**Java Tricky Questions**

**405. Is there any difference between a = a + b and a += b expressions?**

When we add two integral variables e.g. variables of type byte, short, or int in Java, then they are first promoted to int type, and then addition happens.

The += operator implicitly casts the result of addition into the type of variable used to hold the result.
What happens when you put return statement or System.exit () on try or catch block? Will finally block execute?

It is a popular tricky Java interview question. Most of the programmers think that no matter what the finally block will always execute. This question challenges that concept by putting a return statement in the try or catch block or calling System.exit() from try or catch block.

You can answer by saying that finally block executes even if we put a return statement in the try block or catch block. But finally block does not execute if you call System.exit() from try or catch block.

**406. What does the expression 1.0 / 0.0 return? Will there be any compilation error?**

Double class is the source of many tricky interview questions. You may know about the double primitive type and Double class. But while doing floating point arithmetic some people don't pay enough attention to Double.INFINITY, NaN, and -0.0. There are rules that govern the floating point arithmetic calculations involving Double.

The answer to this question is that 1.0 / 0.0 will compile successfully. And it will not throw ArithmeticException. It will just return Double.INFINITY.

**407. Can we use multiple main methods in multiple classes?**

Yes. When we start an application in Java, we just mention the class name to be run to java command. The JVM looks for the main method only in the class whose name is passed to java command. Therefore, there is no conflict amongst the multiple classes having main method.

**408. Does Java allow you to override a private or static method?**

The question is tricky but the answer is very simple. You cannot override a private or static method in Java. If we create a similar method with same return type and same method arguments in child class, then it will hide the superclass method. This is known as method hiding.

Also, you cannot override a private method in sub class because Private method is not visible even in a subclass. Therefore, what you can do is to create another private method with the same name in the child class.

So in both the cases, it is not method overriding. It is either method hiding or a new method.

**409. What happens when you put a key object in a HashMap that is already present?**

In a HashMap there are buckets in which objects are stored. Key objects with same HashCode go to same bucket.

If you put the same key again in a HashMap, then it will replace the old mapping because HashMap doesn't allow duplicate keys. The same key will have same HashCode as previous key object. Due to same HashCode, it will be stored at the same position in the bucket.

**410. How can you make sure that N threads can access N resources without deadlock?**

This question checks your knowledge of writing multi-threading code. If you have experience with deadlock and race conditions, you can easily answer this.

The answer is that by resource ordering you can prevent deadlock. If in our program we always acquire resources in a particular order and release resources in the reverse order, then we can prevent the deadlock.

So a thread waiting for same resource can not get into deadlock while the other thread is trying to get it and holding the resource required by first thread. If both of them release the resources in right order, one of them can acquire it to finish the work.

**411. How can you determine if JVM is 32-bit or 64-bit from Java Program?**

We can find JVM bit size 32 bit or 64 bit by running java command from the command prompt.

Or we can get it from Java program.

Sun has a Java System property to determine the bit size of the
JVM: 32 or 64:

sun.arch.data.model=32 // 32 bit JVM
sun.arch.data.model=64 // 64 bit JVM

We can use System.getProperty("sun.arch.data.model") to determine if it is 32/64 bit from Java program.

**412. What is the right data type to represent Money (like Dollar/Pound) in Java?**

To represent money you need decimal points in the numbers like $1.99.

BigDecimal class provides good methods to represent Money. Using BigDecimal, we can do the calculation with decimal points and correct rounding. But using BigDecimal is a little bit high on memory usage.

We can also use double with predefined precision. But calculation on double can give erroneous results.

**413. How can you do multiple inheritances in Java?**

This is a question to trick people coming from C++ and Scala background to Java. There are many Object Oriented languages that support multiple inheritances. But Java is not one of them.

Answer of this question can be that, Java does support multiple inheritances of by allowing an interface to extend other interfaces. You can implement more than one interface. But you cannot extend multiple classes. So Java doesn't support multiple inheritances of implementation.

But in Java 8, the default method breaks the rule of multiple inheritances behavior.

**414. Is ++ operation thread-safe in Java?**

No, ++ operator is not a thread safe operation. It involves multiple instructions like- reading a value, incrementing it and storing it back into memory. These instructions can overlap between multiple threads. So it can cause issues in multi-threading.

**415. How can you access a non-static variable from the static context?**

We cannot access a non-static variable from the static context in Java. If you write a code like that, then you will get compile time error. It is one of the most common problems for beginner Java programmers, when they try to access instance variable inside the main method in a class.

Since main method is static in Java, and instance variables are non-static, we cannot access instance variable inside main. The solution is to create an instance of the object and then access the instance variables.

**416. Let say there is a method that throws NullPointerException in the superclass. Can we override it with a method that throws RuntimeException?**

This question is checking your understanding of the concepts of method overloading and overriding in Java.

We can throw superclass of RuntimeException in an overridden method, but we cannot do the same if it is a checked Exception.

**417. How can you mark an array volatile in Java?**

If you know multi-threading well then you can easily answer it.

We can mark an array volatile in Java. But it makes only the reference to array volatile, not the whole array.

If one thread changes the reference variable to point to another array, then it will provide a volatile guarantee. But if multiple threads are changing individual array elements, they won't be having same reference due to the reference itself being volatile.

**418. What is a thread local variable in Java?**

Thread-local variable is a variable restricted to a specific thread. It is like thread's own copy of variable that is not shared among multiple threads.
Java provides ThreadLocal class to support thread-local variables. To achieve thread-safety, you can use it. To avoid any memory leak, it is always good to remove a thread-local variable, once its work is done.

**419. What is the difference between sleep() and wait() methods in Java?**

In Java, we use these methods to pause currently running thread.
There is a simple difference between these.

sleep() is actually meant for short pause because it doesn't release lock.

wait() is meant for conditional wait and it can release a lock that can be acquired by another thread to change the condition on which it is waiting.

**420. Can you create an Immutable object that contains a mutable object?**

In Java, it is possible to create an Immutable object that contains a mutable object.

We should not share the reference of the mutable object, since it is inside an immutable object. Instead, we can return a copy of it to other methods.

**421. How can you convert an Array of bytes to String?**

You can convert an Array of bytes to String object by using the String constructor that accepts byte[]. We need to make sure that right character encoding is used. Else we may get different results after conversion.

**422. What is difference between CyclicBarrier and CountDownLatch class?**

CyclicBarrier and CountDownLatch classes were introduced from Java 5.

We can reuse CyclicBarrier even if it is broken, but we cannot reuse CountDownLatch in Java.

**423. What is the difference between StringBuffer and StringBuilder?**

StringBuilder was introduced in Java 5. The main difference between both of them is that StringBuffer methods e.g. length(), capacity(), append() are synchronized. But corresponding methods in StringBuilder are not synchronized.

Due to this difference, concatenation of String using StringBuilder is faster than StringBuffer. Now it is considered bad practice to use StringBuffer, because, in most of the scenarios, we perform string concatenation in the same thread.

**424. Which class contains clone method? Cloneable or Object class?**

It is a very basic trick question. clone() method is defined in Object class. Cloneable is a marker interface that doesn't contain any method.

**425. How will you take thread dump in Java?**

There are platform specific commands to take thread dump in Java.

In Linux/Unix, just use kill -3 PID, where PID is the process id of Java process. It will give the thread dump of Java process.

In Windows, press Ctrl + Break. This will instruct JVM to print thread dump in standard out or err. It can also go to console or log file depending upon your application configuration.

**426. Can you cast an int variable into a byte variable? What happens if the value of int is larger than byte?**

An int is 32 bit in Java. But a byte is just 8 bit in Java. We can cast an int to byte. But we will lose higher 24 bits of int while casting. Because a byte can hold only first 8 bits of int. Remaining 24 bits (32-8 = 24) will be lost.

**427. In Java, can we store a double value in a long variable without explicit casting?**

No, we cannot store a double value into a long variable without casting it to long. The range of double is more than that of long. So we need to type cast.

To answer this question, just remember which one is bigger between double and long in Java.

**428. What will this return 5*0.1 == 0.5? true or false?**

The answer is false because floating point numbers can not be represented exactly in Java, so 5*0.1 is not same as 0.5.

**429. Out of an int and Integer, which one takes more memory?**

An Integer object takes more memory than an int in Java. An Integer is an object and it stores meta-data overhead about the object. An int is a primitive type so its takes less memory and there is no meta-data overhead.

**430. Can we use String in the switch case statement in Java?**

Yes. From Java 7 onwards, String can be used in switch case statement. This gives convenience to programmer. But internally hash code of String is used for the switch statement.

**431. Can we use multiple main methods in same class?**

Yes. You can have multiple methods with name main in the same class. But there should be only one main method with the signature public static void main(String[] args). JVM looks for main with this signature only. Other methods with name main in same class are just ignored.

**432. When creating an abstract class, is it a good idea to call abstract methods inside its constructor?**

No, we should avoid calling abstract methods in the constructor of an abstract class. Because, it can restrict how these abstract methods can be implemented by child classes.

Many IDE give "Overridable method call in constructor" warning for such implementation.

This is a problem of object initialization order. The superclass constructor will run before the child class constructor. It means child class is not yet initialized. But due to presence of overridden method in superclass, the overridden method of subclass is called when the subclass is not fully initialized.

**433. How can you do constructor chaining in Java?**

When we call one constructor from another constructor of the same class, then it is known as constructor chaining in Java. When you have multiple overloaded constructors in a class, you can do constructor chaining.

**434. How can we find the memory usage of JVM from Java code?**

We can use memory management related methods provided in java.lang.Runtime class to get the free memory, total memory and maximum heap memory in Java.

By using these methods, you can find out how much of the heap is used and how much heap space still remains.

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

**435. What is the difference between x == y and x.equals(y) expressions in Java?**

The x == y expression does object reference matching if both a and b are an object and only returns true if both are pointing to the same object in the heap space.

The x.equals(y) expression is used for logical mapping and it is expected from an object to override this method to provide logical equality.

Eg. A Book object may be logically equal to another copy of same Book, but it is a different object which will be false while doing x == y.

**436. How can you guarantee that the garbage collection takes place?**

No. We cannot guarantee the garbage collection in Java. Java documentation explicitly says that GarbageCollection is not guaranteed.

You can call System.gc() to request garbage collection, however, that's what it is - a request. It is upto GC's discretion to run.

**437. What is the relation between x.hashCode() method and x.equals(y) method of Object class?**

x.hashCode() method returns an int hash value corresponding to an object instance.

It is used in hashCode based collection classes like Hashtable, HashMap, LinkedHashMap etc.

hashCode() method is also related to equals() method.

As per Java specification, two objects which are equal to each other using equals() method must have same hash code.

Therefore, two objects with same hashCode may or may not be equal to each other. But two equal objects should have same hash code.

**438. What is a compile time constant in Java?**

A compile time constant is public static final variable. The public modifier is optional here. At compile time, they are replaced with actual values because compiler knows their value up-front and it also knows that it cannot be changed during run-time. So they are constants.

**439. Explain the difference between fail-fast and fail-safe iterators?**

The main difference between fail-fast and fail-safe iterators is whether or not the collection can be modified while it is being iterated.

Fail-safe iterators allow modification of collection in an iteration task. But fail-fast iterators do not allow any modification to collection during iteration.

During iteration, fail-fast iterators fail as soon as they realize that the collection has been modified. Modification can be addition, removal or update of a member. And it will throw a ConcurrentModificationException.
Eg. ArrayList, HashSet, and HashMap are fail-fast.

Fail-safe iterators operate on a copy of the collection. Therefore they do not throw an exception if the collection is modified during iteration.
Eg. ConcurrentHashMap, CopyOnWriteArrayList are fail-safe.

**440. You have a character array and a String. Which one is more secure to store sensitive data (like password, date of birth, etc.)?**

Short answer is, it is safe to store sensitive information in character array.

In Java, String is immutable and it is stored in the String pool. Once a String is created, it stays in the pool in memory until it is garbage collected. You have no control on garbage collection. Therefore, anyone having access to a memory dump can potentially extract the sensitive data and use it.

Whereas, if you use a mutable object like a character array, to store the value, you can set it to blank once you are done with it. Once it is made blank it cannot be used by anyone else.

**441. Why do you use volatile keyword in Java?**

The volatile keyword guarantees global ordering on reads and writes to a variable. This implies that every thread accessing a volatile field will read the variable's current value instead of using a cached value.

By marking the variable volatile, the value of a variable is never cached thread-locally. All reads and writes will go straight to main memory of Java.

**442. What is the difference between poll() and remove() methods of Queue in Java?**

It is a basic question to know the understanding of Queue data structure. Both poll() and remove() methods remove and return the head of the Queue.

When Queue is empty, poll() method fails and it returns null, but remove() method fails and throws Exception.

**443. Can you catch an exception thrown by another thread in Java?**

Yes, it can be done by using Thread.UncaughtExceptionHandler.

Java Documentation says "When a thread is about to terminate due to an uncaught exception the Java Virtual Machine will query the thread for its UncaughtExceptionHandler usingThread.getUncaughtExceptionH will invoke the handler's uncaughtException method, passing the thread and the exception as arguments."

**444. How do you decide which type of Inner Class – Static or Non-Static to use in Java?**

An inner class has full access to the fields and methods of the enclosing class. This is convenient for event handlers, but comes at a cost. Every instance of an inner class retains and requires a reference to its enclosing class.

Due to this cost, there are many situations where static nested classes are preferred over inner classes. When instances of the nested class outlive instances of the enclosing class, the nested class should be static to prevent memory leaks.

At times, due to their "hidden" reference to enclosing class, Inner classes are harder to construct via reflection.

**445. What are the different types of Classloaders in Java?**

Java Classloader is the part of the Java Runtime Environment (JRE) that loads classes on demand into Java Virtual Machine (JVM).

When the JVM is started, three types of class loaders are used:

1. Bootstrap Classloader: It loads core java API file rt.jar classes from folder.
2. Extension Classloader: It loads jar files from lib/ext folder.
3. System/Application Classloader: It loads jar files from path specified in the CLASSPATH environment variable.

Classes may be loaded from the local file system, a remote file system, or even the web.

**446. What are the situations in which you choose HashSet or TreeSet?**

HashSet is better than TressSet in almost every way. It gives O(1) for add(), remove() and contains() operations. Whereas, TressSet gives O(log(N)) for these operations.

Still, TreeSet is useful when you wish to maintain order over the inserted elements or query for a range of elements within the set.

We should use TreeSet when we want to maintain order. Or when there are enough read operations to offset the increased cost of write operations.

**447. What is the use of method references in Java?**

Java 8 has introduced Method references. It allows constructors and methods to be used as lambdas.

The main uses of Method reference are to improve code organization, clarity and terseness.

**448. Do you think Java Enums are more powerful than integer constants?**

Yes. Java Enums provide many features that integer constants cannot. Enums can be considered as final classes with a fixed number of instances. Enums can implement interfaces but cannot extend another class.

While implementing the strategy pattern, we can use this feature of Enums. Especially, when the number of strategies is fixed.

You can also attach meta-data to enum values in Java. Also enum values are typesafe, where as integer constants are not.

You can also define custom behavior in enum values.

**449. Why do we use static initializers in Java?**

In Java, a static initializer can run code during the initial loading of a class and it guarantees that this code will only run once. Also the static code will finish running before a class can be accessed in any way.

Initializing static members from constructors is more work. You have to make sure that every constructor does this. You need to maintain a flag to mark the static work when it is done. You may have to think about synchronization or races conditions for work in static block not initialized from static context.

**450. Your client is complaining that your code is throwing NoClassDefFoundError or NoSuchMethodError, even though you are able to compile your code without error and method exists in your code. What could be the reason behind this?**

Sometimes we upgrade our libraries even with same method name. But we forget to let the client know about the new version. Due this different in version, we get NoClassDefFoundError or NoSuchMethodError at runtime when one library was not compatible with such an upgrade.

Java build tools and IDEs can also produce dependency reports that tell you which libraries depend on that JAR. Mostly, identifying and upgrading the library that depends on the older JAR resolve the issue.

**451. How can you check if a String is a number by using regular expression?**

Regex is a powerful tool for matching patterns and searching patterns.

A numeric String can only contain digits i.e. 0 to 9. It can also contain + and - sign at start of the String. We can create a regular expression for these two rules. One simple example is as follows:

```
Pattern pattern = Pattern.compile(".*\\D.*");
```

**452. What is the difference between the expressions String s = "Temporary" and String s = new String("Temporary ")? Which one is better and more efficient?**

In general, String s = " Temporary " is more efficient to use than String s = new String("Temporary ").

In case of String s = " Temporary ", a String with the value “Temporary” is created in String pool. If another String with the same value is created (e.g., String s2 = " Temporary "), it will reference the same object in the String pool.

But, when you use String s = new String("Temporary "), Java creates a String with the value “Temporary” in the String pool. Also, that String object is then passed to the constructor of the String Object i.e. new String("Temporary "). And this call creates another String object (not in the String pool) with that value.

Therefore, each such call creates an additional String object. E.g. String s2 = new String("Temporary ") creates an extra String object, rather than just reusing the same String object from the String pool.

So String s = “Temporary” is always an efficient way.

**453. In Java, can two equal objects have the different hash code?**

No. It is not possible for two equal objects to have different hashcode. But two objects with same hashcode may or may not be equal.

**454. How can we print an Array in Java?**

We can print an array by using methods of Arrays class. We can either use Arrays.toString() method or we can use Arrays.deepToString() method.
Since array doesn't implement toString() method by itself, just passing an array to System.out.println() will not print its contents. But we can use Arrays.toString() to print each element of an array.

**455. Is it ok to use random numbers in the implementation of hashcode() method in Java?**

No. The hashcode of an object should be always same. If you use random number in hashcode() method, then you may get a different value of hashcode for same object. This will break the hashcode contract.

**456. Between two types of dependency injections, constructor injection and setter dependency injection, which one is better?**

Constructor injection guarantees that a class will be initialized with all its dependencies during creation. But setter injection provides flexibility to set an optional dependency.

If we are using an XML file to describe dependencies, the setter injection is more readable.

In general, it is a good practice to use constructor injection for mandatory dependencies and use setter injection for optional dependencies.

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

In Java, Document Object Model (DOM) parser loads the whole XML into memory and creates a tree based on DOM model. This helps it in quickly locating the nodes, and making a change in the structure of XML.

On the other hand, Simple API for XML (SAX) parser is an event based parser. It doesn't load the whole XML into memory. Due to this reason DOM is faster than SAX but require more memory and is not suitable to parse large XML files.

**458. Between Enumeration and Iterator, which one has better performance in Java?**

Enumeration interface is a read-only interface. It has better performance than Iterator. It is almost twice as fast as compared to an Iterator. It also uses very less memory. Also Enumeration does not have remove() method.

On the other hand, Iterator interface is safer than Enumeration, since it can check whether a collection is modified or not during iteration. If a collection is altered while an Iterator is iterating, then it throws ConcurrentModificationException.

**459. What is the difference between pass by reference and pass by value?**

Whenever an object is passed by value, it means that a copy of the object is passed. Even if changes are made to that object, it doesn’t affect the original value.

Whenever an object is passed by reference, it means that the actual object is not passed, rather a reference of the object is passed. Therefore, any changes made by an external method, are also reflected in the actual object and its reference.

**460. What are the different ways to sort a collection in Java?**

The most popular way to sort a collection in Java is by calling Collections.sort() method. You can provide your custom Comparator to sort() method for sorting the data in your custom way.

The other way is to use a Sorted collection like TreeSet or TreeMap that stores the information in a sorted order and then you can convert it to a List.

**461. Why Collection interface doesn't extend Cloneable and Serializable interfaces?**

Collection interface just 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 may allow duplicate keys, while other collections may not.

A lot of collection implementations have clone method. But many do not. It is not worthwhile to include it in all, since Collection is an abstract representation. What matters is the concrete implementation.

Cloning and serialization come into picture while doing concrete implementation. Therefore, the concrete implementations of collections should decide how they can be cloned or serialized.

**462. What is the difference between a process and a thread in Java?**

A process is simply an execution of a program.

A Thread is a single execution sequence within a process.

A process may contain multiple threads. A Thread is also called as a lightweight process.

**463. What are the benefits of using an unordered array over an ordered array?**

In an ordered array the search time has time complexity of O(log n).

Whereas, in an unordered array, search time complexity is O (n).

In an ordered array, the insert operation has a time complexity of O(n). Whereas, the insertion operation for an unordered array takes constant time of O(1).

Therefore, when we have more writes than reads, it is preferable to use an unordered array.

**464. Between HashSet and TreeSet collections in Java, which one is better?**

A HashSet is Implemented using a HashTable. Therefore, its elements are stored in a random order. The add(), remove(), and contains() methods of a HashSet have constant time complexity O(1).

A TreeSet is implemented using a tree data structure. The elements in a TreeSet are sorted in a natural order. Therefore, add(), remove(), and contains() methods have time complexity of O(logn).

So from performance perspective, HashSet has better performance than TreeSet. But if you want to store elements in a natural sorting order, then TreeSet is a better collection.

**465. When does JVM call the finalize() method?**

JVM instructs the Garbage Collector to call the finalize method, just before releasing an object from the memory. A programmer can implement finalize() method to explicitly release the resources held by the object. This will help in better memory management and avoid any memory leaks.

**466. When would you use Serial Garabage collector or Throughput Garbage collector in Java?**

The Serial Garbage collector is used for small applications that require heap memory upto 100 MB.

The Throughput Garbage collector is used in medium to large size Java applications.

**467. In Java, if you set an object reference to null, will the Garbage Collector immediately free the memory held by that object?**

No. JVM decides to run the Garbage Collector whenever it is low on memory. When Garbage Collector runs, it looks for objects that are available for garbage collection and then frees the memory associated with this object.

So just setting an Object reference null makes it eligible for Garbage Collection, but it does not immediately free the memory.

**468. How can you make an Object eligible for Garbage collection in Java?**

To make an Object eligible for Garbage collection, just make sure that it is unreachable to the program in which it is currently defined / created / used. You can set the object reference to null and make sure no other object refers it. Once the object cannot be reached, Garbage Collection can clean it during the next run.

**469. When do you use Exception or Error in Java? What is the difference between these two?**

Throwable class is the superclass of Exception and Error classes in Java.

When you want to catch the exceptional conditions that your program can create or encounter, then use the Exception class or subclass of Exception.

When you come across situations that are unexpected then use Error class in Java. Also recovering from Error is not possible in most of cases. So it is better to terminate the program.

**470. What is the advantage of PreparedStatement over Statement class in Java?**

PreparedStatements are precompiled statements for database queries. Due to this their performance is much better. Also, we can reuse PreparedStatement objects with different input values to the same query.

Where as, Statement class does not provide these features.

**471. In Java, what is the difference between throw and throws keywords?**

When we want to raise an exception in our code, we use the throw keyword with the name of the exception to be raised.

Where as, throws keyword is used in method declaration. Throws keyword tells us the Exception that can be thrown by this method. Any caller of this method should be prepared to expect this Exception.

Another minor difference is that throw is used only with one exception, but throws can be used with comma-separated list of multiple exceptions.

**472. What happens to the Exception object after the exception handling is done?**

Once the exception handling is complete, the Exception object is not reachable. Then it is garbage collected in the next run of Garbage Collector.

**473. How do you find which client machine is sending request to your servlet in Java?**

We can use the ServletRequest class to find the IP address or host name of the client machine.

There are methods getRemoteAddr() to get the IP address of the client machine and getRemoteHost() to get the host name of the client machine.

**474. What is the difference between a Cookie and a Session object in Java?**

Both Cookie and Session are used during communication between Client and Server. The Client can disable a Cookie. Due to which the Web server cannot send a cookie. But a client cannot disable a session. So a Session always works irrespective of any setting at the client side.

Also a Session can store any Java object. But the Cookie can only store small information in a String object.

**475. Which protocol does Browser and Servlet use to communicate with each other?**

HTTP protocol. The Browser and Servlet communicate with each other by using the HTTP protocol.

**476. What is HTTP Tunneling?**

There are many network communication protocols on the Internet. But HTTP is the most popular among them. HTTP Tunneling is a technique in which HTTP or HTTPS protocol encapsulated the communication done by any other type of protocol. The masking of other protocol requests as HTTP requests is known as HTTP Tunneling.

**477. Why do we use JSP instead of Servlet in Java?**

Since JSP pages are dynamically compiled into servlets, the programmers can easily make updates to the presentation layer code.

For better performance, JSP pages can be pre-compiled.

Also JSP pages provide flexibility to combine static templates like HTML or XML snippets.

In addition, programmers can make logic changes at the class level, without editing the JSP pages that use the class logic.

**478. Is empty '.java' file name a valid source file name in Java?**

Yes. You can create a class and store it in a file with name .java. You can try it yourself, by creating, compiling and running such a file. It will run correctly.

**479. How do you implement Servlet Chaining in Java?**

To implement, Servlet Chaining, there has to be more than one servlet. The output of one servlet has to be sent to a second servlet. The output of the second servlet can be sent to a third servlet, and so on. In this way, a chain of servlets is formed to complete a task.

The last servlet in the chain will be responsible for sending final response to client.

**480. Can you instantiate this class?**

```
public class A
{
   A a = new A();
}
```

No, this class cannot be instantiated, since it will result in recursively calling its constructor.

**481. Why Java does not support operator overloading?**

Java supports Method overloading but does not support operator overloading. It would make the design more complex by adding operator loading. Also it will make more complex compiler.

One more reason is that, it will reduce the performance of JVM by operator overloading, since JCM has to do extra work to find the real meaning of overloaded operators at run time.

**482. Why String class is Immutable or Final in Java?**

Since String objects are cached in a String pool, it makes sense to make the String immutable. The cached String literals are shared between multiple clients. And there is a possibility that one client's action may affect another client's access to String pool.

String is also used as a parameter in many Java classes. Eg. You can pass hostname, port number as String while opening a network connection. If any one can modify your copy of the String, it can change the hostname. Due to this reason, it makes sense to make String final as soon as it is created.

**483. What is the difference between sendRedirect and forward methods?**

When you use sendRedirect method, it creates a new request. When you use the forward method, it just forwards a request to a new target.

In case of sendRedirect, the previous request scope objects are not available, because it creates a new request.

In case of forward method, the previous request scope objects are available after forwarding.

Also the sendRedirect method is considered slower than the forward method.

**484. How do you fix your Serializable class, if it contains a member that is not serializable?**

If you want to make a class Serializable, but find that this class contains members that are not Serializable, then you have to mark those members as transient. This will ensure that this member is not persisted to a stream of bytes during Serialization.

Therefore, Transient keyword of Java comes to help in this scenario.

**485. What is the use of run time polymorphism in Java?**

During the run time the behavior of an Object can change based on its run time state. Due to this run time polymorphism is introduced in Java. If you override a method in a child class, then you are providing run time polymorphism. Nothing will happen at the compile time. But at the run time, JVM decides which method will be called based on the class of the Object.

**486. What are the rules of method overloading and method overriding in Java?**

When we want to overload a method, we need to make sure that the method name remains same. But method signature can vary in the number or datatype of arguments or in the order of arguments.

When we want to override a method, we ensure that the method is not throwing checked exceptions that are new or higher than those declared by the overridden method. Also we make sure that the method name, arguments and return type remain the same.

Also we cannot override Static and Final methods in Java.

**487. What is the difference between a class and an object in Java?**

A Class is a template or a blue print of an Object to be created. An Object is an instance of a Class. A Class defines the methods and member variables. But an Object populates the values of the member variables.

Therefore a class is a blueprint that you use to create objects. An object is an instance of a class – it is a concrete 'thing' that you made using a specific class.

Most of the OOPS concepts are valid only when an Object is created.

**488. Can we create an abstract class that extends another abstract class?**

Yes. An abstract class can extend another abstract class. It does not need to define the methods of parent abstract class. Only the last non-abstract class has to define the abstract methods of a parent abstract class.

**489. Why do you use Upcasting or Downcasting in Java ?**

When we want to cast a Sub class to Super class, we use Upcasting.
It is also known as widening. Upcasting is always allowed in Java.

When we want to cast a Super class to Sub class, we use Downcasting. It is also known as narrowing.

At times, Downcasting can throw the ClassCastException if it fails the type check.

**490. What is the reason to organize classes and interfaces in a package in Java?**

As the name suggests, a package contains a collection of classes. It helps in setting the category of a file. Like- whether it is a Data Access Object (DAO) or an API.

It helps in preventing the collision of Name space.

Also we can introduce access restriction by using package and the right modifiers on a class and its methods.

**491. What is information hiding in Java?**

Information hiding is OOPS concept. In Java you can use encapsulation to do Information hiding. An object can use the access modifiers like-public, private, protected to hide its internal details from another object. This helps in decoupling the internal logic of an object from outside world.

By using Information hiding, an object can change its internal implementation without impacting the outside calling client's code.

**492. Why does Java provide default constructor?**

In Java all the interaction takes place between Object instances. To create an Object instance, JVM needs a constructor. Java does not enforce the rule on a programmer to define a default constructor for every class.

Whenever an object has to be created and programmer has not provided a constructor, Java uses default constructor to create the object. Default constructor also initializes member variables with their default values.

**493. What is the difference between super and this keywords in Java?**

We use super keyword to access the methods of the super class from child class.

We use this keyword to access methods of the same class.

**494. What is the advantage of using Unicode characters in Java?**

Unicode characters have much larger number of characters in the specification.

They also contain Asian and non-western European characters.

Most of the modern technologies, websites and browsers support these Unicode characters.

**495. Can you override an overloaded method in Java?**

Yes. Java allows to override an overloaded method, if that method is not a static or final method.

**496. How can we change the heap size of a JVM?**

Java provides the command line parameters to set the heap size for JVM.
You can specify the values in –Xms and –Xmx parameters. These parameters stand for initial and maximum heap size of JVM.

**497. Why should you define a default constructor in Java?**

In general, Java provides a default constructor with each class. But there are certain cases when we want to define our own version of default constructor.

When we want to construct an object with default values, we create our default constructor.

At times, we can mark the default constructor private. So that any other class cannot create an instance of our class. This technique is generally used in Singleton design pattern.

**498. How will you make an Object Immutable in Java?**

To make an object immutable follow these two rules. One, do not use any setter methods that can change the fields of your class. Two, make the fields final. By following these rules, the member variables cannot be changed after initialization. This will ensure that member variables of an Object do not change. And thus the Object will be considered Immutable.

**499. How can you prevent SQL Injection in Java Code?**

In Java, you can use PreparedStatement to prevent SQL injection. In a PreparedStatement you can pass the precompiled SQL queries with pre-defined parameters. This helps in checking the type of parameters to SQL queries. So it protects your code from SQL injection attacks.

**500. Which two methods should be always implemented by HashMap key Object?**

Any object that we want to use as key for HashMap or in any other hash based collection data structure e.g. Hashtable, or ConcurrentHashMap must implement equals() and hashCode() method.

**501. Why an Object used as Key in HashMap should be Immutable?**

The Key object should be immutable so that hashCode() method always return the same value for that object.

The Hashcode returned by hashCode() method depends on values of member variables of an object. If an object is mutable, then the member variables can change. Once the member variables change, the Hashcode changes. If the same object returns different hash code at different times, then it is not reliable to be used in the HashMap.

Let say, when you insert the object, the Hashcode is X, the HashMap will store it in bucket X. But when you search for it the Hashcode is Y, then HashMap will look for the object in bucket Y. So you are not getting what you stored.

To solve this, a key object should be immutable.

Although, the compiler does not enforce this rule, a good programmer always remembers this rule.

**502. How can we share an object between multiple threads?**

There are many ways to share same object between multiple threads. You can use a BlockingQueue to pass an object from one thread to another thread.

You can also use Exchanger class for this purpose. An Exchanger is a bidirectional form of a SynchronousQueue in Java. You can use it to swap the objects as well.

**503. How can you determine if your program has a deadlock?**

If we suspect that our application is stuck due to a Deadlock, then we just take a thread dump by using the command specific to environment in which your application is running. Eg. In Linux you can use command kill -3.

In case of deadlock, you will see in thread dump the current status and stack trace of threads in the JVM, and one or more of them will be stuck with message deadlock.

Also you can do this programmatically by using the ThreadMXBean class that ships with the JDK.

If you don't need programmatic detection you can do this via JConsole. On the thread tab there is a "detect deadlock" button.