Multithreading, Concurrency, and Thread Basics Questions

1. **Is it possible to make array volatile in Java?**

Answer: Yes, it is possible to make an array volatile in Java, but only the reference which is pointing to an array, not the whole array. Therefore, if one thread changes the reference variable points to another array, which will provide a volatile guarantee.

However, if several threads are altering particular array elements, there won't be any happens before assurance provided by the volatile modifier for such modification.

If the purpose is to provide memory visibility guarantee for individual indices of the array, volatile is of no practical use for you. Instead, you must rely on an alternative mechanism for thread-safety in that particular case e.g. use of a synchronized keyword.

2. **From the two, which would be easier to write: synchronization code for ten threads or two threads?**

Answer: Both will have the same level of complexity in terms of writing the code because synchronization is independent of a number of threads.

Although the choice of synchronization could be subject to a number of threads because the number of thread present more conflict.

Therefore, you would opt for an advanced synchronization technique e.g. lock stripping, which requires more intricate code and proficiency.

3. **How would you call wait() method? Would you use if block or loop, and why?**

Answer: wait() method should always be called in loop. It is likely that, until thread gets CPU to start running again, the condition may not hold. Therefore, it is always advised to check condition in loop before continuing.

4. **What is defined as false sharing in the context of multithreading?**

Answer: False sharing is known to be one of the familiar performance issues on multi-core systems, whereby for each process it has its local cache.

False sharing can be hard to identify since the thread may be retrieving completely different global variables that occur to be fairly close together in memory.

Similar to many other concurrency issues, the main way to avoid false sharing is to carefully review your code and supporting your data structure with the size of a cache line.

5. **What is defined as busy spin, and why should you use it?**

Answer: Busy spin is known as one of the techniques to wait for events without freeing CPU. This is often done to avoid losing data in CPU cache, which could get lost if the thread is paused and resumed in some other core.

As a result, if you are working on a low latency system where your order processing thread isn't in any particular order, rather than sleeping or calling wait(), you can just loop and then review the queue for new messages.

This is only valuable if you need to wait for a very little amount of time e.g. in microseconds or nanoseconds. LMAX Disruptor frameworks, a high-performance inter-thread messaging library has a BusySpinWait Strategy, which is centered on this model and uses a busy spin loop for EventProcessors waiting on the barrier.

LMAX Disruptor frameworks, a high-performance inter-thread messaging library has a BusySpinWait Strategy, which is centered on this model and uses a busy spin loop for EventProcessors waiting on the barrier.

6. **How do you take thread dump in Java?**

Answer: By using **kill -3 PID** in Linux, where PID is the process id of Java process, you are able to take a thread dump of Java application. In Windows, you can press **Ctrl + Break.**

This will instruct JVM to print thread dump in standard out and it may go to console or log file depending on your application configuration.

7. **Is Swing thread-safe?**

*Answer***:** No, Swing is not thread-safe. You aren't able to update Swing components e.g. JTable, JList or Jpanel from any thread. In fact, they must be updated from a GUI or AWT thread. This is why Swing's provide

In fact, they must be updated from a GUI or AWT thread. This is why Swing's provide invokeAndWait() and invokeLater() method to request GUI update from alternative threads.

These methods put update requests in AWT threads queue and wait for the update or return straight away for an asynchronous update.

8. **Describe what a thread-local variable is in Java**

Answer: Thread-local variables are variables restricted to a thread. It is like thread's own copy which is not shared between a multitude of threads. Java offers a ThreadLoval class to upkeep thread-local variables. This is one of the many ways to guarantee thread-safety.

However, it is important to be mindful while using thread local variable in a controlled environment e.g. with web servers where worker thread outlives any application variable.

Any thread local variable which is not taken away once its work is done can hypothetically cause a memory leak in Java application.

9. **What is the difference between sleep and wait in Java?**

Answer: Both are used to pause thread that is currently running, however, sleep() is meant for short pause because it does not release lock, while wait() is meant for conditional wait.

This is why it releases lock, which can then be developed by a different thread to alter the condition of which it is waiting.

10. **What is defined as an immutable object? How would you create an immutable object in Java?**

Answer: Immutable objects are defined as those whose state cannot be changed once it has been made, any alteration will result in a new object e.g. String, Integer, and other wrapper class.

Basic Java Interview Questions, and Data Types

1. **What is the right data type to represent a price in Java?**

Answer: If memory is not a concern and performance is not critical, BigDecimal will be the right data type represent a price in Java. If not, double with predefined precision.

2. **How would you convert bytes to String?**

Answer: To convert bytes to String, you would use Strong constructor which accepts byte[].

However, you should be mindful of the right character encoding otherwise the platform's default character encoding will be used, which may not necessarily be the same.

3. **Is it possible to cast an int value into a byte variable? What would happen if the value of int is larger than byte?**

Answer: Yes, it is possible but int is 32 bit long in Java, while byte is 8 bit long in Java. Therefore when you can cast an int to byte higher, 24 bits are gone and a byte can only hold a value between -128 to 128.

4. **Which class contains method: Cloneable or Object?**

Answer: java.lang.Cloneable is marker interface and does not contain at all any method. Clone method is well-defined in the object class.

Remember that clone() is a native method, therefore it is applied in C or C++ or any other native programming language.

5. **Is ++ operator thread-safe in Java?**

Answer: ++ is not thread-safe in Java because it involves multiple commands such as reading a value, implicating it, and then storing it back into memory.

This can be overlapped between multiple threads.

6. **Is it possible to store a double value in a long variable without casting?**

Answer: No, it is not possible to store a double value into a long variable without casting since the range of double is more, meaning you would need to type cast.

7. **Which one will take more memory: an int or Integer?**

Answer: An integer object will take more memory as it stores meta data overhead about the object. An int is primitive, therefore it takes less space.

8. **Why is String immutable in Java?**

Answer: String is immutable in Java since Java designer thought that String will be greatly used, making it immutable. It lets some optimisation easy sharing, and same String object between multiple clients.

A key step in that direction was the idea of putting away String literals in String pool. The aim was to moderate temporary String object by sharing them and in order to share, they must have to be from immutable class.

It is worth noting, that it isn't possible to share a mutable project with two parties which are unfamiliar to each other.

9. **Is it possible to use String in the switch case?**

Answer: Yes, this is possible from Java 7 onward. String can be used in switch case, but it is just syntactic sugar. Internal string hash code is used for the switch.

10. **What is constructor chaining in Java?**

Answer: Constructor chaining in Java is when you call one constructor from another. This generally occurs when you have multiple, overloaded constructor in the class.

JVM Internals and Garbage Collection Interview Questions

1. **What is the size of int in 64-bit JVM?**

Answer: The size of an int variable is constant in Java, it is always 32-bit regardless of platform. This means the size of primitive int is identical in both 32-bit and 64-bit Java Virtual Machine.

2. **What is the size of an int variable in 32-bit and 64-bit JVM?**

Answer: The size of int is identical in both 32-bit and 64-bit JVM, and it is always 32-bits or 4 bytes.

3. **How does WeakHashMap work?**

Answer: WeakHashMap operates like a normal HashMap but uses WeakReference for keys. Meaning if the key object does not devise any reference then both key/value mapping will become appropriate for garbage collection.

4. **How do you identify if JVM is 32-bit or 64-bit from Java Program?**

Answer: This can be identified by checking some system properties such as sun.arch.data.model or os.arch.

5. **What is the maximum heap size of 32-bit and 64-bit JVM?**

Answer: In theory, the maximum heap memory you can assign to a 32-bitJVM is 2^32, which is 4GB, but practically the bounds are much smaller.

It also depends on operating systems e.g. from 1.5GB in Windows to almost 3GB in Solaris. 64-bit JVM allows you to stipulate larger heap size, hypothetically 2^64, which is quite large but practically you can specify heap space up to 100GBs.

6. **Explain the difference between JRE, JDK, JVM, and JIT.**

Answer: JRE is an abbreviation of Java Runtime Environment that consist of sets of files needed by JVM throughout the runtime.

JVM is an abbreviation of Java Virtual Machine which delivers the runtime environment for collected Java Bytecode. JVM is in control of the conversion of the byte code into machine readable code.

JDK is an abbreviation for Java Development Kit which contains JRE including development tools for the purpose of development. JDK is required to write and execute a Java code.

JIT is an abbreviation of Just in Time compilation and this helps to improve the performance of Java application by converting Java byte code into native code when they cross certain threshold i.e. the mostly hot code is transformed into native code.

7. **Explain Java Heap Space and Garbage collection.**

Answer: When a Java process has started using Java command, memory is distributed to it. Part of this memory is used to build heap space, which is used to assign memory to objects every time they are formed in the program.

Part of this memory is used to build heap space, which is used to assign memory to objects every time they are formed in the program.

Garbage collection is the procedure inside JVM which reclaims memory from dead objects for future distribution.

8. **Can you guarantee the garbage collection process?**

Answer: No, the garbage collection cannot be guaranteed, though you can make a request using System.gc() or Runtime.gc() method.

9. **How do you locate memory usage from a Java program? How much of the percent is used?**

Answer: You have the ability to use memory related methods from java.lang.Runtime class to get the free memory, total memory and maximum heap memory in Java.

From using these methods, you are able to find out how much percent of the heap is used and how much heap space is outstanding.

Runtime.freeMemory() return the amount of free memory in bytes, Runtime.totalMemory() returns the total memory in bytes and Runtime.maxMemory() returns the maximum memory in bytes.

10. **What is the difference between Stack and Heap in Java?**

Answer: Stack and Heap are different memory areas in the JVM, and they are used for different purposes.

The stack is usually much smaller than heap memory and also isn’t shared amongst multiple threads, but heap is shared among all threads in JVM.

Basic Java Concepts Interview Questions

1. **What is the difference between ‘a == b’ and ‘a.equals(b)’?**

Answer: The ‘a = b’ does object reference matching if both **a** and **b** are an object and only return true if both are pointing to the same object in the heap space. However, a.equals(b) is used for logical mapping and it is likely from an object to supersede this method to provide logical equality.

For example, String class overrides this equals() method so that you can associate two Strings, which are not the same object but covers the same letters.

2. **What is a.hashCode() used for? How is it related to a.equals(b)?**

Answer: hashCode() method returns an int hash value corresponding to an object. It is used in hash based collection classes e.g. HashTable, HashMap, LinkedHashMap. It is very closely related to equals() method.

According to the Java specification, two objects which are identical to each other using equals() method needs to have the same hash code.

3. **What is the difference between final, finalize and finally?**

Answer: Final is a modifier which you can apply to variable, methods, and classes. If you create a variable final, this means its value cannot be changed once initialized.

Finalize is a method, which is called just before an object is a garbage collected, allowing it a final chance to save itself, but the call to finalize is not definite.

Finally is a keyword which is used in exception handling, along with try and catch. The finally block is always implemented regardless of whether an exception is thrown from try block or not.

4. **What is a compile time constant in Java? What is the risk of using it?**

Answer: Public static final variables are also known as the compile time constant, the public is optional there. They are substituted with actual values at compile time because compiler recognizes their value up-front, and also recognize that it cannot be altered during runtime.

One of the issues is that if you choose to use a public static final variable from in-house or a third party library, and their value changed later, then your client will still be using the old value even after you deploy a new version of JARs.

This can be avoided by ensuring you compile your program when you upgrade dependency JAR files.

Java Collections Framework Interview Questions

1. **What is the difference between List, Set, Map and Queue in Java?**

Answer: List, Set, and Map are three significant interfaces of Java collection framework.

Set provides an unordered collection of unique objects i.e. set does not allow duplicates, while Map provides a data structure based on key value pair and hashing.

The difference between List and Set interface in Java is that List allows duplicates while Set does not allow duplicates. All implementation of Set honor this agreement. Map holds two objects per entry e.g. key and value, and it may contain duplicate values but keys are always unique.

One more difference between List and Set is that List is an ordered collection, List's contract maintains insertion order or element. Set is an unordered collection, therefore you get no assurance on which order element will be stored.

Nevertheless some of the set implementation e.g. LinkedHashSet retains order.

A queue is also ordered, but you will only ever touch elements at one end. All elements get inserted at the 'end' and removed from the 'beginning' (or head) of the queue.

You are able to find out how many elements are in the queue, but you are not able to find out what, for example, the 'third' element is.

2. **What is the difference between poll() and remove() method?**

Answer: Both poll() and remove() take out the object from the Queue but if poll() fails, then it returns null. However, if remove() fails, it throws exception.

3. **What is the difference between LinkedHashMap and PriorityQueue in Java?**

Answer: PriorityQueue guarantees that the lowest or highest priority element always remains at the head of the queue. However, LinkedHashMap maintains the order on which elements are inserted.

When you repeat over a PriorityQueue, iterator does not promise any order but iterator of LinkedHashMap does promise the order on which elements are put in.

4. **What is the difference between ArrayList and LinkedList in Java?**

Answer: The main difference between them is that ArrayList is supported by array data structure, supports random access. LinkedList is backed by linked list data structure and doesn't support random access.

5. **How do you print Array in Java?**

Answer: You can print an array by using the Arrays.toString() and Arrays.deepToString() method. Since Array does not implement to String() by itself, just passing an array to System.out.printIn() will not print its content but Array.to.String will print each element.

6. **Is LinkedList in Java a doubly or singly linked list?**

Answer: LinkedList is a doubly linked list, and you can review the code in JDK. In Eclipse, you are able to use the shortcut, **Ctrl + T** to directly open this class in editor.

7. **What is the difference between Hashtable and HashMap?**

Answer: There are several differences between the two classes, including:

- Hashtable is a legacy class and current from JDK 1, HashMap was introduced and added later.

- Hashtable is synchronized and slower whereas HashMap is not synchronized and faster.
- Hashtable does not allow null keys but HashMap allows one null key.

8. **How does HashSet work internally in Java?**

Answer: HashSet is internally implemented using a HashMap. Since a Map needs a key and value, a default value is used for all keys. Like HashMap, HashSet does not allow identical keys and only one null key – you are only able to store one null object in HashSet.

9. **Is it possible for two unequal objects to have the same hashcode?**

Answer: Yes, two unequal objects can have the same hashcode. This is why collision can occur in hashmap. The equal hashcode contract only says that two equal objects must have the identical hashcode, but there is no indication to say anything about the unequal object.

10. **What is the difference between Comparator and Comparable in Java?**

Answer: The comparable interface is used to define the natural order of object while Comparator is used to describe custom order. Comparable can always be one, but it is possible to have multiple comparators to define a customised order for objects.

Java IO and NIO Interview Questions

From a Java interview point of view, IO is very important. Ideally, you should have a good knowledge of old Java IO, NIO, and NIO2. Additionally, it is worth having knowledge of some operating systems and disk IO fundamentals. The following are some frequently asked questions regarding Java IO:

1. **What is the byte order of ByteBuffer?**

Answer: The byte order is used when reading or writing multibyte values, and when creating buffers that are views of this byte buffer. The order of a new byte buffer is always BIG_ENDIAN.

2. **What is the difference between direct buffer and non-direct buffer in Java?**

Answer: Byte buffer is one of the important class of Java NIO API. This was initially introduced in java.nio package on JDK 1.4. It lets you function on heap byte arrays as well as with direct memory, which occurs outside the JVM.

It lets you function on heap byte arrays as well as with direct memory, which occurs outside the JVM.

The main difference between direct and non-direct byte buffers are their memory location, non-direct byte buffers are just a wrapper around byte array and they reside in Java Heap memory.

Meanwhile, direct byte buffer is outside of JVM and memory is not assigned from the heap.

A byte buffer is either direct or non-direct. Given a direct byte buffer, the Java Virtual Machine will make a best effort to complete native I/O operations directly upon it.

It will try to evade copying the buffer's content to (or from) an intermediate buffer before (or after) each invocation of one of the underlying operating system's native I/O operations.

3. **What is the memory mapped buffer in Java?**

Answer: Java IO has been considerably fast after the introduction of NIO and memory mapped file offers fastest IO operation possible in Java.

A key advantage of memory mapped file is that operating system is responsible for reading and writing and even if your program malfunctioned just after writing into memory. OS will take care of writing content to file.

4. **What is the difference between TCP and UDP protocol?**

Answer: TCP and UDP are two transport layer protocols, which are widely used on the internet for transferring data from one host to another.

TCP is a connection-oriented protocol, whereas UDP is a connectionless protocol. Therefore, a connection is recognized between client and server before they can send data.

With UDP being a connectionless protocol, the point-to-point connection is not recognized before sending messages. For that reason, UDP is more fit for multicast distribution of the message.

There are many differences between TCP and UDP, but during the Java interview it is worth mentioning that TCP is connection oriented, dependable, sluggish, offers definite delivery and preserves the order of messages, while UDP is connectionless, unpredictable, no ordering assurance, but a fast protocol.

TCP is also much higher than UDP, as it transfers more meta data per packer than UDP.

Additionally, the header size of TCP is 20 bytes, compared to 8 bytes header of UDP. If you are in a position where you can't afford to lose any messages, use TCP. If you need high-speed data transmission, where loss of a single packet is acceptable, use UDP.

If you need high-speed data transmission, where loss of a single packet is acceptable, use UDP.

Date, Time, and Calendar Interview Questions in Java

1. **Is SimpleDateFormat safe to use in the multithreaded program?**

Answer: No, DataFormat and all its implementation including SimpleDateFormat is not thread-safe, hence should not be used in the multithreaded program until external thread-safety measures are applied e.g. confining SimpleDateFormat object into ThreadLocal variable.

If you do not do that, you will get wrong results while analyzing or configuring dates in Java.

2. **How would you format a data in Java? i.e. in the DDMMYYY format.**

Answer: This can be done by using either SimpleDateFormat class or java-time library to format a date in Java. DateFormat class lets you format the date on many common formats.

3. **How do you show the time zone in a formatted date In Kava?**

Answer: You are able to put time zone information in formatted Date using z attribute of DateFormat class.

4. **What is the difference between java.util.Date and java.sql.Date in Java?**

Answer: Both are known as Date class, but there is some difference between java.util.Date and jaca.sql.Date, for example, Former is used whenever a Date is required in Java application while later is used to read and store DATE SQL type from the database.

Another significant difference is java.util.Date stores both date and time values, while java.sql.date only stores date information, without any time apart. According to the Javadoc specification, java.sql.date is a thin wrapper around a millisecond value that lets JDBC distinguish as an SQL DATE value.

To fit in with the meaning of SQL DATE, the millisecond values bound by a java.sql.Date instance must be 'normalised'. However, setting the hours, minutes, seconds, and milliseconds to zero in the specific time zone with which the case is connected.

Unit Testing JUnit Interview Questions

1. **How do you test static method?**

Answer: PowerMock library can be used to test static methods in Java.

2. **How do you test a method for an exception using JUnit?**

Answer: One part of unit testing a Java method is checking exception thrown by that method. In a Java unit test, it should really prove correct exception thrown in exceptional case and no exception would be thrown in normal case.

In order to test any Java method for throwing exception in Junit4, you need to make sure that argument provided to that method, from the test must result in expected exception, otherwise JUnit test will fail.

A testing method called speed(), returns speed as distance/time, but before calculating speed it checks whether time and distance are positive or negative and if time is zero or negative it throws IllegalArgumentException.

3. **What is the difference between @Before and @BeforeClass annotation?**

Answer: The code marker @Before is executed before each test, while @BeforeClass runs once before the entire test fixture. If your text class has ten tests, @Before code will be executed ten times, but @BeforeClass will be executed only once.

In general, you would use @BeforeClass when numerous tests are required to share the same computationally expensive setup code. Starting a database connection falls into this category. You are able to move code from @BeforeClass into @Before, but your test run may be delayed.

@BeforeClass is run as static initializer, therefore it will run before the class instance of your test fixture is created.

Java Interview Questions from OOP and Design Patterns

1. **What is the difference between abstract class and interface in Java?**

Answer: There are various differences between abstract class and interface in Java, however, the most significant would be Java's restriction on permitting a class to extend just one class but lets it implement multiple interfaces.

An abstract class is good to define default behavior for a family of class, but the interface is good to outline which is then used to leverage Polymorphism.

2. **Explain Liskov Substitution Principle.**

Answer: According to the Liskov Substitution Principle, Subtypes must be appropriate for super type i.e. methods or functions which use super class type must be able to work with object of sub class with no issues.

LSP is closely related to Single Responsibility Principle and Interface Segregation Principle. If a class has more functionality, then subclass might not upkeep some of the functionality and does violate LSP.

To follow the LSP SOLID Design Principle, derived class or subclass must improve functionality, but not lessen them. LSP represent 'L' on SOLID acronym.

3. **What is Law of Demeter violation and why does it matter?**

Answer: Java is centered on application programming and structuring code. If you have good knowledge of common coding best practices, patterns, and what not do, then you can write good code.

Law of Demeter suggests you 'talk to friends and not stranger', therefore used to reduce coupling between classes.

4. **What is Adapter pattern and when would you use it?**

Answer: Adapter pattern provides interface conversion. For example, if your client is using some interface but you have something else, you can write an adapter to bridge them together.

5. **What is an abstract class? How is it different from an interface, and why would you use it?**

Answer: An abstract class is a class which can have state, code, and implementation, but an interface is a contract which is totally abstract.

The abstract class and inheritance equally take precautions that most of the code is written with abstract and high-level classes, therefore it can influence Inheritance and Polymorphism.

6. **Which is better: constructor injection or setter dependency injection?**

Answer: Both have their advantages and disadvantages. Constructor injection guaranteed that class will be initialized with all its dependency. However, setter dependency injection offers flexibility to set an optional dependency.

Setter dependency injection is also more understandable if you are using XML file to define dependency. A general rule of thumb is to use constructor injection for compulsory dependency and use setter injection for non-compulsory dependency.

7. **What is the difference between Adapter and Decorator pattern?**

Answer: Though they are both similar, the difference is the intent of each pattern. The adapter pattern is used to bridge the gap in the middle of two interfaces, but Decorator pattern is used to add an extra level of indirection to support distributed, controlled or intelligent access.

8. **What is Template method pattern?**

Answer: Template pattern provides an outline of an algorithm and lets you configure or customise its steps. For example, you are able to view a sorting algorithm as a template to sort object.

It describes steps for sorting but lets you arrange how to associate them using Comparable or something comparable in another language. This pattern uses double dispatch to supplement another level of indirection.

9. **What is the difference between Inheritance and Composition?**

Answer: Both allow code reuse, however, Composition is more flexible than Inheritance because it lets you switch to a different implementation at run-time. Code written using Composition is also better and easier to test than code including inheritance hierarchies.

10. **Explain overloading and overriding in Java.**

Answer: They both let you write two methods of different functionality but with the same name, but overloading accumulates time activity while overriding is run-time activity. You can overload a method in the same class, however, you can only override a method in child classes.

It is worth noting that Inheritance is necessary for overriding.

Miscellaneous Java Interview Questions

1. **What is the difference between nester static class and top level class?**

Answer: A public top level class must have the same name as the name of the source file – there is no obligation for nested static class.

A nester static class is at all times inside a top level class and you need to use the name of the top-level class to refer nested static class. For example, HashMap.Entry is a nester static class, whereby HashMap is a top level class and Entry is a nested static class.

2. **Is it possible to write a regular expression to check if String is a number?**

Answer: A numeric String is only able to contain digits i.e. 0-9 and +/- sign. By using this information you can write following regular expression to check if given String is number or not.

3. **What is the difference between throw and throws in Java?**

Answer: The throw is used to actually throw an instance of java.lang.throwable class, meaning you can throw both Error and Exception using throw keyword.

However, throws is used as part of method declaration and indicate which kind of exceptions are thrown by this method, so that its caller can handle them.

It is compulsory to assert any unhandled checked exception in throws clause in Java.

4. **What is the difference between Serilizable and Externalizable in Java?**

Answer: Serializable interface is used to make Java classes serializable so that they can be transmitted over the network or their state can be kept on disk. However, it influences default serialization built-in JVM, which is pricey, fragile, and unsecured.

Externalizable lets you fully control the Serialization process, identify a customer binary format and enhance security measure.

5. **What is the difference between DOM and SAX parser in Java?**

Answer: DOM parser loads the whole XML into memory to create a tree based DOM model. This helps it quickly locate nodes and make a change in the structure of XML. SAX parser is an event based parser and does not load the whole XML into memory.

For this reason, DOM is quicker than SAX but it needs more memory and is not fitting to parse large XML files.

6. **Explain 5 features introduced in JDK 1.7.**

Answer:

- try-with-resource statements free you from closing streams, and resources when you are finished with them, Java automatically closes this.
- Fork-join pool to implement something like the Map-reduce pattern in Java, which allows String variable, and literal into switch statements.
- Diamond operator for improving type inference, so there is no need to assert generic type on the right-hand side of variable declaration any longer, meaning the results are more clear and the code is more concise.
- Improved exception handling, which lets you catch multiple exceptions in the same catch block.

7. **Explain 5 features introduced in JDK 1.8.**

Answer:

- **Lambda Expression**: This allows you to pass an anonymous function as object.
- **Stream API**: Allows you to take advantage of multiple cores of modern CPU and lets you write concise code.
- **Date and Time API**: There is a solid and easy to use date and time library in JDK.
- **Extension Methods**: You can include static and default method into your interface.
- **Repeated Annotation**: This lets you apply the same annotation multiple times on a type.

8. **What is the difference between Maven and ANT in Java?**

Answer: Both are a build tool and used to create a Java application build but Maven is more advanced. It provides a standard structure for Java projects based on the ‘convention over configuration’ concept and routinely manages dependencies (JAR files on which your application is dependent) for Java application.