Java Interview Questions

Contents

Chapter 1

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**. We analyze each feature separately in the following sections.

1.1 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 for more details and examples on encapsulation.

1.2 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.

1.3 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.

1.4 Abstraction

Abstraction 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.

1.5 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.

Chapter 2

General Questions about Java

2.1 What is JVM ? Why is Java called the Platform Independent Programming Language?

A Java virtual machine (JVM) is a process virtual machine that can execute Java bytecode. 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.

2.2 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, and Java Debugger), in order for a user to develop, compile and execute Java applications.

2.3 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, 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.

2.4 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.

2.5 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 between the primitive types and their corresponding object wrapper classes. For example, the compiler converts an int to an Integer, a double to a Double, and so on. If the conversion goes the other way, this operation is called `unboxing`.

2.6 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.

2.7 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. 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.

2.8 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.

2.9 What is the difference between an Interface and an Abstract class ?

Java provides and supports the creation both of abstract classes 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.

2.10 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.

Chapter 3

Java Threads

3.1 What is the difference between processes and threads ?

A process is an execution of a program, while a Thread is a single execution sequence within a process. A process can contain multiple threads. A Thread is sometimes called a lightweight process.

3.2 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 to be created:

- A class may extend the Thread class.
- A class may implement the Runnable interface.
- An application can use the Executor framework, in order to create a thread pool.

The Runnable interface is preferred, as it does not require an object to inherit the Thread 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.

3.3 Explain the available thread states in a high-level.

During its execution, a thread can reside in one of the following states:

- **Runnable**: A thread becomes ready to run, but does not necessarily start running immediately.
- **Running**: The processor is actively executing the thread code.
- **Waiting**: A thread is in a blocked state waiting for some external processing to finish.
- **Sleeping**: The thread is forced to sleep.
- **Blocked on I/O**: Waiting for an I/O operation to complete.
- **Blocked on Synchronization**: Waiting to acquire a lock.
- **Dead**: The thread has finished its execution.

3.4 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).

3.5 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.

3.6 What's a deadlock ?

A condition that occurs when two processes are waiting for each other to complete, before proceeding. The result is that both processes wait endlessly.

3.7 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.

Chapter 4

Java Collections

4.1 What are the basic interfaces of Java Collections Framework ?

Java Collections Framework 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, which represents a group of objects known as its elements.
- Set, which is a collection that cannot contain duplicate elements.
- List, which is an ordered collection and can contain duplicate elements.
- Map, which is an object that maps keys to values and cannot contain duplicate keys.

4.2 Why Collection doesn't extend Cloneable and Serializable interfaces ?

The Collection interface specifies groups of objects known as elements. Each concrete implementation of a Collection 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.

4.3 What is an Iterator ?

The Iterator interface provides a number of methods that are able to iterate over any Collection. Each Java Collection contains the iterator method that returns an Iterator instance. Iterators are capable of removing elements from the underlying collection during the iteration.

4.4 What differences exist between Iterator and ListIterator ?

The differences of these elements are listed below:

- An Iterator can be used to traverse the Set and List collections, while the ListIterator can be used to iterate only over Lists.
- The Iterator can traverse a collection only in forward direction, while the ListIterator can traverse a List in both directions.
- The ListIterator implements the Iterator interface and contains extra functionality, such as adding an element, replacing an element, getting the index position for previous and next elements, etc.

4.5 What is difference between fail-fast and fail-safe ?

The Iterator's 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`, while fail-safe iterator never throws such an exception.

4.6 How HashMap works in Java ?

A HashMap in Java stores key-value pairs. The HashMap requires a hash function and uses hashCode and equals methods, in order to put and retrieve elements to and from the collection respectively. When the put method is invoked, the HashMap 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 are its capacity, its load factor and the threshold resizing.

4.7 What is the importance of hashCode() and equals() methods ?

In Java, a HashMap uses the hashCode and equals methods to determine the index of the key-value pair and to detect duplicates. More specifically, the hashCode method is used in order to determine where the specified key will be stored. Since different keys may produce the same hash value, the equals 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.

4.8 What differences exist between HashMap and Hashtable ?

Both the HashMap and Hashtable classes implement the Map interface and thus, have very similar characteristics. However, they differ in the following features:

- A HashMap allows the existence of null keys and values, while a Hashtable doesn't allow neither null keys, nor null values.
- A Hashtable is synchronized, while a HashMap is not. Thus, HashMap is preferred in single-threaded environments, while a Hashtable is suitable for multi-threaded environments.
- A HashMap provides its set of keys and a Java application can iterate over them. Thus, a HashMap is fail-fast. On the other hand, a Hashtable provides an Enumeration of its keys.
- The Hashtable class is considered to be a legacy class.

4.9 What is difference between Array and ArrayList ? When will you use Array over ArrayList ?

The Array and ArrayList classes differ on the following features:

- Arrays can contain primitive or objects, while an ArrayList can contain only objects.
- Arrays have fixed size, while an ArrayList is dynamic.
- An ArrayList provides more methods and features, such as addAll, removeAll, 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.

4.10 What is difference between ArrayList and LinkedList ?

Both the ArrayList and LinkedList classes implement the List interface, but they differ on the following features:

- An ArrayList is an index based data structure backed by an Array. It provides random access to its elements with a performance equal to O(1). On the other hand, a LinkedList 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 compared to an ArrayList, 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 consumes more memory than an ArrayList, because every node in a LinkedList stores two references, one for its previous element and one for its next element.

Check also our article ArrayList vs. LinkedList.

4.11 What is Comparable and Comparator interface ? List their differences.

Java provides the Comparable interface, which contains only one method, called compareTo. 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 interface, which contains two methods, called compare and equals. 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.

4.12 What is Java Priority Queue ?

The PriorityQueue 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. A PriorityQueue doesn't allow null values, those objects that doesn't provide natural ordering, or those objects that don't have any comparator associated with them. Finally, the Java PriorityQueue is not thread-safe and it requires O(log(n)) time for its enqueing and dequeing operations.

4.13 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 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.

4.14 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).

4.15 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, instead of an ArrayList.
- 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 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 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.

4.16 What's the difference between Enumeration and Iterator interfaces ?

Enumeration is twice as fast as compared to an Iterator and uses very less memory. However, the Iterator is much safer compared to Enumeration, because other threads are not able to modify the collection object that is currently traversed by the iterator. Also, Iteratorsallow the caller to remove elements from the underlying collection, something which is not possible with Enumerations.

4.17 What is the difference between HashSet and TreeSet ?

The HashSet is Implemented using a hash table and thus, its elements are not ordered. The add, remove, and contains methods of a HashSet have constant time complexity O(1). On the other hand, a TreeSet is implemented using a tree structure. The elements in a TreeSet are sorted, and thus, the add, remove, and contains methods have time complexity of O(logn).

Chapter 5

Garbage Collectors

5.1 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.

5.2 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.

5.3 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.

5.4 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.

5.5 What is structure of Java Heap ? What is Perm Gen space in Heap ?

The JVM has a heap 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.

5.6 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).

5.7 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.

5.8 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.

Chapter 6

Exception Handling

6.1 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.

6.2 What is the difference between Exception and Error in java ?

Exception and Error classes are both subclasses of the Throwable class. The Exception class is used for exceptional conditions that a user's program should catch. The Error class defines exceptions that are not excepted to be caught by the user program.

6.3 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.

6.4 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.

6.5 What will happen to the Exception object after exception handling ?

The Exception object will be garbage collected in the next garbage collection.

6.6 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.

Chapter 7

Java Applets

7.1 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.

7.2 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.

7.3 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.

7.4 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.

7.5 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.

7.6 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.

7.7 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.

7.8 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.

7.9 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.

Chapter 8

Swing

8.1 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 may be displayed in such a way that several List items are visible. A List supports the selection of one or more List items.

8.2 What is a layout manager ?

A layout manager is the used to organize the components in a container.

8.3 What is the difference between a Scrollbar and a JScrollPane ?

A Scrollbar is a Component, but not a Container. A ScrollPane is a Container. A ScrollPane handles its own events and performs its own scrolling.

8.4 Which Swing methods are thread-safe ?

There are only three thread-safe methods: repaint, revalidate, and invalidate.

8.5 Name three Component subclasses that support painting.

The Canvas, Frame, Panel, and Applet classes support painting.

8.6 What is clipping ?

Clipping is defined as the process of confining paint operations to a limited area or shape.

8.7 What is the difference between a MenuItem and a CheckboxMenuItem ?

The CheckboxMenuItem class extends the MenuItem class and supports a menu item that may be either checked or unchecked.

8.8 How are the elements of a BorderLayout organized ?

The elements of a BorderLayout are organized at the borders (North, South, East, and West) and the center of a container.

8.9 How are the elements of a GridBagLayout organized ?

The elements of a GridBagLayout 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.

8.10 What is the difference between a Window and a Frame ?

The Frame class extends the Window class and defines a main application window that can have a menu bar.

8.11 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.

8.12 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.

8.13 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.

8.14 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.

8.15 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.

Chapter 9

JDBC

9.1 What is JDBC ?

JDBC is an abstraction layer that allows users to choose between databases. JDBC enables developers to write database applications in Java, without having to concern themselves with the underlying details of a particular database.

9.2 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, Statement, PreparedStatement, CallableStatement, ResultSet and Driver.

9.3 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.

9.4 What is the advantage of PreparedStatement over Statement ?

PreparedStatements are precompiled and thus, their performance is much better. Also, PreparedStatement objects can be reused with different input values to their queries.

9.5 What is the use of CallableStatement ? Name the method, which is used to prepare a CallableStatement.

A CallableStatement 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 is the following: `CallableStament.prepareCall();`

9.6 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. In the end of the connection, the request is returned to the pool and can be used to satisfy future requests.

Chapter 10

Remote Method Invocation (RMI)

10.1 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.

10.2 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.

10.3 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.

10.4 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.

10.5 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.

10.6 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.

10.7 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.

10.8 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.

10.9 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.

10.10 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.

10.11 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.

10.12 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.

10.13 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.

Chapter 11

Servlets

11.1 What is a Servlet ?

The 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.

11.2 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.

11.3 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.

11.4 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.

11.5 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.

11.6 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.

11.7 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.

11.8 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.

11.9 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.

11.10 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.

11.11 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.
- **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.

11.12 What is a cookie ? What is the difference between session and cookie ?

A cookie 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.

11.13 Which protocol will be used by browser and servlet to communicate ?

The browser communicates with a servlet by using the HTTP protocol.

11.14 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.

11.15 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.

11.16 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.

Chapter 12

JSP

12.1 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.

12.2 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.

12.3 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.

12.4 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.

12.5 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.

12.6 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.

12.7 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.

12.8 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.

12.9 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`