers, database connections, etc.

**46. What will happen to the Exception object after exception handling ?** The [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html) object will be garbage collected in the next garbage collection.

**47. How does finally block differ from finalize() method ?** A finally block will be executed whether or not an exception is thrown and is used to release those resources held by the application. Finalize is a protected method of the Object class, which is called by the Java Virtual Machine (JVM) just before an object is garbage collected.

Java Applets

**48. What is an Applet ?** A java applet is program that can be included in a HTML page and be executed in a java enabled client browser. Applets are used for creating dynamic and interactive web applications.

**49. Explain the life cycle of an Applet.** An applet may undergo the following states:

* **Init**: An applet is initialized each time is loaded.
* **Start**: Begin the execution of an applet.
* **Stop**: Stop the execution of an applet.
* **Destroy**: Perform a final cleanup, before unloading the applet.

**50. What happens when an applet is loaded ?** First of all, an instance of the applet’s controlling class is created. Then, the applet initializes itself and finally, it starts running.

**51. What is the difference between an Applet and a Java Application ?** Applets are executed within a java enabled browser, but a Java application is a standalone Java program that can be executed outside of a browser. However, they both require the existence of a Java Virtual Machine (JVM). Furthermore, a Java application requires a main method with a specific signature, in order to start its execution. Java applets don’t need such a method to start their execution. Finally, Java applets typically use a restrictive security policy, while Java applications usually use more relaxed security policies.

**52. What are the restrictions imposed on Java applets ?** Mostly due to security reasons, the following restrictions are imposed on Java applets:

* An applet cannot load libraries or define native methods.
* An applet cannot ordinarily read or write files on the execution host.
* An applet cannot read certain system properties.
* An applet cannot make network connections except to the host that it came from.
* An applet cannot start any program on the host that’s executing it.

**53. What are untrusted applets ?** Untrusted applets are those Java applets that cannot access or execute local system files. By default, all downloaded applets are considered as untrusted.

**54. What is the difference between applets loaded over the internet and applets loaded via the file system ?**Regarding the case where an applet is loaded over the internet, the applet is loaded by the applet classloader and is subject to the restrictions enforced by the applet security manager. Regarding the case where an applet is loaded from the client’s local disk, the applet is loaded by the file system loader. Applets loaded via the file system are allowed to read files, write files and to load libraries on the client. Also, applets loaded via the file system are allowed to execute processes and finally, applets loaded via the file system are not passed through the byte code verifier.

**55. What is the applet class loader, and what does it provide ?** When an applet is loaded over the internet, the applet is loaded by the applet classloader. The class loader enforces the Java name space hierarchy. Also, the class loader guarantees that a unique namespace exists for classes that come from the local file system, and that a unique namespace exists for each network source. When a browser loads an applet over the net, that applet’s classes are placed in a private namespace associated with the applet’s origin. Then, those classes loaded by the class loader are passed through the verifier.The verifier checks that the class file conforms to the Java language specification . Among other things, the verifier ensures that there are no stack overflows or underflows and that the parameters to all bytecode instructions are correct.

**56. What is the applet security manager, and what does it provide ?** The applet security manager is a mechanism to impose restrictions on Java applets. A browser may only have one security manager. The security manager is established at startup, and it cannot thereafter be replaced, overloaded, overridden, or extended.

[Swing](http://examples.javacodegeeks.com/desktop-java/swing/)

**57. What is the difference between a Choice and a List ?** A Choice is displayed in a compact form that must be pulled down, in order for a user to be able to see the list of all available choices. Only one item may be selected from a Choice. A [List](http://examples.javacodegeeks.com/desktop-java/swing/jlist/create-jlist-example/) may be displayed in such a way that several List items are visible. A List supports the selection of one or more List items.

**58. What is a layout manager ?** A layout manager is the used to organize the components in a container.

**59. What is the difference between a Scrollbar and a JScrollPane ?** A [Scrollbar](http://docs.oracle.com/javase/7/docs/api/java/awt/Scrollbar.html) is a [Component](http://docs.oracle.com/javase/7/docs/api/java/awt/Component.html), but not a [Container](http://docs.oracle.com/javase/7/docs/api/java/awt/Container.html). A [ScrollPane](http://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html) is a [Container](http://docs.oracle.com/javase/7/docs/api/java/awt/Container.html). A [ScrollPane](http://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html) handles its own events and performs its own scrolling.

**60. Which Swing methods are thread-safe ?** There are only three thread-safe methods: repaint, revalidate, and invalidate.

**61. Name three Component subclasses that support painting.** The [Canvas](http://docs.oracle.com/javase/7/docs/api/java/awt/Canvas.html), [Frame](http://docs.oracle.com/javase/7/docs/api/java/awt/Frame.html), [Panel](http://docs.oracle.com/javase/7/docs/api/java/awt/Panel.html), and Applet classes support painting.

**62. What is clipping ?** Clipping is defined as the process of confining paint operations to a limited area or shape.

**63. What is the difference between a MenuItem and a CheckboxMenuItem ?** The [CheckboxMenuItem](http://docs.oracle.com/javase/7/docs/api/java/awt/CheckboxMenuItem.html) class extends the [MenuItem](http://docs.oracle.com/javase/7/docs/api/java/awt/MenuItem.html) class and supports a menu item that may be either checked or unchecked.

**64. How are the elements of a BorderLayout organized ?** The elements of a [BorderLayout](http://docs.oracle.com/javase/7/docs/api/java/awt/BorderLayout.html) are organized at the borders (North, South, East, and West) and the center of a container.

**65. How are the elements of a GridBagLayout organized ?** The elements of a [GridBagLayout](http://docs.oracle.com/javase/7/docs/api/java/awt/GridBagLayout.html) are organized according to a grid. The elements are of different sizes and may occupy more than one row or column of the grid. Thus, the rows and columns may have different sizes.

**66. What is the difference between a Window and a Frame ?** The [Frame](http://docs.oracle.com/javase/7/docs/api/java/awt/Frame.html) class extends the [Window](http://docs.oracle.com/javase/7/docs/api/java/awt/Window.html) class and defines a main application window that can have a menu bar.

**67. What is the relationship between clipping and repainting ?** When a window is repainted by the AWT painting thread, it sets the clipping regions to the area of the window that requires repainting.

**68. What is the relationship between an event-listener interface and an event-adapter class ?** An event-listener interface defines the methods that must be implemented by an event handler for a particular event. An event adapter provides a default implementation of an event-listener interface.

**69. How can a GUI component handle its own events ?** A GUI component can handle its own events, by implementing the corresponding event-listener interface and adding itself as its own event listener.

**70. What advantage do Java’s layout managers provide over traditional windowing systems ?** Java uses layout managers to lay out components in a consistent manner, across all windowing platforms. Since layout managers aren’t tied to absolute sizing and positioning, they are able to accomodate platform-specific differences among windowing systems.

**71. What is the design pattern that Java uses for all Swing components ?** The design pattern used by Java for all Swing components is the Model View Controller (MVC) pattern.

JDBC

**72. What is JDBC ?** JDBC is an abstraction layer that allows users to choose between databases. [JDBC enables developers to write database applications in Java](http://www.javacodegeeks.com/2014/03/java-8-friday-java-8-will-revolutionize-database-access.html), without having to concern themselves with the underlying details of a particular database.

**73. Explain the role of Driver in JDBC.** The JDBC Driver provides vendor-specific implementations of the abstract classes provided by the JDBC API. Each driver must provide implementations for the following classes of the java.sql package:[Connection](http://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html), [Statement](http://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html), [PreparedStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html), [CallableStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html), [ResultSet](http://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html) and [Driver](http://docs.oracle.com/javase/7/docs/api/java/sql/Driver.html).

**74. What is the purpose Class.forName method ?** This method is used to method is used to load the driver that will establish a connection to the database.

**75. What is the advantage of PreparedStatement over Statement ?** PreparedStatements are precompiled and thus, [their performance is much better](http://examples.javacodegeeks.com/core-java/sql/batch-statement-execution-example/). Also, PreparedStatement objects can be reused with different input values to their queries.

**76. What is the use of CallableStatement ? Name the method, which is used to prepare a CallableStatement.** A [CallableStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html) is used to execute stored procedures. Stored procedures are stored and offered by a database. Stored procedures may take input values from the user and may return a result. The usage of stored procedures is highly encouraged, because it offers security and modularity.The method that prepares a [CallableStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html) is the following:

|  |  |
| --- | --- |
| 1 | CallableStament.prepareCall(); |

**77. What does Connection pooling mean ?** The interaction with a database can be costly, regarding the opening and closing of database connections. Especially, when the number of database clients increases, this cost is very high and a large number of resources is consumed.A pool of database connections is obtained at start up by the application server and is maintained in a pool. A request for a connection is served by a [connection residing in the pool](http://examples.javacodegeeks.com/enterprise-java/hibernate/hibernate-connection-pool-configuration-with-c3p0-example/). In the end of the connection, the request is returned to the pool and can be used to satisfy future requests.

Remote Method Invocation (RMI)

**78. What is RMI ?** The Java Remote Method Invocation (Java RMI) is a Java API that performs the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java classes and distributed garbage collection. Remote Method Invocation (RMI) can also be seen as the process of activating a method on a remotely running object. RMI offers location transparency because a user feels that a method is executed on a locally running object. Check some [RMI Tips here](http://www.javacodegeeks.com/2013/11/two-things-to-remember-when-using-java-rmi.html).

**79. What is the basic principle of RMI architecture ?** The RMI architecture is based on a very important principle which states that the definition of the behavior and the implementation of that behavior, are separate concepts. RMI allows the code that defines the behavior and the code that implements the behavior to remain separate and to run on separate JVMs.

**80. What are the layers of RMI Architecture ?** The RMI architecture consists of the following layers:

* **Stub and Skeleton layer**: This layer lies just beneath the view of the developer. This layer is responsible for intercepting method calls made by the client to the interface and redirect these calls to a remote RMI Service.
* **Remote Reference Layer**: The second layer of the RMI architecture deals with the interpretation of references made from the client to the server’s remote objects. This layer interprets and manages references made from clients to the remote service objects. The connection is a one-to-one (unicast) link.
* **Transport layer**: This layer is responsible for connecting the two JVM participating in the service. This layer is based on TCP/IP connections between machines in a network. It provides basic connectivity, as well as some firewall penetration strategies.

**81. What is the role of Remote Interface in RMI ?** The Remote interface serves to identify interfaces whose methods may be invoked from a non-local virtual machine. Any object that is a remote object must directly or indirectly implement this interface. A class that implements a remote interface should declare the remote interfaces being implemented, define the constructor for each remote object and provide an implementation for each remote method in all remote interfaces.

**82. What is the role of the java.rmi.Naming Class ?** The java.rmi.Naming class provides methods for storing and obtaining references to remote objects in the remote object registry. Each method of the Naming class takes as one of its arguments a name that is a String in URL format.

**83. What is meant by binding in RMI ?** Binding is the process of associating or registering a name for a remote object, which can be used at a later time, in order to look up that remote object. A remote object can be associated with a name using the bind or rebind methods of the Naming class.

**84. What is the difference between using bind() and rebind() methods of Naming Class ?** The bind method bind is responsible for binding the specified name to a remote object, while the rebind method is responsible for rebinding the specified name to a new remote object. In case a binding exists for that name, the binding is replaced.

**85. What are the steps involved to make work a RMI program ?** The following steps must be involved in order for a RMI program to work properly:

* Compilation of all source files.
* Generatation of the stubs using rmic.
* Start the rmiregistry.
* Start the RMIServer.
* Run the client program.

**86. What is the role of stub in RMI ?** A stub for a remote object acts as a client’s local representative or proxy for the remote object. The caller invokes a method on the local stub, which is responsible for executing the method on the remote object. When a stub’s method is invoked, it undergoes the following steps:

* It initiates a connection to the remote JVM containing the remote object.
* It marshals the parameters to the remote JVM.
* It waits for the result of the method invocation and execution.
* It unmarshals the return value or an exception if the method has not been successfully executed.
* It returns the value to the caller.

**87. What is DGC ? And how does it work ?** DGC stands for Distributed Garbage Collection. Remote Method Invocation (RMI) uses DGC for automatic garbage collection. Since RMI involves remote object references across JVM’s, garbage collection can be quite difficult. DGC uses a reference counting algorithm to provide automatic memory management for remote objects.

**88. What is the purpose of using RMISecurityManager in RMI ?** RMISecurityManager provides a security manager that can be used by RMI applications, which use downloaded code. The class loader of RMI will not download any classes from remote locations, if the security manager has not been set.

**89. Explain Marshalling and demarshalling.** When an application wants to pass its memory objects across a network to another host or persist it to storage, the in-memory representation must be converted to a suitable format. This process is called marshalling and the revert operation is called demarshalling.

**90. Explain Serialization and Deserialization.** Java provides a mechanism, called object serialization where an object can be represented as a sequence of bytes and includes the object’s data, as well as information about the object’s type, and the types of data stored in the object. Thus, serialization can be seen as a way of flattening objects, in order to be stored on disk, and later, read back and reconstituted. Deserialisation is the reverse process of converting an object from its flattened state to a live object.

Servlets

**91. What is a Servlet ?** [The servlet](http://examples.javacodegeeks.com/enterprise-java/servlet/sample-java-servlet/) is a Java programming language class used to process client requests and generate dynamic web content. Servlets are mostly used to process or store data submitted by an HTML form, provide dynamic content and manage state information that does not exist in the stateless HTTP protocol.

**92. Explain the architechure of a Servlet.** The core abstraction that must be implemented by all servlets is the javax.servlet.Servlet interface. Each servlet must implement it either directly or indirectly, either by extending javax.servlet.GenericServlet or javax.servlet.http.HTTPServlet. Finally, each servlet is able to serve multiple requests in parallel using multithreading.

**93. What is the difference between an Applet and a Servlet ?** An Applet is a client side java program that runs within a Web browser on the client machine. On the other hand, a servlet is a server side component that runs on the web server.An applet can use the user interface classes, while a servlet does not have a user interface. Instead, a servlet waits for client’s HTTP requests and generates a response in every request.

**94. What is the difference between GenericServlet and HttpServlet ?** GenericServlet is a generalized and protocol-independent servlet that implements the Servlet and ServletConfig interfaces. Those servlets extending the GenericServlet class shall override the service method. Finally, in order to develop an HTTP servlet for use on the Web that serves requests using the HTTP protocol, your servlet must extend the HttpServlet instead. Check [Servlet examples here](http://examples.javacodegeeks.com/tag/servlet/).

**95. Explain the life cycle of a Servlet.** On every client’s request, the Servlet Engine loads the servlets and invokes its init methods, in order for the servlet to be initialized. Then, the Servlet object handles all subsequent requests coming from that client, by invoking the service method for each request separately. Finally, the servlet is removed by calling the server’s destroy method.

**96. What is the difference between doGet() and doPost() ?** doGET: The GET method appends the name-value pairs on the request’s URL. Thus, there is a limit on the number of characters and subsequently on the number of values that can be used in a client’s request. Furthermore, the values of the request are made visible and thus, sensitive information must not be passed in that way. doPOST: The POST method overcomes the limit imposed by the GET request, by sending the values of the request inside its body. Also, there is no limitations on the number of values to be sent across. Finally, the sensitive information passed through a POST request is not visible to an external client.

**97. What is meant by a Web Application ?** A Web application is a dynamic extension of a Web or application server. There are two types of web applications: presentation-oriented and service-oriented. A presentation-oriented Web application generates interactive web pages, which contain various types of markup language and dynamic content in response to requests. On the other hand, a service-oriented web application implements the endpoint of a web service. In general, a Web application can be seen as a collection of servlets installed under a specific subset of the server’s URL namespace.

**98. What is a Server Side Include (SSI) ?** Server Side Includes (SSI) is a simple interpreted server-side scripting language, used almost exclusively for the Web, and is embedded with a servlet tag. The most frequent use of SSI is to include the contents of one or more files into a Web page on a Web server. When a Web page is accessed by a browser, the Web server replaces the servlet tag in that Web page with the hyper text generated by the corresponding servlet.

**99. What is Servlet Chaining ?** Servlet Chaining is the method where the output of one servlet is sent to a second servlet. The output of the second servlet can be sent to a third servlet, and so on. The last servlet in the chain is responsible for sending the response to the client.

**100. How do you find out what client machine is making a request to your servlet ?** The ServletRequest class has functions for finding out the IP address or host name of the client machine. getRemoteAddr() gets the IP address of the client machine and getRemoteHost() gets the host name of the client machine. See example [here](http://examples.javacodegeeks.com/enterprise-java/servlet/get-client-s-address-and-hostname-in-servlet/).

**101. What is the structure of the HTTP response ?** The HTTP response consists of three parts:

* **Status Code**: describes the status of the response. It can be used to check if the request has been successfully completed. In case the request failed, the status code can be used to find out the reason behind the failure. If your servlet does not return a status code, the success status code, HttpServletResponse.SC\_OK, is returned by default.
* **HTTP Headers**: they contain more information about the response. For example, the headers may specify the date/time after which the response is considered stale, or the form of encoding used to safely transfer the entity to the user. See [how to retrieve headers in Servlet here](http://examples.javacodegeeks.com/enterprise-java/servlet/get-all-request-headers-in-servlet/).
* **Body**: it contains the content of the response. The body may contain HTML code, an image, etc. The body consists of the data bytes transmitted in an HTTP transaction message immediately following the headers.

**102. What is a cookie ? What is the difference between session and cookie ?** [A cookie](http://examples.javacodegeeks.com/core-java/net/urlconnection/get-cookies-from-http-connection/) is a bit of information that the Web server sends to the browser. The browser stores the cookies for each Web server in a local file. In a future request, the browser, along with the request, sends all stored cookies for that specific Web server.The differences between session and a cookie are the following:

* The session should work, regardless of the settings on the client browser. The client may have chosen to disable cookies. However, the sessions still work, as the client has no ability to disable them in the server side.
* The session and cookies also differ in the amount of information the can store. The HTTP session is capable of storing any Java object, while a cookie can only store String objects.

**103. Which protocol will be used by browser and servlet to communicate ?** The browser communicates with a servlet by using the HTTP protocol.

**104. What is HTTP Tunneling ?** HTTP Tunneling is a technique by which, communications performed using various network protocols are encapsulated using the HTTP or HTTPS protocols. The HTTP protocol therefore acts as a wrapper for a channel that the network protocol being tunneled uses to communicate. The masking of other protocol requests as HTTP requests is HTTP Tunneling.

**105. What’s the difference between sendRedirect and forward methods ?** The sendRedirect method creates a new request, while the forward method just forwards a request to a new target. The previous request scope objects are not available after a redirect, because it results in a new request. On the other hand, the previous request scope objects are available after forwarding. FInally, in general, the sendRedirect method is considered to be slower compare to the forward method.

**106. What is URL Encoding and URL Decoding ?** The URL encoding procedure is responsible for replacing all the spaces and every other extra special character of a URL, into their corresponding Hex representation. In correspondence, URL decoding is the exact opposite procedure.

JSP

**107. What is a JSP Page ?** A Java Server Page (JSP) is a text document that contains two types of text: static data and JSP elements. Static data can be expressed in any text-based format, such as HTML or XML. JSP is a technology that mixes static content with dynamically-generated content. See [JSP example here](http://examples.javacodegeeks.com/enterprise-java/jsp/sample-jsp-java-server-page/).

**108. How are the JSP requests handled ?** On the arrival of a JSP request, the browser first requests a page with a .jsp extension. Then, the Web server reads the request and using the JSP compiler, the Web server converts the JSP page into a servlet class. Notice that the JSP file is compiled only on the first request of the page, or if the JSP file has changed.The generated servlet class is invoked, in order to handle the browser’s request. Once the execution of the request is over, the servlet sends a response back to the client. See [how to get Request parameters in a JSP](http://examples.javacodegeeks.com/enterprise-java/jsp/get-request-parameter-in-jsp-page/).

**109. What are the advantages of JSP ?** The advantages of using the JSP technology are shown below:

* JSP pages are dynamically compiled into servlets and thus, the developers can easily make updates to presentation code.
* JSP pages can be pre-compiled.
* JSP pages can be easily combined to static templates, including HTML or XML fragments, with code that generates dynamic content.
* Developers can offer customized JSP tag libraries that page authors access using an XML-like syntax.
* Developers can make logic changes at the component level, without editing the individual pages that use the application’s logic.

**110. What are Directives ? What are the different types of Directives available in JSP ?** Directives are instructions that are processed by the JSP engine, when the page is compiled to a servlet. Directives are used to set page-level instructions, insert data from external files, and specify custom tag libraries. Directives are defined between < %@ and % >.The different types of directives are shown below:

* Include directive: it is used to include a file and merges the content of the file with the current page.
* Page directive: it is used to define specific attributes in the JSP page, like error page and buffer.
* Taglib: it is used to declare a custom tag library which is used in the page.

**111. What are JSP actions ?** JSP actions use constructs in XML syntax to control the behavior of the servlet engine. JSP actions are executed when a JSP page is requested. They can be dynamically inserted into a file, re-use JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.Some of the available actions are listed below:

* jsp:include – includes a file, when the JSP page is requested.
* jsp:useBean – finds or instantiates a JavaBean.
* jsp:setProperty – sets the property of a JavaBean.
* jsp:getProperty – gets the property of a JavaBean.
* jsp:forward – forwards the requester to a new page.
* jsp:plugin – generates browser-specific code.

**112. What are Scriptlets ?** In Java Server Pages (JSP) technology, a scriptlet is a piece of Java-code embedded in a JSP page. The scriptlet is everything inside the tags. Between these tags, a user can add any valid scriplet.

**113. What are Decalarations ?** Declarations are similar to variable declarations in Java. Declarations are used to declare variables for subsequent use in expressions or scriptlets. To add a declaration, you must use the sequences to enclose your declarations.

**114. What are Expressions ?** A JSP expression is used to insert the value of a scripting language expression, converted into a string, into the data stream returned to the client, by the web server. Expressions are defined between <% = and %>tags.

**115. What is meant by implicit objects and what are they ?** JSP implicit objects are those Java objects that the JSP Container makes available to developers in each page. A developer can call them directly, without being explicitly declared. JSP Implicit Objects are also called pre-defined variables.The following objects are considered implicit in a JSP page:

* application
* page
* request
* response
* session
* exception
* out
* config
* pageContext

[Home](https://www.javacodegeeks.com/) » [Java](https://www.javacodegeeks.com/category/java/) » [Enterprise Java](https://www.javacodegeeks.com/category/java/enterprise-java/) » 69 Spring Interview Questions and Answers – The ULTIMATE List (PDF Download)

### About Theodora Fragkouli



Theodora has graduated from Computer Engineering and Informatics Department in the University of Patras. She also holds a Master degree in Economics from the National and Technical University of Athens. During her studies she has been involved with a large number of projects ranging from programming and software engineering to telecommunications, hardware design and analysis. She works as a junior Software Engineer in the telecommunications sector where she is mainly involved with projects based on Java and Big Data technologies.

https://www.javacodegeeks.com/wp-content/themes/jarida/images/author_site.png

# 69 Spring Interview Questions and Answers – The ULTIMATE List

**Last updated Oct. 3, 2016**

This is a summary of some of the most important questions concerning the Spring Framework, that you may be asked to answer in an interview or in an interview test procedure! There is no need to worry for your next interview test, because Java Code Geeks are here for you!

The majority of the things you may be asked is collected in the list below. All core modules, from basic Spring functionality such as Spring Beans, up to [Spring MVC](http://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-hello-world-example/)framework are presented and described in short. After checking the interview questions, you should check our [Spring Tutorials page](http://www.javacodegeeks.com/tutorials/java-tutorials/enterprise-java-tutorials/spring-tutorials/).

So, let’s go…!

## Table of Contents

* [Spring overview](https://www.javacodegeeks.com/2014/05/spring-interview-questions-and-answers.html#1)
* [Dependency Injection](https://www.javacodegeeks.com/2014/05/spring-interview-questions-and-answers.html#2)
* [Spring Beans](https://www.javacodegeeks.com/2014/05/spring-interview-questions-and-answers.html#3)
* [Spring Annotations](https://www.javacodegeeks.com/2014/05/spring-interview-questions-and-answers.html#4)
* [Spring Data Access](https://www.javacodegeeks.com/2014/05/spring-interview-questions-and-answers.html#5)
* [Spring Aspect Oriented Programming (AOP)](https://www.javacodegeeks.com/2014/05/spring-interview-questions-and-answers.html#6)
* [Spring Model View Controller (MVC)](https://www.javacodegeeks.com/2014/05/spring-interview-questions-and-answers.html#7)

## Spring overview

### 1. What is Spring?

Spring is an open source development framework for [Enterprise Java](http://www.javacodegeeks.com/tutorials/java-tutorials/enterprise-java-tutorials/). The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make Java EE development easier to use and promote good programming practice by enabling a [POJO-based programming model](http://www.javacodegeeks.com/2012/09/how-to-write-better-pojo-services.html).

### 2. What are benefits of Spring Framework?

* **Lightweight:**Spring is lightweight when it comes to size and transparency. The basic version of spring framework is around 2MB.
* **Inversion of control (IOC):** Loose coupling is achieved in Spring, with the [Inversion of Control technique](http://www.javacodegeeks.com/2011/08/what-is-dependency-inversion-is-it-ioc.html). The objects give their dependencies instead of creating or looking for dependent objects.
* **Aspect oriented (AOP):** [Spring supports Aspect oriented programming](http://www.javacodegeeks.com/2011/01/aspect-oriented-programming-spring-aop.html) and separates application business logic from system services.
* **Container:** Spring contains and manages the life cycle and configuration of application objects.
* **MVC Framework:** Spring’s web framework is a well-designed [web MVC framework](http://www.javacodegeeks.com/2011/02/spring-mvc-development-tutorial.html), which provides a great alternative to web frameworks.
* **Transaction Management:** Spring provides a consistent transaction management interface that can scale down to a local transaction and scale up to global transactions (JTA).
* **Exception Handling:** Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO) into consistent, unchecked exceptions.

### 3. Which are the Spring framework modules?

The basic modules of the Spring framework are :

* Core module
* Bean module
* Context module
* Expression Language module
* [JDBC module](http://examples.javacodegeeks.com/enterprise-java/spring/jdbc/spring-jdbctemplate-example/)
* [ORM module](http://examples.javacodegeeks.com/enterprise-java/spring/jpaorm/spring-hibernate-mysql-and-maven-showcase/)
* OXM module
* Java Messaging Service(JMS) module
* Transaction module
* Web module
* Web-Servlet module
* Web-Struts module
* Web-Portlet module

### 4. Explain the Core Container (Application context) module

This is the basic Spring module, which provides the fundamental functionality of the Spring framework. BeanFactory is the heart of any spring-based application. Spring framework was built on the top of this module, which makes the Spring container.

### 5. BeanFactory – BeanFactory implementation example

A BeanFactory is an implementation of the factory pattern that applies Inversion of Control to separate the application’s configuration and dependencies from the actual application code.

The most commonly used BeanFactory implementation is the XmlBeanFactory class.

### 6. XMLBeanFactory

The most useful one is org.springframework.beans.factory.xml.XmlBeanFactory, which loads its beans based on the definitions contained in an XML file. This container reads the configuration metadata from an XML file and uses it to create a fully configured system or application.

### 7. Explain the AOP module

The AOP module is used for developing aspects for our Spring-enabled application. Much of the support has been provided by the AOP Alliance in order to ensure the interoperability between [Spring and other AOP frameworks](http://www.javacodegeeks.com/2014/02/applying-aspect-oriented-programming.html). This module also introduces metadata programming to Spring.

### 8. Explain the JDBC abstraction and DAO module

With the [JDBC abstraction and DAO module](http://examples.javacodegeeks.com/enterprise-java/spring/jdbc/spring-jdbctemplate-example/) we can be sure that we keep up the database code clean and simple, and prevent problems that result from a failure to close database resources. It provides a layer of meaningful exceptions on top of the error messages given by several database servers. It also makes use of Spring’s AOP module to provide transaction management services for objects in a Spring application.

### 9. Explain the object/relational mapping integration module

Spring also supports for using of an [object/relational mapping (ORM) too](http://www.javacodegeeks.com/2011/12/persistence-layer-with-spring-31-and_14.html)l over straight JDBC by providing the ORM module. Spring provides support to tie into several popular ORM frameworks, including [Hibernate](http://www.javacodegeeks.com/2010/05/jboss-42x-spring-3-jpa-hibernate.html), JDO, and [iBATIS SQL Maps](http://www.javacodegeeks.com/2012/02/mybatis-3-spring-integration-tutorial.html). Spring’s transaction management supports each of these ORM frameworks as well as JDBC.

### 10. Explain the web module

The [Spring web module](http://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-hello-world-example/) is built on the application context module, providing a context that is appropriate for web-based applications. This module also contains support for several web-oriented tasks such as transparently handling multipart requests for file uploads and programmatic binding of request parameters to your business objects. It also contains integration support with Jakarta Struts.

### 11. Explain the Spring MVC module

MVC framework is provided by Spring for building web applications. Spring can easily be integrated with other MVC frameworks, but [Spring’s MVC framework](http://www.javacodegeeks.com/2012/09/spring-adding-spring-mvc-part-1.html) is a better choice, since it uses IoC to provide for a clean separation of controller logic from business objects. With Spring MVC you can declaratively bind request parameters to your business objects.

### 12. Spring configuration file

Spring configuration file is an XML file. This file contains the classes information and describes how these classes are configured and introduced to each other.

### 13. What is Spring IoC container?

The Spring IoC is responsible for creating the objects,managing them (with dependency injection (DI)), wiring them together, configuring them, as also managing their complete lifecycle.

### 14. What are the benefits of IOC?

IOC or dependency injection minimizes the amount of code in an application. It makes easy to test applications, since no singletons or JNDI lookup mechanisms are required in unit tests. Loose coupling is promoted with minimal effort and least intrusive mechanism. IOC containers support eager instantiation and lazy loading of services.

### 15. What are the common implementations of the ApplicationContext?

The **FileSystemXmlApplicationContext** container loads the definitions of the beans from an XML file. The full path of the XML bean configuration file must be provided to the constructor.  
The **ClassPathXmlApplicationContext** container also loads the definitions of the beans from an XML file. Here, you need to set CLASSPATH properly because this container will look bean configuration XML file in CLASSPATH.  
The **WebXmlApplicationContext:** container loads the XML file with definitions of all beans from within a web application.

### 16. What is the difference between Bean Factory and ApplicationContext?

Application contexts provide a means for resolving text messages, a generic way to load file resources (such as images), they can publish events to beans that are registered as listeners. In addition, operations on the container or beans in the container, which have to be handled in a programmatic fashion with a bean factory, can be handled declaratively in an application context. The application context implements MessageSource, an interface used to obtain localized messages, with the actual implementation being pluggable.

### 17. What does a Spring application look like?

* An interface that defines the functions.
* The implementation that contains properties, its setter and getter methods, functions etc.,
* [Spring AOP](http://examples.javacodegeeks.com/enterprise-java/spring/aop/spring-aop-example/)
* The Spring configuration XML file.
* Client program that uses the function

## Dependency Injection

### 18. What is Dependency Injection in Spring?

[Dependency Injection](http://www.javacodegeeks.com/2014/02/dependency-injection-options-for-java.html), an aspect of Inversion of Control (IoC), is a general concept, and it can be expressed in many different ways.This concept says that you do not create your objects but describe how they should be created. You don’t directly connect your components and services together in code but describe which services are needed by which components in a configuration file. A container (the IOC container) is then responsible for hooking it all up.

### 19. What are the different types of IoC (dependency injection)?

* **Constructor-based dependency injection:** Constructor-based DI is accomplished when the container invokes a class constructor with a number of arguments, each representing a dependency on other class.
* **Setter-based dependency injection:** Setter-based DI is accomplished by the container calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

### 20. Which DI would you suggest Constructor-based or setter-based DI?

You can use both Constructor-based and Setter-based Dependency Injection. The best solution is using constructor arguments for mandatory dependencies and setters for optional dependencies.

## Spring Beans

### 21. What are Spring beans?

The [Spring Beans](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-3-bean-reference-example/) are Java Objects that form the backbone of a Spring application. They are instantiated, assembled, and managed by the Spring IoC container. These beans are created with the configuration metadata that is supplied to the container, for example, in the form of XML <bean/> definitions.

Beans defined in spring framework are singleton beans. There is an attribute in bean tag named "singleton" if specified true then bean becomes singleton and if set to false then the bean becomes a prototype bean. By default it is set to true. So, all the beans in spring framework are by default singleton beans.

### 22. What does a Spring Bean definition contain?

A Spring Bean definition contains all configuration metadata which is needed for the container to know how to create a bean, its lifecycle details and its dependencies.

### 23. How do you provide configuration metadata to the Spring Container?

There are three important methods to provide configuration metadata to the Spring Container:

* XML based configuration file.
* Annotation-based configuration
* [Java-based configuration](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-3-java-config-example/)

### 24. How do you define the scope of a bean?

When defining a <bean> in Spring, we can also declare a scope for the bean. It can be defined through the scope attribute in the bean definition. For example, when Spring has to produce a new bean instance each time one is needed, the bean’s scope attribute to be prototype. On the other hand, when the same instance of a bean must be returned by Spring every time it is needed, the the bean scope attribute must be set to singleton.

### 25. Explain the bean scopes supported by Spring

There are five scoped provided by the Spring Framework supports following five scopes:

* In **singleton** scope, Spring scopes the bean definition to a single instance per Spring IoC container.
* In **prototype** scope, a single bean definition has any number of object instances.
* In **request** scope, a bean is defined to an HTTP request. This scope is valid only in a web-aware Spring ApplicationContext.
* In **session**scope, a bean definition is scoped to an HTTP session. This scope is also valid only in a web-aware Spring ApplicationContext.
* In **global-session** scope, a bean definition is scoped to a global HTTP session. This is also a case used in a web-aware Spring ApplicationContext.

The default scope of a Spring Bean is Singleton.

### 26. Are Singleton beans thread safe in Spring Framework?

No, singleton beans are not thread-safe in Spring framework.

### 27. Explain Bean lifecycle in Spring framework

* The spring container finds the bean’s definition from the XML file and instantiates the bean.
* Spring populates all of the properties as specified in the bean definition (DI).
* If the bean implements BeanNameAware interface, spring passes the bean’s id to setBeanName() method.
* If Bean implements BeanFactoryAware interface, spring passes the beanfactory to setBeanFactory() method.
* If there are any bean BeanPostProcessors associated with the bean, Spring calls postProcesserBeforeInitialization()method.
* If the bean implements IntializingBean, its afterPropertySet() method is called. If the bean has init method declaration, the specified initialization method is called.
* If there are any BeanPostProcessors associated with the bean, their postProcessAfterInitialization() methods will be called.
* If the bean implements DisposableBean, it will call the destroy() method.

### 28. Which are the important beans lifecycle methods? Can you override them?

There are two important bean lifecycle methods. The first one is setup which is called when the bean is loaded in to the container. The second method is the teardown method which is called when the bean is unloaded from the container.  
The bean tag has two important attributes (init-method and destroy-method) with which you can define your own custom initialization and destroy methods. There are also the correspondive annotations(@PostConstruct and @PreDestroy).

### 29. What are inner beans in Spring?

When a bean is only used as a property of another bean it can be declared as an inner bean. Spring’s XML-based configuration metadata provides the use of <bean/> element inside the <property/> or <constructor-arg/> elements of a bean definition, in order to define the so-called inner bean. Inner beans are always anonymous and they are always scoped as prototypes.

### 30. How can you inject a Java Collection in Spring?

Spring offers the following types of [collection configuration elements](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-collections-list-set-map-and-properties-example/):

* The <list> type is used for injecting a list of values, in the case that duplicates are allowed.
* The <set> type is used for wiring a set of values but without any duplicates.
* The <map> type is used to inject a collection of name-value pairs where name and value can be of any type.
* The <props> type can be used to inject a collection of name-value pairs where the name and value are both Strings.

### 31. What is bean wiring?

Wiring, or else bean wiring is the case when beans are combined together within the Spring container. When wiring beans, the Spring container needs to know what beans are needed and how the container should use dependency injection to tie them together.

### 32. What is bean auto wiring?

The Spring container is able to [autowire relationships](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-autowire-example/) between collaborating beans. This means that it is possible to automatically let Spring resolve collaborators (other beans) for a bean by inspecting the contents of the BeanFactory without using <constructor-arg> and <property> elements.

### 33. Explain different modes of auto wiring?

The autowiring functionality has five modes which can be used to instruct Spring container to use autowiring for dependency injection:

* **no:** This is default setting. Explicit bean reference should be used for wiring.
* **byName:** When autowiring byName, the Spring container looks at the properties of the beans on which autowireattribute is set to byName in the XML configuration file. It then tries to match and wire its properties with the beans defined by the same names in the configuration file.
* **byType:** When autowiring by datatype, the Spring container looks at the properties of the beans on which autowireattribute is set to byType in the XML configuration file. It then tries to match and wire a property if its type matches with exactly one of the beans name in configuration file. If more than one such beans exist, a fatal exception is thrown.
* **constructor:** This mode is similar to byType, but type applies to constructor arguments. If there is not exactly one bean of the constructor argument type in the container, a fatal error is raised.
* **autodetect:**Spring first tries to wire using autowire by constructor, if it does not work, Spring tries to autowire by byType.

### 34. Are there limitations with autowiring?

Limitations of autowiring are:

* **Overriding:**You can still specify dependencies using <constructor-arg> and <property> settings which will always override autowiring.
* **Primitive data types:** You cannot autowire simple properties such as primitives, Strings, and Classes.
* **Confusing nature:** Autowiring is less exact than explicit wiring, so if possible prefer using explicit wiring.

### 35. Can you inject null and empty string values in Spring?

Yes, you can.

## Spring Annotations

### 36. What is Spring Java-Based Configuration? Give some annotation example.

[Java based configuration](http://www.javacodegeeks.com/2013/04/spring-java-configuration.html) option enables you to write most of your Spring configuration without XML but with the help of few Java-based annotations.  
An example is the @Configuration annotation, that indicates that the class can be used by the Spring IoC container as a source of bean definitions. Another example is the@Bean annotated method that will return an object that should be registered as a bean in the Spring application context.

### 37. What is Annotation-based container configuration?

An alternative to XML setups is provided by annotation-based configuration which relies on the bytecode metadata for wiring up components instead of angle-bracket declarations. Instead of using XML to describe a bean wiring, the developer moves the configuration into the component class itself by using annotations on the relevant class, method, or field declaration.

### 38. How do you turn on annotation wiring?

Annotation wiring is not turned on in the Spring container by default. In order to use annotation based wiring we must enable it in our Spring configuration file by configuring <context:annotation-config/> element.

### 39. @Required annotation

This annotation simply indicates that the affected bean property must be populated at configuration time, through an explicit property value in a bean definition or through autowiring. The container throws BeanInitializationException if the affected bean property has not been populated.

### 40. @Autowired annotation

The @Autowired annotation provides more fine-grained control over where and how autowiring should be accomplished. It can be used to autowire bean on the setter method just like @Required annotation, on the constructor, on a property or pn methods with arbitrary names and/or multiple arguments.

### 41. @Qualifier annotation

When there are more than one beans of the same type and only one is needed to be wired with a property, the @Qualifierannotation is used along with @Autowired annotation to remove the confusion by specifying which exact bean will be wired.

## Spring Data Access

### 42. How can JDBC be used more efficiently in the Spring framework?

When using the Spring JDBC framework the burden of resource management and error handling is reduced. So developers only need to write the statements and queries to get the data to and from the database. JDBC can be used more efficiently with the help of a template class provided by Spring framework, which is the JdbcTemplate (example [here](http://examples.javacodegeeks.com/enterprise-java/spring/jdbc/spring-jdbctemplate-example/)).

### 43. JdbcTemplate

JdbcTemplate class provides many convenience methods for doing things such as converting database data into primitives or objects, executing prepared and callable statements, and providing custom database error handling.

### 44. Spring DAO support

The [Data Access Object (DAO) support in Spring](http://www.javacodegeeks.com/2012/09/spring-dao-and-service-layer.html) is aimed at making it easy to work with data access technologies like JDBC, Hibernate or JDO in a consistent way. This allows us to switch between the persistence technologies fairly easily and to code without worrying about catching exceptions that are specific to each technology.

### 45. What are the ways to access Hibernate by using Spring?

There are two ways to access Hibernate with Spring:

* Inversion of Control with a Hibernate Template and Callback.
* Extending HibernateDAOSupport and Applying an AOP Interceptor node.

### 46. ORM’s Spring support

Spring supports the following ORM’s:

* Hibernate
* iBatis
* JPA (Java Persistence API)
* TopLink
* JDO (Java Data Objects)
* OJB

### 47. How can we integrate Spring and Hibernate using HibernateDaoSupport?

Use Spring’s SessionFactory called LocalSessionFactory. The integration process is of 3 steps:

* Configure the Hibernate SessionFactory
* Extend a DAO Implementation from HibernateDaoSupport
* Wire in Transaction Support with AOP

### 48. Types of the transaction management Spring support

Spring supports two types of transaction management:

* **Programmatic transaction management:** This means that you have managed the transaction with the help of programming. That gives you extreme flexibility, but it is difficult to maintain.
* **Declarative transaction management:** This means you separate [transaction management from the business code](http://www.javacodegeeks.com/2011/09/spring-declarative-transactions-example.html). You only use annotations or XML based configuration to manage the transactions.

### 49. What are the benefits of the Spring Framework’s transaction management?

* It provides a consistent programming model across different transaction APIs such as JTA, JDBC, Hibernate, JPA, and JDO.
* It provides a simpler API for programmatic transaction management than a number of complex transaction APIs such as JTA.
* It supports declarative transaction management.
* It integrates very well with Spring’s various data access abstractions.

### 50. Which Transaction management type is more preferable?

Most users of the Spring Framework choose declarative transaction management because it is the option with the least impact on application code, and hence is most consistent with the ideals of a non-invasive lightweight container. Declarative transaction management is preferable over programmatic transaction management though it is less flexible than programmatic transaction management, which allows you to control transactions through your code.

## Spring Aspect Oriented Programming (AOP)

### 51. Explain AOP

[Aspect-oriented programming](http://www.javacodegeeks.com/2014/02/applying-aspect-oriented-programming.html), or AOP, is a programming technique that allows programmers to modularize crosscutting concerns, or behavior that cuts across the typical divisions of responsibility, such as logging and transaction management.

### 52. Aspect

The core construct of AOP is the aspect, which encapsulates behaviors affecting multiple classes into reusable modules. It ia a module which has a set of APIs providing cross-cutting requirements. For example, a logging module would be called AOP aspect for logging. An application can have any number of aspects depending on the requirement. In Spring AOP, aspects are implemented using regular classes annotated with the @Aspect annotation (@AspectJ style).

### 53. What is the difference between concern and cross-cutting concern in Spring AOP

The Concern is behavior we want to have in a module of an application. A Concern may be defined as a functionality we want to implement.  
The cross-cutting concern is a concern which is applicable throughout the application and it affects the entire application. For example, logging, [security](http://www.javacodegeeks.com/2013/04/spring-aop-in-security-controlling-creation-of-ui-components-via-aspects.html) and data transfer are the concerns which are needed in almost every module of an application, hence they are cross-cutting concerns.

### 54. Join point

The join point represents a point in an application where we can plug-in an AOP aspect. It is the actual place in the application where an action will be taken using Spring AOP framework.

### 55. Advice

The advice is the actual action that will be taken either before or after the method execution. This is actual piece of code that is invoked during the program execution by the Spring AOP framework.

Spring aspects can work with five kinds of advice:

* **before:** Run advice before the a method execution.
* **after:** Run advice after the a method execution regardless of its outcome.
* **after-returning:** Run advice after the a method execution only if method completes successfully.
* **after-throwing:** Run advice after the a method execution only if method exits by throwing an exception.
* **around:** Run advice before and after the advised method is invoked.

### 56. Pointcut

The pointcut is a set of one or more joinpoints where an advice should be executed. You can specify pointcuts using expressions or patterns.

### 57. What is Introduction?

An Introduction allows us to add new methods or attributes to existing classes.

### 58. What is Target object?

The target object is an object being advised by one or more aspects. It will always be a proxy object. It is also referred to as the advised object.

### 59. What is a Proxy?

A proxy is an object that is created after applying advice to a target object. When you think of client objects the target object and the proxy object are the same.

### 60. What are the different types of AutoProxying?

* BeanNameAutoProxyCreator
* DefaultAdvisorAutoProxyCreator
* Metadata autoproxying

### 61. What is Weaving? What are the different points where weaving can be applied?

Weaving is the process of linking aspects with other application types or objects to create an advised object.  
Weaving can be done at compile time, at load time, or at runtime.

### 62. Explain XML Schema-based aspect implementation?

In this implementation case, aspects are implemented using regular classes along with XML based configuration.

### 63. Explain annotation-based (@AspectJ based) aspect implementation

This implementation case (@AspectJ based implementation) refers to a style of declaring aspects as regular Java classes annotated with Java 5 annotations.

## Spring Model View Controller (MVC)

### 64. What is Spring MVC framework?

Spring comes with a [full-featured MVC framework for building web applications](http://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-hello-world-example/). Although Spring can easily be integrated with other MVC frameworks, such as Struts, Spring’s MVC framework uses IoC to provide a clean separation of controller logic from business objects. It also allows to declaratively bind request parameters to business objects.

### 65. DispatcherServlet

The Spring Web MVC framework is designed around a DispatcherServlet that handles all the HTTP requests and responses.

### 66. WebApplicationContext

The WebApplicationContext is an extension of the plain ApplicationContext that has some extra features necessary for web applications. It differs from a normal ApplicationContext in that it is capable of resolving themes, and that it knows which servlet it is associated with.

### 67. What is Controller in Spring MVC framework?

Controllers provide access to the application behavior that you typically define through a service interface. Controllers interpret user input and transform it into a model that is represented to the user by the view. Spring implements a controller in a very abstract way, which enables you to create a wide variety of controllers.

### 68. @Controller annotation

The @Controller annotation indicates that a particular class serves the role of a controller. Spring does not require you to extend any controller base class or reference the Servlet API.

### 69. @RequestMapping annotation

@RequestMapping annotation is used to map a URL to either an entire class or a particular handler method.