Java Interview questions

Website :
Category:
Statut: VALIDATED

QUESTION 1

Question : How can you convert Map to List?

Response:
We know that Map contains key-value pairs, whereas a list contains only objects.

Since Entry class contains both key-value pair,
Entry class will helps us to convert from Map (HashMap) to List (ArrayList).
By using Map.entrySet() you will get Set object, which intern you can use it to convert to list object.

```
public static void main(String a[]){
        Map<String, String> wordMap = new HashMap<String, String>();
        Set<Entry<String, String>> set = wordMap.entrySet();
        List<Entry<String, String>> list = new ArrayList<Entry<String, String>>(set);
}
```

QUESTION 2

Question : What is strictfp keyword?

Response:

By using strictfp keyword,
we can ensure that floating point operations take place precisely.

QUESTION 3

Question : What is System.out in Java?

Response:
Here out is an instance of PrintStream.
It is a static member variable in System class.
This is called standard output stream, connected to console.

QUESTION 4

Question : What is difference between ServletOuptputStream and PrintWriter?

Response:
ServletOutputStream: ServletResponse.getOutputStream() returns a ServletOutputStream suitable for writing binary data in the response.
The servletcontainer does not encode the binary data, it sends the raw data as it is.

PrintWriter: ServletResponse.getWriter() returns PrintWriter object which sends character text to the client.

The PrintWriter uses the characterencoding returned by getCharacterEncoding().
If the response'scharacter encoding has not been specified then it does default character encoding.

QUESTION 5

Question : What is difference between the Value Object and JDO?

Response:
Java Data Objects (JDO) is really a technology of persistence used to keep objects of Java in databases which provides the advantage to its developers by manipulating all details at applications level and helps developers to concentrate on coding that is not database specific.

But, the Value Objects represents an abstracted design blueprint, which provides a generic holder of data known as Data transfer Object which can hold data so that it can be transferred between the customer and databases. JDO provides the advantage of persisting data, while the Value Object is in charge of keeping data provisionally during the period of data transfer only. In other words, the Value Object does not provide persistence.

QUESTION 6

Question : What is the difference between Servlet and Filter?

Response:
A filter is an object that can transform the header and content (or both) of a request or response. Filters differ from web components in that filters usually do not themselves create a response. Instead, a filter provides functionality that can be “attached” to any kind of web resource. Consequently, a filter should not have any dependencies on a web resource for which it is acting as a filter; this way it can be composed with more than one type of web resource.

The main tasks that a filter can perform are as follows:

1) Query the request and act accordingly.
2) Block the request-and-response pair from passing any further.
3) Modify the request headers and data. You do this by providing a customized version of the request.
4) Modify the response headers and data. You do this by providing a customized version of the response.
5) Interact with external resources.

Servlet is used for performing the action which needs to be taken for particular request like user login, get the response based on user role, interacts with database for getting the data, business logic execution, etc.

QUESTION 7

Question :
Hibernate Eager vs Lazy Fetch Type

Response:
The relationships are defined through joins in database. Hibernate represents joins in the form of associations like One-to-One, One-to-Many and Many-to-One. It is required to define Fetch Type when you use any of these associations. Fetch Type decides on whether or not to load all the data belongs to associations as soon as you fetch data from parent table. Fetch type supports two types of loading: Lazy and Eager. By default, Fetch type would be Lazy.

FetchType.LAZY: It fetches the child entities lazily, that is, at the time of fetching parent entity it just fetches proxy (created by cglib or any other utility) of the child entities and when you access any property of child entity then it is actually fetched by hibernate.

FetchType.EAGER: it fetches the child entities along with parent.

Lazy initialization improves performance by avoiding unnecessary computation and reduce memory

requirements.

Eager initialization takes more memory consumption and processing speed is slow.

Having said that, depends on the situation either one of these initialization can be used.

QUESTION 8

Question : Can Java 8 default methods override equals, hashCode and toString?

Response:
The methods declared in java.lang.Object class can not be override in Java 8 default methods. It is forbidden to define default methods in interfaces for methods in java.lang.Object.

Default interface methods can be overwritten in classes implementing the interface and the class implementation of the method has a higher precedence than the interface implementation, even if the method is implemented in a superclass. Since all classes inherit from java.lang.Object, the methods in java.lang.Object would have precedence over the default method in the interface and be invoked instead.

QUESTION 9

Question : What is POJO?

Response:
POJO stands for “Plain Old Java Object” — it’s a pure data structure that has fields with getters and possibly setters, and may override some methods from Object (e.g. equals) or some other interface like Serializable, but does not have behavior of its own.

POJO is an ordinary Java object, not bound by any special restriction and not requiring any class path.

A Java Bean is a specification which requires a Java Class to be serializable, have a no-arg constructor and a getter and setter for each field

Therefore a bean is a POJO but a POJO may not be a bean, except that some tools refer to classes as beans when strictly they are not (such as in Spring) but this usually for the better since it makes the classes easier to use.

QUESTION 10

Question : What is the difference between Closure and Lambda in Java 8?

Response:
Lambdas are a language construct (anonymous functions), closures are an implementation technique to implement first-class functions (whether anonymous or not).

A lambda is just an anonymous function - a function defined with no name. In some languages, they are equivalent to named functions. In fact, the function definition is re-written as binding a lambda to a variable internally. In other languages, like Python, there are some (rather needless) distinctions between them, but they behave the same way otherwise.

A closure is a function that is evaluated in its own environment, which has one or more bound variables that can be accessed when the function is called. They come from the functional programming world, where there are a number of concepts in play. Closures are like lambda functions, but smarter in the sense that they have the ability to interact with variables from the outside environment of where the closure is defined.

QUESTION 11

Question :
Does Java 8 Lambda supports recursive call?

Response:
In general, Lambda implementations are mostly anonymous functions. In recursion, a method calls itself. Since anonymous function doesnot have a name, it cannot be called by itself. That means an anonymous Lambda can not be called by itself. But if we have a Lambda function declaration as a member variable or class variable, Java 8 supports recursion with Lambda functions. Java 8 does not support Lambda function declaration with local variable.

QUESTION 12

Question :
Difference between map and flatMap methods in Java 8

Response:
Both map and flatMap can be applied to a Stream<T> and they both return a Stream<R>. The difference is that the map operation produces one output value for each input value, whereas the flatMap operation produces an arbitrary number (zero or more) values for each input value. This is reflected in the arguments to each operation.

The map operation takes a Function, which is called for each value in the input stream and produces one result value, which is sent to the output stream.

The flatMap operation takes a function that conceptually wants to consume one value and produce an arbitrary number of values. However, in Java, it's cumbersome for a method to return an arbitrary number of values, since methods can return only zero or one value. One could imagine an API where the mapper function for flatMap takes a value and returns an array or a List of values, which are then sent to the output. Given that this is the streams library, a particularly apt way to represent an arbitrary number of return values is for the mapper function itself to return a stream! The values from the stream returned by the mapper are drained from the stream and are passed to the output stream. The "clumps" of values returned by each call to the mapper function are not distinguished at all in the output stream, thus the output is said to have been "flattened."

QUESTION 13

Question :
List Java-8 Streams terminal operations.

Response:
Java-8 Stream terminal operations produces a non-stream, result such as primitive value, a collection or no value at all. Terminal operations are typically preceded by intermediate operations which return another Stream which allows operations to be connected in a form of a query.

Here is the list of all Stream terminal operations:

toArray()
collect()
count()
reduce()
forEach()
forEachOrdered()
min()
max()
anyMatch()
allMatch()
noneMatch()
findAny()
findFirst()

QUESTION 14

Question :

List Java-8 Streams intermediate operations.

Response:
Java 8 Stream intermediate operations return another Stream which allows you to call multiple operations in a form of a query. Stream intermediate operations do not get executed until a terminal operation is invoked. All Intermediate operations are lazy, so they're not executed until a result of a processing is actually needed. Traversal of the Stream does not begin until the terminal operation of the pipeline is executed.

Here is the list of all Stream intermediate operations:

filter()
map()
flatMap()
distinct()
sorted()
peek()
limit()
skip()

QUESTION 15

Question :
How Java-8 Streams differ from collections?

Response:
First of all, please note that "Streams are not collections". java.util.stream is introduced to process elements in sequence. Streams are wrappers for collections and arrays. They wrap an existing collection to support operations expressed with lambdas, so you specify what you want to do, not how to do it. Also don't get confused with InputStream, java.util.stream does not have any relationship with InputStream or OutputStreams. java.util.stream are part of functional programming.

Streams differ from collection framework in several ways:

No storage. A stream is not a data structure that stores elements; instead, it conveys elements from a source such as a data structure, an array, a generator function, or an I/O channel, through a pipeline of computational operations.
Functional in nature. An operation on a stream produces a result, but does not modify its source. For example, filtering a Stream obtained from a collection produces a new Stream without the filtered elements, rather than removing elements from the source collection.
Laziness-seeking. Many stream operations, such as filtering, mapping, or duplicate removal, can be implemented lazily, exposing opportunities for optimization. For example, "find the first String with three consecutive vowels" need not examine all the input strings. Stream operations are divided into intermediate (Stream-producing) operations and terminal (value- or side-effect-producing) operations. Intermediate

operations are always lazy.
Possibly unbounded. While collections have a finite size, streams need not. Short-circuiting operations such as limit(n) or findFirst() can allow computations on infinite streams to complete in finite time.
Consumable. The elements of a stream are only visited once during the life of a stream. Like an Iterator, a new stream must be generated to revisit the same elements of the source.

QUESTION 16

Question : What are the various ways to obtain Streams in Java-8?

Response:
In Java-8, Streams can be obtained in a number of ways. Some examples are:

From a Collection via the stream() and parallelStream() methods.
From an array via Arrays.stream(Object[]).
From static factory methods on the stream classes, such as Stream.of(Object[]), IntStream.range(int, int) or Stream.iterate(Object, UnaryOperator).
The lines of a file can be obtained from BufferedReader.lines().
Streams of file paths can be obtained from methods in Files.
Streams of random numbers can be obtained from Random.ints().
Numerous other stream-bearing methods in the JDK, including BitSet.stream(), Pattern.splitAsStream(java.lang.CharSequence), and JarFile.stream().

QUESTION 17

Question : Difference between constructor injection and setter injection in Spring.

Response:
We need the assurance from the Inversion of control (IoC) container that, before using any bean, the injection of necessary beans must be done.

In setter injection strategy, we trust the Inversion of control (IoC) container that it will first create the bean first but will do the injection right before using the bean using the setter methods. And the injection is done according to your configuration. If you somehow misses to specify any beans to inject in the configuration, the injection will not be done for those beans and your dependent bean will not function accordingly when it will be in use!

But in constructor injection strategy, container imposes (or must impose) to provide the dependencies properly while constructing the bean. This was addressed as "container-agnostic manner", as we are required to provide dependencies while creating the bean, thus making the visibility of dependency, independent of any IoC container.

QUESTION 18

Question :
Differences between BeanFactory and the ApplicationContext in Spring framework.

Response:

Spring BeanFactory Container: This is the simplest container providing the basic support for DI (dependency injection) and is defined by the org.springframework.beans.factory.BeanFactory interface. The BeanFactory and related interfaces, such as BeanFactoryAware, InitializingBean, DisposableBean, are still present in Spring for the purpose of backward compatibility with a large number of third-party frameworks that integrate with Spring.

Spring ApplicationContext Container: This container adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by the org.springframework.context.ApplicationContext interface.

The ApplicationContext container includes all functionality of the BeanFactorycontainer, so it is generally recommended over BeanFactory. BeanFactory can still be used for lightweight applications like mobile devices or applet-based applications where data volume and speed is significant.

QUESTION 19

Question : What is Spring IoC container?

Response:
Inversion of Control (IoC) is also known as dependency injection (DI). It is a process whereby objects define their dependencies, that is, the other objects they work with, only through constructor arguments, arguments to a factory method, or properties that are set on the object instance after it is constructed or returned from a factory method. The container then injects those dependencies when it creates the bean. This process is fundamentally the inverse, hence the name Inversion of Control (IoC), of the bean itself controlling the instantiation or location of its dependencies by using direct construction of classes, or a mechanism such as the Service Locator pattern.

The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

To instantiate the application class.
To configure the object.

To assemble the dependencies between the objects.
There are two types of IoC containers. They are:

BeanFactory
ApplicationContext

QUESTION 20

Question : What are the standard Spring build-in events?

Response:
Spring core framework provides application level event firing and event listening which is based on the standard Observer design pattern. There are built-in application events available or we can create our own custom events in spring. Here is the list of standard built-in Spring events:

ContextRefreshedEvent: Event fired when an ApplicationContext gets initialized or refreshed (refreshed via context.refresh() call).

ContextStartedEvent: This event is published when the ApplicationContext is started. Event fired when context.start() method is called.

ContextStoppedEvent: This event is published when the ApplicationContext is stopped. Event fired when context.stop() method is called.

ContextClosedEvent: This event is published when the ApplicationContext is closed. Event fired when context.close() method is called. A closed context reaches its end of life; it cannot be refreshed or restarted.

RequestHandledEvent: This event can only be used in spring MVC environment. It is called just after an HTTP request is completed.

QUESTION 21

Question : What are the different types of bean scope in Spring framework?

Response:
In the spring bean configurations, bean attribute called 'scope' defines what kind of object has to created and returned. There are 5 types of bean scopes available, they are:

1) singleton: Returns a single bean instance per Spring IoC container.

2) prototype: Returns a new bean instance each time when requested.

3) request: Returns a single instance for every HTTP request call.

4) session: Returns a single instance for every HTTP session.

5) global session: global session scope is equal as session scope on portlet-based web applications.

If no bean scope is specified in bean configuration file, then it will be by default 'singleton'.

QUESTION 22

Question : What are the key components of Spring Boot framework?

Response:
Spring Boot Framework has mainly four major components.

Spring Boot Starters: The main responsibility of Spring Boot Starter is to combine a group of common or related dependencies into single dependencies. Spring Boot starters can help to reduce the number of manually added dependencies just by adding one dependency. So instead of manually specifying the dependencies just add one starter. Examples are spring-boot-starter-web, spring-boot-starter-test, spring-boot-starter-data-jpa, etc.

Spring Boot AutoConfigurator: One of the common complaint with Spring is, we need to make lot of XML based configurations. Spring Boot AutoConfigurator will simplify all these XML based configurations. It also reduces the number of annotations.

Spring Boot CLI: Spring Boot CLI(Command Line Interface) is a Spring Boot software to run and test Spring Boot applications from command prompt. When we run Spring Boot applications using CLI, then it internally uses Spring Boot Starter and Spring Boot AutoConfigurate components to resolve all dependencies and execute the application.

Spring Boot Actuator: Spring Boot Actuator is a sub-project of Spring Boot. It adds several production grade services to your application with little effort on your part. Actuators enable production-ready features to a Spring Boot application, without having to actually implement these things yourself. The Spring Boot Actuator is mainly used to get the internals of running application like health, metrics, info, dump, environment, etc. which is similar to your production environment monitoring setup

QUESTION 23

Question : What is the difference between Spring Boot and the Spring framework?

Response:

Spring framework is a Injection dependency framework at first targeting managing life-cycle of Java components (beans).
Today, Spring framework is pretty bloated with tons facilities/helpers on top of it.
But if you look at the big picture, it is still a framework that glue things together, a middle man to MVC frameworks (Struts 1,2, JSF etc), ORM frameworks (Hibernate, iBatis, JOOQ etc) and other necessary facilities (Quartz, Email, you can tell,
whatever you need, most likely, there's a Spring support).
It takes quite a lengthy tutorial to set Spring framework up and running because Spring framework nature is to provide flexibility of choices to you.

Spring boot on the other hand is built on a totally different mantra.
It's basically a suite, pre-configured, pre-sugared set of frameworks/technologies to reduce boiler plate configuration providing you the shortest way to have a Spring web application up and running with smallest line of code/configuration out-of-the-box.
As you can see from there Spring Boot page, it took less than 20 lines of code to have a simple RESTful application up and running with almost zero configuration.
It definitely has a ton of way to configure application to match your need.

QUESTION 24

Question :

How to reload Spring Boot Application without restarting server?

Response:

During our development time, we need to restart the server to reflect our changes.
But with the introduction of spring-boot-devtools with Spring Boot application,
we don't need to restart our server. Include below dependency in your pom.xml file.
By default, any entry on the classpath that points to a folder will be monitored for changes.

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-devtools</artifactId>
   <optional>true</optional>
</dependency>
```

With this dependency any changes you save, the embedded tomcat will restart.
This helps developers to improve the productivity.

QUESTION 25

Question : What is Spring Boot Initializr?

Response:
The Spring Initializr is ultimately a web application that can generate a Spring Boot project structure for you. It doesn't generate any application code, but it will give you a basic project structure and either a Maven or a Gradle build specification to build your code with. All you need to do is write the application code.

Spring Initializr can be used several ways, including:

A web-based interface.
Via Spring ToolSuite.
Using the Spring Boot CLI.

QUESTION 26

Question : Can we reuse Java-8 Streams?

Response:
A stream should be operated on (invoking an intermediate or terminal stream operation) only once. A stream implementation may throw IllegalStateException if it detects that the stream is being reused.

So the answer is no, streams are not meant to be reused.

QUESTION 27

Question : How does Maven resolve version conflicts of dependencies?

Response:
Maven works on the principle of nearest wins strategy while resolving the dependency conflicts, that means whichever version it finds nearer in the tree, it will take that version and ignore the other versions. Actually Maven is little bit lazy, so whenever it starts looking for a dependency it starts traversing the tree from the root and whichever version it found earlier, it will select that and returns from their without going further. If it goes further their might be a chance that it can find some newer version but as it returns from there and take the older version with it to resolve the dependencies.

Honestly speaking it is not a fault of maven because it wants to finish the job as soon as possible. Most importantly maven doesn't know which version your application is expecting so Maven will say to you, Hey, it is yours responsibility to let me know which version you want and if you don't tell me I will work my own way i.e. nearer the better.

QUESTION 28

Question :
What is Spring Boot Actuator?

Response:
Spring Boot Actuator is a sub-project of Spring Boot.
It adds several production grade services to your application with little effort on your part. Actuators enable production-ready features to a Spring Boot application, without having to actually implement these things yourself. The Spring Boot Actuator is mainly used to get the internals of running application like health, metrics, info, dump, environment, etc. which is similar to your production environment monitoring setup.

Here are some of the most common Spring Boot Actuator endpoints:

/health: It shows application health information.
/info: It displays arbitrary application info.
/metrics: It gives all metrics related information for the current application.
/trace: It displays trace information for last few HTTP requests.

QUESTION 29

Question :
What are the advantages and disadvantages of Spring Boot?

Response:
This answer is based on my personal opinion, it may vary from person to person:

Spring Boot Advantages

1.Simplified & version conflict free dependency management through the starter POMs.
2.We can quickly setup and run standalone, web applications and micro services at very less time.
3.You can just assemble the jar artifact which comes with an embedded Tomact, Jetty or Undertow application server and you are ready to go.
4.Spring Boot provides HTTP endpoints to access application internals like detailed metrics, application inner working, health status, etc.
5.No XML based configurations at all. Very much simplified properties. The beans are initialized, configured

and wired automatically.
6.The Spring Initializer provides a project generator to make you productive with the certain technology stack from the beginning. You can create a skeleton project with web, data access (relational and NoSQL datastores), cloud, or messaging support.

Spring Boot Disadvantages

1.Spring boot may unnecessarily increase the deployment binary size with unused dependencies.
2.If you are a control freak, I doubt Spring Boot would fit your needs.
3.Spring Boot sticks good with micro services. The Spring Boot artifacts can be deployed directly into Docker containers. In a large and monolithic based applications, I would not encourage you to use Spring Boot.

QUESTION 30

Question : What Embedded Containers Does Spring Boot Support?

Response:
Spring boot is a Java based framework that supports application services. It runs as a standalone jar with an embedded servlet container or as a WAR file inside a container.

The spring boot framework supports three different types of embedded servlet containers: Tomcat (default), Jetty and Undertow.

QUESTION 31

Question : What is @SpringBootApplication annotation in Spring boot project?

Response:
@SpringBootApplication annotation was introduced in Spring Boot 1.2.0 version. This annotation is equivalent to declaring these 3 annotations.

1) @Configuration

2) @EnableAutoConfiguration

3) @ComponentScan

The following are the parameters accepted in the @SpringBootApplication annotation:

exclude: Exclude the list of classes from the auto configuration.

excludeNames: Exclude the list of fully qualified class names from the auto configuration. This parameter added since spring boot 1.3.0.

scanBasePackageClasses: Provide the list of classes that has to be applied for the @ComponentScan.

scanBasePackages Provide the list of packages that has to be applied for the @ComponentScan. This parameter added since spring boot 1.3.0.

QUESTION 32

Question : Java-8: Interface with default methods vs Abstract class.

Response:
After introducing Default Method, it seems that interfaces and abstract classes are same. However, they are still different concept in Java 8.

Abstract class can define constructor. They are more structured and can have a state associated with them. While in contrast, default method can be implemented only in the terms of invoking other interface methods, with no reference to a particular implementation's state. Hence, both use for different purposes and choosing between two really depends on the scenario context.

Default methods in Java interface enables interface evolution. Given an existing interface, if you wish to add a method to it without breaking the binary compatibility with older versions of the interface, you have two options at hands: add a default or a static method. Indeed, any abstract method added to the interface would have to be implemented by the classes or interfaces implementing this interface.

A static method is unique to a class. A default method is unique to an instance of the class. If you add a default method to an existing interface, classes and interfaces which implement this interface do not need to implement it.

QUESTION 33

Question : What is interface default method in java 8?

Response:
Before Java 8, the interface only contains method signatures. With Java 8 new feature Default Methods or Defender Methods, you can include method body within the interface.

All you need to do is add "default" keyword infront of the implementation method with in the interface. This is the new way of declaring the method body in Java 8 for an interface, default methods comes along with implementation.

Why do we need to implement a method within the interface?

Let's say you have an interface which has multiple methods, and multiple classes are implementing this interface. One of the method implementation can be common across the class, we can make that method as a default method, so that the implementation is common for all classes.

How to work with existing interfaces?

Second scenario where you have already existing application, for a new requirements we have to add a method to the existing interface. If we add new method then we need to implement it through out the implementation classes. By using the Java 8 default method we can add a default implementation of that method which resolves the problem.

When working with multiple inheritance:

If we have two interfaces, one with default method and another with just method signature (normal way of defining method in the interfaces).

Refer Java 8 Interface Default Methods for more details.

QUESTION 34

Question :
What is stream pipelining in Java 8?

Response:
Stream pipelining is the concept of chaining operations together. This is done by splitting the operations that can happen on a stream into two categories. They are "intermediate operations" and "terminal operations". Each intermediate operation returns an instance of Stream itself when it runs, an arbitrary number of intermediate operations can, therefore, be set up to process data forming a processing pipeline. There must then be a terminal operation which returns a final value and terminates the pipeline.

QUESTION 35

Question : What is the difference between ORM, JPA and Hibernate?

Response:
ORM: Object Relational Mapping is concept/process of converting the data from Object oriented language to relational DB and vice versa. For example in java its done with the help of reflection and jdbc.

Hibernate: Its the implementation of above concept.

JPA: Its the one step above ORM. Its high level API and specification so that different ORM tools can implement so that it provides the flexibility to developer to change the implementation from one ORM to another (for example if application uses the JPA api and implementaion is hibernate. In future it can switch to IBatis if required. But on the other if application directly lock the implementation with Hibernate without JPA platform, switiching is going to be herculean task)

QUESTION 36

Question : Difference between ConcurrentHashMap and Collections.synchronizedMap(Map)?

Response:
The "scalability issues" for Hashtable are present in exactly the same way in Collections.synchronizedMap(Map) - they use very simple synchronization, which means that only one thread can access the map at the same time.

This is not much of an issue when you have simple inserts and lookups (unless you do it extremely intensively), but becomes a big problem when you need to iterate over the entire Map, which can take a long time for a large Map - while one thread does that, all others have to wait if they want to insert or lookup anything.

The ConcurrentHashMap uses very sophisticated techniques to reduce the need for synchronization and allow parallel read access by multiple threads without synchronization and, more importantly, provides an Iterator that requires no synchronization and even allows the Map to be modified during interation (though it makes no guarantees whether or not elements that were inserted during iteration will be returned).

Here are the differences:

Property	HashMap	Hashtable	ConcurrentHashMap
Null as a key	Allowed	Not Allowed	Not Allowed
Is thread-safe	No	Yes	Yes
Lock mechanism	N/A	Locks the whole map	Locks the portion
Iterator	Fail-fast	Fail-fast	Fail-safe

QUESTION 37

Question :
What is the difference between ConcurrentHashMap and Hashtable in Java?

Response:

1) Hashtable is belongs to the Collection framework; ConcurrentHashMap belongs to the Executor framework.

2) Hashtable uses single lock for whole data. ConcurrentHashMap uses multiple locks on segment level (16 by default) instead of object level i.e. whole Map.

3) ConcurrentHashMap locking is applied only for updates. In case of retrievals, it allows full concurrency, retrievals reflect the results of the most recently completed update operations. So reads can happen very fast while writes are done with a lock.

4) ConcurrentHashMap doesn't throw a ConcurrentModificationException if one thread tries to modify it while another is iterating over it and does not allow null values.

5) ConcurrentHashMap returns Iterator, which fails-safe (i.e. iterator will make a copy of the internal data structure) on concurrent modification.

6) ConcurrentHashMap uses a database shards logic (Segment[] segments), i.e. divides the data into shards(segments) than puts locks on each shard (segment) instead of putting a single lock for whole data (Map).

7) ConcurrentHashMap is more efficient for threaded applications.

QUESTION 38

Question : Is the spring singleton bean thread safe?

Response:
Spring framework does not do anything under the hood concerning the multi-threaded behavior of a singleton bean.
It is the developer's responsibility to deal with concurrency issue and thread safety of the singleton bean.

While practically, most spring beans have no mutable state, and as such are trivially thread safe. But if your bean has mutable state, so you need to ensure thread safety.
The most easy and obvious solution for this problem is to change bean scope of mutable beans from "singleton" to "prototype".

QUESTION 39

Question : Why ConcurrentHashMap is faster than Hashtable in Java?

Response:
ConcurrentHashMap is introduced in Java 1.5. ConcurrentHashMap uses multiple buckets to store data. This

avoids read locks and greatly improves performance over a HashTable. Both are thread safe, but there are obvious performance wins with ConcurrentHashMap.

When you read from a ConcurrentHashMap using get(), there are no locks, contrary to the HashTable for which all operations are simply synchronized. HashTable was released in old versions of Java whereas ConcurrentHashMap is a java 1.5 thing.

HashMap is the best thing to use in a single threaded application.

QUESTION 40

Question : What are the limitations and disadvantages of spring autowiring?

Response:
Autowiring works best when it is used consistently across a project. If autowiring is not used in general, it might be confusing to developers to use it to wire only one or two bean definitions. Consider the limitations and disadvantages of autowiring:

Explicit dependencies in property and constructor-arg settings always override autowiring. You cannot autowire so-called simple properties such as primitives, Strings, and Classes (and arrays of such simple properties). This limitation is by-design.

Autowiring is less exact than explicit wiring. Although, as noted in the above table, Spring is careful to avoid guessing in case of ambiguity that might have unexpected results, the relationships between your Spring-managed objects are no longer documented explicitly.

Wiring information may not be available to tools that may generate documentation from a Spring container.

Multiple bean definitions within the container may match the type specified by the setter method or constructor argument to be autowired. For arrays, collections, or Maps, this is not necessarily a problem. However for dependencies that expect a single value, this ambiguity is not arbitrarily resolved. If no unique bean definition is available, an exception is thrown.

QUESTION 41

Question : What are different types of spring auto-wiring modes?

Response:
In Spring framework, you can wire beans automatically with auto-wiring feature. To enable it, just define the “autowire” attribute in <bean>. In Spring, 5 Auto-wiring modes are supported.

no: Default, no auto wiring, set it manually via “ref” attribute

byName: Auto wiring by property name. If the name of a bean is same as the name of other bean property, auto wire it.

byType: Auto wiring by property data type. If data type of a bean is compatible with the data type of other bean property, auto wire it.

constructor: byType mode in constructor argument.

autodetect: If a default constructor is found, use “autowired by constructor”; Otherwise, use “autowire by type”.

QUESTION 42

Question : What is IoC or inversion of control?

Response:
Inversion of control (IoC) is the principle where the control flow of a program is inverted: instead the programmer controls the flow of a program, the external sources (framework, services, other components) take control of it. As the name implies Inversion of control means now we have inverted the control of creating the object from our own using new operator to container or framework. Now it’s the responsibility of container to create object as required.

QUESTION 43

Question :
How to make a bean as singleton in spring?

Response:
Beans defined in spring framework are singleton beans. There is an attribute in bean tag called ‘singleton’. If it is specified as true then bean becomes singleton and if it sets to false then the bean becomes a prototype bean, it is non-singleton class. By default this property is set to true. So, all the beans in spring framework are by default singleton beans.

QUESTION 44

Question : What is difference between BeanFactory and ApplicationContext in spring?

Response:
Use an ApplicationContext unless you have a good reason for not doing so.

Because the ApplicationContext includes all functionality of the BeanFactory, it is generally recommended over the BeanFactory, except for a few situations such as in an Applet where memory consumption might be critical and a few extra kilobytes might make a difference. However, for most typical enterprise applications and systems, the ApplicationContext is what you will want to use.

Spring 2.0 and later makes heavy use of the BeanPostProcessor extension point (to effect proxying and so on). If you use only a plain BeanFactory, a fair amount of support such as transactions and AOP will not take effect, at least not without some extra steps on your part. This situation could be confusing because nothing is actually wrong with the configuration.

At very high level, here are the differences:

BeanFactory

1) Bean instantiation/wiring

ApplicationContext

1) Bean instantiation/wiring
2) Automatic BeanPostProcessor registration
3) Automatic BeanFactoryPostProcessor registration
4) Convenient MessageSource access (for i18n)
5) ApplicationEvent publication

QUESTION 45

Question :
What is BeanFactory in Spring?

Response:
A BeanFactory is like a factory class that contains a collection of beans. The BeanFactory holds Bean Definitions of multiple beans within itself and then instantiates the bean whenever asked for by clients.

The BeanFactory is the actual container which instantiates, configures, and manages a number of beans. These beans typically collaborate with one another, and thus have dependencies between themselves. These dependencies are reflected in the configuration data used by the BeanFactory

BeanFactory also takes part in the life cycle of a bean, making calls to custom initialization and destruction methods.

QUESTION 46

Question : What are the different types of dependency injections in spring?

Response:
Spring supports 2 types of dependency injection, they are:

1) Constructor-based dependency injection: It is accomplished when the container invokes a class constructor with a number of arguments, each representing a dependency on other class.

2) Setter-based dependency injection: It is accomplished by the container calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

QUESTION 47

Question : What is Dependency Injection?

Response:
Any application is composed of many objects that collaborate with each other to perform some useful stuff. Traditionally each object is responsible for obtaining its own references to the dependent objects (dependencies) it collaborate with. This leads to highly coupled classes and hard-to-test code.

For example, consider a Car object. A Car depends on Wheels, Engine, Fuel, Battery, etc to run. Traditionally we define the brand of such dependent objects along with the definition of the Car object.

```
Class Car{
  private Wheel wh= new NepaliRubberWheel();
  private Battery bt= new ExcideBattery();
  //rest
}
```

Here, the Car object is responsible for creating the dependent objects.

What if we want to change the type of its dependent object - say Wheel - after the initial NepaliRubberWheel() punctures? We need to recreate the Car object with its new dependency say ChineseRubberWheel(), but only the Car manufacturer can do that.

Then what the Dependency Injection do us for ...

When using Dependency Injection, objects are given their dependencies at run time rather than compile time (car manufacturing time). So that we can now change the Wheel whenever we want. Here, the Dependency

(Wheel) can be injected into Car at run time.

Inversion of Control (IoC) is a general concept, and it can be expressed in many different ways and Dependency Injection is merely one concrete example of Inversion of Control.

This concept says that you do not create your objects but describe how they should be created. You don't directly connect your components and services together in code but describe which services are needed by which components in a configuration file. A container is then responsible for hooking it all up.

QUESTION 48

Question : How Struts control data flow?

Response:
Struts implements the MVC pattern through the use of ActionForwards and ActionMappings to keep control-flow decisions out of presentation layer.

When you deploy your struts application:

Having received the request the ActionServlet locates the correct Form Bean from the struts-config.xml. The ActionServlet invokes the FormBean Validator logic. If the Form Bean finds any error, it populates the ActionError Object

Using the Struts-Config.xml the ActionServlet locates and invokes the Action Class which invokes the model and returns an ActionForward object to the ActionServlet.

The ActionServlet uses the ActionForward Object to dispatch to the correct view component.

QUESTION 49

Question : What is the difference between application server and web server?

Response:
Web Server is designed to serve HTTP Content. Application Server can also serve HTTP Content but is not limited to just HTTP. It can be provided other protocol support such as RMI/RPC

Web Server is mostly designed to serve static content, though most Web Servers have plugins to support scripting languages like Perl, PHP, ASP, JSP etc. through which these servers can generate dynamic HTTP content.

Most of the application servers have Web Server as integral part of them, that means App Server can do whatever Web Server is capable of. Additionally Application Server have components and features to support Application level services such as Connection Pooling, Object Pooling, Transaction Support, Messaging services etc.

As web servers are well suited for static content and app servers for dynamic content, most of the production environments have web server acting as reverse proxy to app server. That means while service a page request, static contents such as images/Static html is served by web server that interprets the request. Using some kind of filtering technique (mostly extension of requested resource) web server identifies dynamic content request and transparently forwards to app server

Example of such configuration is Apache HTTP Server and BEA WebLogic Server. Apache HTTP Server is Web Server and BEA WebLogic is Application Server.

QUESTION 50

Question : What is Spring?

Response:
Spring is an open source development framework for enterprise Java. The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make J2EE development easier to use and promote good programming practice by enabling a POJO-based programming model.

Basically Spring is a framework for dependency-injection which is a pattern that allows to build very decoupled systems.

Spring is a good framework for web development. Spring MVC is one of the many parts of Spring, and is a web framework making use of the general features of Spring, like dependency injection. It is a pretty generic framework in that it is very configurable: you can use different DB layers (Hibernate, iBatis, plain JDBC), different view layers (JSP, Velocity, Freemarker...)

QUESTION 51

Question : What is ActionMapping in struts?

Response:
An ActionMapping represents the information that the controller, RequestProcessor, knows about the mapping of a particular request to an instance of a particular Action class. The ActionMapping instance used to select a particular Action is passed on to that Action, thereby providing access to any custom configuration information included with the ActionMapping object.

ActionMapping object contains all the mapping information in struts_config.xml. ActionServlet create the instance of ActionMapping and load all the struts_config.xml data to ActionMapping object.

QUESTION 52

Question : Can we have constructor in abstract class?

Response:
You can not instantiate abstract class in java. In order to use abstract class in Java, You need to extend it and provide a concrete class. Abstract class is commonly used to define base class for a type hierarchy with default implementation, which is applicable to all child classes. Now based on these details, can we have constructor in abstract class? The answer is YES, we can have. You can either explicitly provide constructor to abstract class or if you don't, compiler will add default constructor of no argument in abstract class. This is true for all classes and its also applies on abstract class.

QUESTION 53

Question : Can Enum implements any interface in Java?

Response:
Yes, Enum can implement any interface in Java. Since enum is a type, similar to any other class and interface, it can implement any interface in java. This gives lot of flexibility in some cases to use Enum to implement some other behavior.

QUESTION 54

Question : Can Enum extend any class in Java?

Response:
Enum can not extend any class in java, the reason is by default, Enum extends abstract base class java.lang.Enum. Since java does not support multiple inheritance for classes, Enum can not extend another class.

QUESTION 55

Question : What is the difference between HTTP methods GET and POST?

Response:
HTTP works as a request-response protocol between a client and server. A web browser may be the client, and an application on a computer that hosts a web site may be the server. Two commonly used HTTP methods to make a request to the server are GET and POST.

When you use GET method, the data will be sent to the server as a query parameters. These are appended to the URL as a key value pair. In the below URL, you can see how data is passed to the server as key value pair. These values will be visible at the address bar. URL character length is limited, so you can not use it if you are sending large data. GET is recommended to use for querying information from server, kind of search operations. GET requests should never be used when dealing with sensitive data.

http://java2novice.com/history?name=madhu&language=java

POST method sends data as part of HTTP message body, data sent to the server, will not be visible to the user. POST requests cannot be cached. It does not have any character length restrictions. POST is recommended to submits data to be processed to a specified resource.

QUESTION 56

Question : How Struts control data flow?

Response:
Struts implements the MVC pattern through the use of ActionForwards and ActionMappings to keep control-flow decisions out of presentation layer.

When you deploy your struts application:

Having received the request the ActionServlet locates the correct Form Bean from the struts-config.xml. The ActionServlet invokes the FormBean Validator logic. If the Form Bean finds any error, it populates the ActionError Object

Using the Struts-Config.xml the ActionServlet locates and invokes the Action Class which invokes the model and returns an ActionForward object to the ActionServlet.

The ActionServlet uses the ActionForward Object to dispatch to the correct view component.

QUESTION 57

Question : What is MVC pattern?

Response:
MVC is a design pattern called Model-View-Controller. It decouples data access logic from business logic.

Model:

The Model contains the core of the application's functionality. It encapsulates the state of the application. Sometimes the only functionality it contains is state. It knows nothing about the view or controller.

View:

The view provides the presentation of the model. It is the look and feel of the application. The view can access the model getters, but it has no knowledge of the setters. In addition, it knows nothing about the controller. The view should be notified when changes to the model occur.

Controller:

The controller reacts to the user input. It creates and sets the model and helps to identify which view should be part of response.

QUESTION 58

Question : What is ActionServlet in struts?

Response:
ActionServlet provides the "controller" in the Model-View-Controller (MVC) design pattern for web applications that is commonly known as "Model 2".
All the requests to the server goes through the controller.
Controller is responsible for handling all the requests. Struts Flow start with ActionServlet then call to process() method of RequestProcessor.

QUESTION 59

Question : What is difference between CountDownLatch and CyclicBarrier in Java?

Response:
Both CyclicBarrier and CountDownLatch are used to implement a scenario where one Thread waits for one or more Thread to complete their job before starts processing. The differences are:

1) CyclicBarrier is resulable, CountDownLatch is not.
2) Both CyclicBarrier and CountDownLatch wait for fixed number of threads.
3) CountDownLatch is advanceable but CyclicBarrier is not.

QUESTION 60

Question : What is HTTP basic authentication?

Response:
In the context of a HTTP transaction, basic access authentication is a method for an HTTP user agent to provide a user name and password when making a request.

HTTP Basic authentication implementation is the simplest technique for enforcing access controls to web resources because it doesn't require cookies, session identifier and login pages. Rather, HTTP Basic authentication uses static, standard HTTP headers which means that no handshakes have to be done in anticipation.

When the user agent wants to send the server authentication credentials it may use the Authorization header. The Authorization header is constructed as follows:

1) Username and password are combined into a string "username:password"
2) The resulting string is then encoded using Base64 encoding
3) The authorization method and a space i.e. "Basic " is then put before the encoded string.

For example, if the user agent uses 'Aladdin' as the username and 'open sesame' as the password then the header is formed as follows:

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

QUESTION 61

Question : What is functional interface in java?

Response:
In Java, a Marker interface is an interface without any methods or fields declaration, means it is an empty interface. Similarly, a Functional Interface is an interface with just one abstract method declared in it. Runnable interface is an example of a Functional Interface. It has only run() method declared in it.

Lambda expression works on functional interfaces to replace anonymous classes.

@FunctionalInterface is a new annotation added in Java 8 to indicate that an interface declaration is intended to be a functional interface as defined by the Java Language Specification. @FunctionalInterface can be used for compiler level errors when the interface you have annotated is not a valid Functional Interface.

QUESTION 62

Question : What is difference between Lambda Expression and Anonymous class?

Response:
The key difference between Anonymous class and Lambda expression is the usage of 'this' keyword.
In the anonymous classes,
'this' keyword resolves to anonymous class itself,
whereas for lambda expression 'this' keyword resolves to enclosing class where lambda expression is written.

Another difference between lambda expression and anonymous class is in the way these two are compiled.
Java compiler compiles lambda expressions and convert them into private method of the class.
It uses invokedynamic instruction that was added in Java 7 to bind this method dynamically.

QUESTION 63

Question : What is HQL (Hibernate Query Language)?

Response:
HQL stands for Hibernate Query Language. Hibernate allows the user to express queries in its own portable SQL extension and this is called as HQL.

Hibernate uses a powerful query language (HQL) that is similar in appearance to SQL. Compared with SQL, however, HQL is fully object-oriented and understands notions like inheritance, polymorphism and association.

QUESTION 64

Question : How garbage collector knows that the object is not in use and needs to be removed?

Response:
Garbage collector reclaims objects that are no longer being used, clears their memory,
and keeps the memory available for future allocations.

This is done via bookkeeping the references to the objects. Any unreferenced object is a garbage and will be collected.

QUESTION 65

Question : Can Java thread object invoke start method twice?

```
package com.java2novice.exmpcode;

public class MyExmpCode extends Thread{

	public void run(){
		System.out.println("Run");
	}

	public static void main(String a[]){
		Thread t1 = new Thread(new MyExmpCode());
		t1.start();
		t1.start();
	}
}
```

Response:
No, it throws IllegalThreadStateException

QUESTION 66

Question : Give the list of Java Object class methods.

Response:
clone() - Creates and returns a copy of this object.
equals() - Indicates whether some other object is "equal to" this one.
finalize() - Called by the garbage collector on an object when garbage collection
determines that there are no more references to the object.
getClass() - Returns the runtime class of an object.
hashCode() - Returns a hash code value for the object.
notify() - Wakes up a single thread that is waiting on this object's monitor.
notifyAll() - Wakes up all threads that are waiting on this object's monitor.
toString() - Returns a string representation of the object.
wait() - Causes current thread to wait until another thread invokes the notify() method
or the notifyAll() method for this object.

QUESTION 67

Question : Can we call servlet destory() from service()?

Response:
As you know, destory() is part of servlet life cycle methods, it is used to kill the servlet instance.
Servlet Engine is used to call destory().
In case, if you call destroy
method from service(), it just execute the code written in the destory(), but it wont kill the servlet instance. destroy() will be called before killing the servlet instance by servlet engine.

QUESTION 68

Question : Can we override static method?

Response:
We cannot override static methods.
Static methods are belogs to class, not belongs to object.

Inheritance will not be applicable for class members

QUESTION 69

Question : Can you list serialization methods?

Response:
Serialization interface does not have any methods.
It is a marker interface.
It just tells that your class can be serializable.

QUESTION 70

Question : What is the difference between super() and this()?

Response:
super() is used to call super class constructor,
whereas
this() used to call constructors in the same class,
means to call parameterized constructors.

QUESTION 71

Question : How to prevent a method from being overridden?

Response:
By specifying final keyword to the method you can avoid overriding
in a subcalss. Similarlly one can use final at class level to
prevent creating subclasses.

QUESTION 72

Question : Can we create abstract classes without any abstract methods?

Response:
Yes, we can create abstract classes without any abstract methods.

QUESTION 73

Question : How to destroy the session in servlets?

Response:
By calling invalidate() method on session object, we can destory the session.

QUESTION 74

Question : Can we have static methods in interface?

Response:
By default, all methods in an interface are declared as public, abstract.It will never be static.

this concept is changed with java 8.
Java 8 came with new feature called "default methods" with in interfaces

https://www.java2novice.com/java-8/interface-default-methods/

QUESTION 75

Question : What is transient variable?

Response:
transient variables cannot be serialized.
During serialization process,transient variable states will not be serialized.
State of the value will be always defaulted after deserialization.

QUESTION 76

Question : Incase, there is a return at the end of try block, will execute finally block?

Response:
Yes, the finally block will be executed even after writing return statement
at the end fo try block.

It returns after executing finally block.

QUESTION 77

Question : What is abstract class or abstract method?

Response:
We cannot create instance for an abstract class. We can able to create
instance for its subclass only. By specifying abstract keyword just before
class, we can make a class as abstract class.

```
public abstract class MyAbstractClass{

}
```

Abstract class may or may not contains abstract methods. Abstract method is
just method signature, it does not containes any implementation. Its subclass
must provide implementation for abstract methods. Abstract methods are looks
like as given below:

```
public abstract int getLength();
```

QUESTION 78

Question : What is default value of a boolean?

Response:
Default value of a boolean is false.

QUESTION 79

Question : When to use LinkedList or ArrayList?

Response:
Accessing elements are faster with ArrayList, because it is index based.
But accessing is difficult with LinkedList. It is slow access.
This is to access any element, you need to navigate through the elements one by one.
But insertion and deletion is much faster with LinkedList, because
if you know the node, just change the pointers before or after nodes.
Insertion and deletion is slow with ArrayList, this is because, during
these operations ArrayList need to adjust the indexes according to
deletion or insetion if you are performing on middle indexes.
Means, an ArrayList having 10 elements, if you are inserting at index 5, then
you need to shift the indexes above 5 to one more.

QUESTION 80

Question : What is daemon thread?

Response:
Daemon thread is a low priority thread. It runs intermittently
in the back ground, and takes care of the garbage collection
operation for the java runtime system. By calling setDaemon()
method is used to create a daemon thread.

QUESTION 81

Question : Does each thread in java uses separate stack?

Response:
In Java every thread maintains its own separate stack.
It is called Runtime Stack but they share the same memory.

QUESTION 82

Question : What is the difference between Enumeration and Iterator?

Response:
The functionality of Enumeration and the Iterator are same. You can get remove()
from Iterator to remove an element, while while Enumeration does not have remove()
method.
Using Enumeration you can only traverse and fetch the objects, where as using
Iterator we can also add and remove the objects.

So Iterator can be useful if you wanto manipulate the list and Enumeration is for read-only access.

QUESTION 83

Question : Find out below switch statement output.

```
public static void main(String a[]){
        int price = 6;
        switch (price) {
                case 2: System.out.println("It is: 2");
                default: System.out.println("It is: default");
                case 5: System.out.println("It is: 5");
                case 9: System.out.println("It is: 9");
        }
}
```

Response:
It is: default

It is: default
It is: 5
It is: 9

QUESTION 84

Question : Does system.exit() in try block executes code in finally block?

```
try{
        System.out.println("I am in try block");
        System.exit(1);
    } catch(Exception ex){
        ex.printStackTrace();
    } finally {
        System.out.println("I am in finally block!!!");
    }
```

Response:
It will not execute finally block. The program will be terminated
after System.exit() statement.

QUESTION 85

Question : What is fail-fast in java?

Response:
A fail-fast system is nothing but immediately report any failure that
is likely to lead to failure.
When a problem occurs, a fail-fast system fails immediately.
In Java, we can find this behavior with iterators.
Incase, you have called iterator on a collection object, and another
thread tries to modify the collection object, then concurrent modification
exception will be thrown. This is called fail-fast.

QUESTION 86

Question : What is final, finally and finalize?

Response:
final:

final is a keyword. The variable decleared as final should be initialized only once and cannot be changed. Java classes declared as final cannot be extended. Methods declared as final cannot be overridden.

finally:

finally is a block. The finally block always executes when the

try block exits. This ensures that the finally block is executed even if an unexpected exception occurs. But finally is useful for more than just exception handling - it allows the programmer to avoid having cleanup code accidentally bypassed by a return, continue, or break. Putting cleanup code in a finally block is always a good practice, even when no exceptions are anticipated.

finalize:

finalize is a method. Before an object is garbage collected, the runtime system calls its finalize() method. You can write system resources release code in finalize() method before getting garbage collected.

QUESTION 87

Question : In java, are true and false keywords?

Response:

true, false, and null might seem like keywords, but they are actually literals.

You cannot use them as identifiers in your programs.

QUESTION 88

Question : What are the different session tracking methods?

Response:

Cookies:

You can use HTTP cookies to store information. Cookies will be stored at browser side.

URL rewriting:

With this method, the information is carried through url as request parameters. In general added parameter will be sessionid, userid.

HttpSession:

Using HttpSession, we can store information at server side. Http Session provides methods to handle session related information.

Hidden form fields:

By using hidden form fields we can insert information in the webpages and these information will be sent to the server. These fields are not visible directly to the user, but can be viewed using view source option from the browsers. The hidden form fields are as given below:

<input type='hidden' name='siteName' value='java2novice'/>

QUESTION 89

Question : What is the purpose of garbage collection?

Response:
The garbage collection process is to identify the objects which are no longer referenced or needed by a program so that their resources can be reclaimed and reused.

These identified objects will be discarded

QUESTION 90

Question : What are the types of ResultSet?

Response:
The type of a ResultSet object determines the level of its functionality in two areas: the ways in which the cursor can be manipulated, and how concurrent changes made to the underlying data source are reflected by the ResultSet object. The sensitivity of a ResultSet object is determined by one of three different ResultSet types:

TYPE_FORWARD_ONLY:

The result set cannot be scrolled; its cursor moves forward only, from before the first row to after the last row. The rows contained in the result set depend on how the underlying database generates the results. That is, it contains the rows that satisfy the query at either the time the query is executed or as the rows are retrieved.

TYPE_SCROLL_INSENSITIVE:

The result can be scrolled; its cursor can move both forward and backward relative to the current position, and it can move to an absolute position. The result set is insensitive to changes made to the underlying data source

while it is open. It contains the rows that satisfy the query at either the time the query is executed or as the rows are retrieved.

TYPE_SCROLL_SENSITIVE:

The result can be scrolled; its cursor can move both forward and backward relative to the current position, and it can move to an absolute position. The result set reflects changes made to the underlying data source while the result set remains open.

The default ResultSet type is TYPE_FORWARD_ONLY.

QUESTION 91

Question : What is difference between wait and sleep methods in java?

Response:

sleep():

It is a static method on Thread class. It makes the current thread into the "Not Runnable" state for specified amount of time. During this time, the thread keeps the lock (monitors) it has acquired.

wait():

It is a method on Object class. It makes the current thread into the "Not Runnable" state. Wait is called on a object, not a thread. Before calling wait() method, the object should be synchronized, means the object should be inside synchronized block. The call to wait() releases the acquired lock.

QUESTION 92

Question : What is servlet context?

Response:

The servlet context is an interface which helps to communicate with other servlets. It contains information about the Web application and container.

It is kind of application environment.
Using the context, a servlet can obtain URL references to resources, and store attributes that other servlets in the context can use.

QUESTION 93

Question : What happens if one of the members in a class does not implement Serializable interface?

Response:
When you try to serialize an object which implements Serializableinterface,

incase if the object includes a reference of an non serializable object then NotSerializableException will be thrown.

QUESTION 94

Question : What is race condition?

Response:
A race condition is a situation in which two or more threads or processes are reading or writing some shared data,
and the final result depends on the timing of how the threads are scheduled.

Race conditions can lead to unpredictable results and subtle program bugs.
A thread can prevent this from happening by locking an object.
When an object is locked by one thread and another thread tries to call a synchronized method on the same object,
the second thread will block until the object is unlocked.

QUESTION 95

Question : How to get current time in milli seconds?

Response:
System.currentTimeMillis() returns the current time in milliseconds.

It is a static method, returns long type.

QUESTION 96

Question : What is java static import?

Response:
By using static imports, we can import the static members from a class
rather than the classes from a given package.

For example, Thread class has static sleep method,

```
import static java.lang.Thread;
public class MyStaticImportTest {
        public static void main(String[] a) {
                try{
                        sleep(100);
                } catch(Exception ex){

                }
        }
}
```

QUESTION 97

Question : What is the initial state of a thread when it is started?

Response:
When the thread is createdn and started,
initially it will be in the ready state.

QUESTION 98

Question : What is difference between StringBuffer and StringBuilder?

Response:
The only difference between StringBuffer and StringBuilder is
StringBuffer is thread-safe, that is StringBuffer is synchronized.

QUESTION 99

Question : When to use String and StringBuffer?

Response:

We know that String is immutable object.
We can not change the value of a String object once it is initiated.
If we try to change the value of the existing String object then it creates new object rather than changing the value of the existing object.

So incase, we are going to do more modifications on String, then use StringBuffer.
StringBuffer updates the existing objects value, rather creating new object.

QUESTION 100

Question : What is wrapper class in java?

Response:
Everything in java is an object, except primitives.
Primitives are int, short, long, boolean, etc.
Since they are not objects, they cannot return as objects, and collection of objects.
To support this, java provides wrapper classes to move primitives to objects.

Some of the wrapper classes are Integer, Long, Boolean, etc.

QUESTION 101

Question : Is Iterator a Class?

Response:
Iterator is an interface. It is not a class.
It is used to iterate through each and every element in a list.
Iterator is implemented Iterator design pattern.

QUESTION 102

Question : What is java classpath?

Response:
The classpath is an environment variable.
It is used to let the compiler know where the class files are available for import.

QUESTION 103

Question : Can a class in java be private?

Response:
We can not declare top level class as private.
Java allows only public and default modifier for top level classes in java.
Inner classes can be private.

QUESTION 104

Question : Is null a keyword in java?

Response:
The null value is not a keyword in java. true and false are also not
keywords in java.
They are reserved words in java language.

Java 11 utilise 54 mots réservés qui ne peuvent pas être utilisés comme identifiant.

Les mots réservés (reserved words) peuvent être regroupés en deux catégories :

51 mots clés (keywords) dont 2 ne sont pas utilisés
3 valeurs littérales (literals) : true, false et null

QUESTION 105

Question : What is the super class for Exception and Error?

Response:
The super class or base class for Exception and Error is Throwable.

QUESTION 106

Question : What is Class.forName()?

Response:

Class.forName() loads the class into the ClassLoader.

QUESTION 107

Question : Can interface be final?

Response:
No. We can not instantiate interfaces, so in order to make interfaces useful we must create subclasses.
The final keyword makes a class unable to be extended.

QUESTION 108

Question : What is the difference between exception and error?

Response:
An error is an irrecoverable condition occurring at runtime like out of memory error.
These kind of jvm errors cannot be handled at runtime.
Exceptions are because of condition failures, which can be handled easily at runtime.

QUESTION 109

Question : What is default value of a local variables?

Response:
The local variables are not initialized to any default values.
We should not use local variables without initialization.

Even the java compiler throws error.

QUESTION 110

Question : What is local class in java?

Response:

In java, local classes can be defined in a block as in a
method body or local block.

QUESTION 111

Question : Can we initialise uninitialized final variable?

Response:
Yes.
We can initialise blank final variable in constructor, ONLY in construtor.
The condition here is the final variable should be non-static.

QUESTION 112

Question : Can we declare abstract method as final?

Response:
No, we can not declare abstract method as final.
We have to proved implementation to abstract methods in subclasses.

QUESTION 113

Question : Can we have finally block without catch block?

Response:
Yes, we can have finally block without catch block.

QUESTION 114

Question : What is pass by value and pass by reference?

Response:
Pass by value: Passing a copy of the value, not the original reference.
Pass by reference: Passing the address of the object, so that you can access the original object.

QUESTION 115

Question : Can we declare main method as private?

Response:
Yes, we can declare main method as private.
It compiles without any errors, but in runtime, it says main method is not public.

QUESTION 116

Question : What is the difference between preemptive scheduling and time slicing?

Response:
Preemptive scheduling: The highest priority task executes
until it enters the waiting or dead states or a higher priority task comes into existence.

Time slicing: A task executes for a predefined slice of time and then reenters the pool of ready tasks.
The scheduler then determines which task should execute next, based on priority and other factors.

QUESTION 117

Question : Can non-static member classes (Local classes) have static members?

Response:
No, non-static member classes cannot have static members.
Because, an instance of a non-static member class or local class must be
created in the context of an instance of the enclosing class.

You can declare constants, means static final variables.

QUESTION 118

Question : What are the environment variables do we neet to set to run Java?

Response:
We need to set two environment variables those are PATH and CLASSPATH.

QUESTION 119

Question : Can you serialize static fields of a class?

Response:
Since static fields are not part of object state, they are part of class, serialization ignores the static fields.

QUESTION 120

Question : What is the difference between declaring a variable and defining a variable?

Response:
When variable declaration we just mention the type of the variable and it's name, it does not have any reference to live object.
But defining means combination of declaration and initialization.

The examples are as given below:

Declaration:
List list;
Defining:
List list = new ArrayList();

QUESTION 121

Question : Where can we use serialization?

Response:
Whenever an object has to sent over the network, those objects should be serialized.
Also if the state of an object is to be saved, objects need to be serialized.

QUESTION 122

Question : What modifiers are allowed for methods in an Interface?

Response:

Only public and abstract modifiers are allowed for methods in an interfaces.

java8 default ?

QUESTION 123

Question : What is the purpose of Runtime and System class?

Response:
The purpose of the Runtime class is to provide access to the Java runtime system.
The runtime information like memory availability, invoking the garbage collector, etc.

The purpose of the System class is to provide access to system resources.
It contains accessibility to standard input, standart output, error output streams, current time in millis, terminating the application, etc.

QUESTION 124

Question : Which one is faster? ArrayList or Vector? Why?

Response:
ArrayList is faster than Vector.
The reason is synchronization. Vector is synchronized.
As we know synchronization reduces the performance.

QUESTION 125

Question : What is the difference between static synchronized and synchronized methods?

Response:
Static synchronized methods synchronize on the class object.
If one thread is executing a static synchronized method, all other threads trying to execute any static synchronized methods will be blocked.

Non-static synchronized methods synchronize on this ie the instance of the class.
If one thread is executing a synchronized method, all other threads trying to execute any synchronized methods will be blocked.

QUESTION 126

Question : What is the order of catch blocks when catching more than one exception?

Response:
When you are handling multiple catch blocks, make sure that you are specifing exception sub classes first, then followed by exception super classes.
Otherwise we will get compile time error.

QUESTION 127

Question : What is the difference between the prefix and postfix forms of the increment(++) operator?

Response:
The prefix form first performs the increment operation and then returns the value of the increment operation. The postfix form first returns the current value of the expression and then performs the increment operation on that value.

For example:
int count=1;
System.out.println(++count);

displays 2. And

int count=1;
System.out.println(count++);

displays 1.

QUESTION 128

Question : What is hashCode?

Response:
The hashcode of a Java Object is simply a number, it is 32-bit signed int, that allows an object to be managed by a hash-based data structure. We know that hash code is an unique id number allocated to an object by JVM.
But actually speaking, Hash code is not an unique number for an object.

If two objects are equals then these two objects should return same hash code.

So we have to implement hashcode() method of a class in such way that if two objects are equals, ie compared by equal() method of that class, then those two objects must return same hash code.

If you are overriding hashCode you need to override equals method also.

QUESTION 129

Question : What is hashCode?

Response:
The hashcode of a Java Object is simply a number, it is 32-bit signed int, that allows an object to be managed by a hash-based data structure.

We know that hash code is an unique id number allocated to an object by JVM.
But actually speaking, Hash code is not an unique number for an object.

If two objects are equals then these two objects should return same hash code.

So we have to implement hashcode() method of a class in such way that if two objects are equals, ie compared by equal() method of that class,
then those two objects must return same hash code. If you are overriding hashCode you need to override equals method also.

QUESTION 130

Question : What is the difference between Hashtable and HashMap?

Response:
The basic differences are Hashtable is synchronized and HashMap is not synchronized.
Hashtable does not allow null values, and HashMap allows null values.

QUESTION 131

Question : What are the restrictions when overriding a method?

Response:
Overriding methods must have the same name, parameter list, and same return type. i.e., they must have the exact signature of the method we are going to override, including return type.

The overriding method cannot be less visible than the method it overrides. i.e., a public method cannot be override to private.
The overriding method may not throw any exceptions that may not be thrown by the overridden method.

QUESTION 132

Question : What is the use of assert keyword?

Response:
Java assertion feature allows developer to put assert statements in Java source code to help unit testing and debugging.
Assert keyword validates certain expressions.
It replaces the if block effectively and throws an AssertionError on failure.

QUESTION 133

Question : What is adapter class?

Response:
An adapter class provides the default implementation of all methods in an event listener interface.
Adapter classes are very useful when you want to process only few of the events that are handled by a particular event listener interface.
You can define a new class by extending one of the adapter classes and implement only those events relevant to you.

QUESTION 134

Question : What is difference between break, continue and return statements?

Response:
The break statement results in the termination of the loop, it will come out of the loop and stops further iterations.

The continue statement stops the current execution of the iteration and proceeds to the next iteration.

The return statement takes you out of the method. It stops executing the method and returns from the method execution.

QUESTION 135

Question : What is the difference between while and do-while statements?

Response:
The while statement verifies the condition before entering into the loop to see whether the next loop iteration should occur or not.

The do-while statement executes the first iteration without checking the condition, it verifies the condition after finishing each iteration.

The do-while statement will always execute the body of a loop at least once.

QUESTION 136

Question : When does the compiler provides the default constructor?

Response:
The compiler provides a default constructor if no other constructors are available in the class.
In case the class contains parametarized constructors, compiler doesnot provide the default constructor.

QUESTION 137

Question : What are the differences between C++ and Java ?

Response:
Java doesnot support pointers. Pointers are tricky to use and troublesome.

Java does not support multiple inheritances because it causes more problems than it solves. Instead Java supports multiple interface inheritance, which allows an object to inherit many method signatures from different interfaces with the condition that the inheriting object must implement those inherited methods. The multiple interface inheritance also allows an object to behave polymorphically on those methods.

Java does not include structures or unions.

Java does not support destructors but adds a finalize() method. Finalize methods are invoked by the garbage collector prior to reclaiming the memory occupied by the object, which has the finalize() method. This means you do not know when the objects are going to be finalized. Avoid using finalize() method to release non-memory resources like file handles, sockets, database connections etc because Java has only a finite number of

these resources and you do not know when the garbage collection is going to kick in to release these resources through the finalize() method.

All the code in Java program is encapsulated within classes therefore Java does not have global variables or functions.

C++ requires explicit memory management, while Java includes automatic garbage collection.

QUESTION 138

Question : What are the advantages of java package ?

Response:
Java packages helps to resolve naming conflicts when different packages have classes with the same names. This also helps you organize files within your project.

For example, java.io package do something related to I/O and java.net package do something to do with network and so on.
If we tend to put all .java files into a single package, as the project gets bigger, then it would become a nightmare to manage all your files.

QUESTION 139

Question : What is dynamic class loading?

Response:
Dynamic loading is a technique for programmatically invoking the functions of a class loader at run time.
Let us look at how to load classes dynamically by using
Class.forName (String className); method, it is a static method.

The above static method returns the class object associated with the class name.
The string className can be supplied dynamically at run time.
Once the class is dynamically loaded the class.newInstance () method returns an instance of the loaded class. It is just like creating a class object with no arguments.

A ClassNotFoundException is thrown when an application tries to load in a class through its class name, but no definition for the class with the specified name could be found.

QUESTION 140

Question : What happens if you do not provide a constructor?

Response:
Java does not actually require an explicit constructor in the class description.
If you do not include a constructor, the Java compiler will create a default constructor in the byte code with an empty argument.

QUESTION 141

Question : Difference between shallow cloning and deep cloning of objects?

Response:
The default behavior of an object's clone() method automatically yields a shallow copy. So to achieve a deep copy the classes must be edited or adjusted.

Shallow copy:
Generally clone method of an object, creates a new instance of the same class and copies all the fields to the new instance and returns it. This is called shallow copy.
Object class provides a clone method and provides support for the shallow copy. It returns 'Object' as type and you need to explicitly cast back to your original object.
Since the Object class has the clone method, you cannot use it in all your classes. The class which you want to be cloned should implement clone method and overwrite it.
It should provide its own meaning for copy or to the least it should invoke the super.clone().
Also you have to implement Cloneable marker interface or else you will get CloneNotSupportedException.
When you invoke the super.clone() then you are dependent on the Object class's implementation and what you get is a shallow copy.

Deep copy:
When you need a deep copy then you need to implement it yourself. When the copied object contains some other object its references are copied recursively in deep copy. When you implement deep copy be careful as you might fall for cyclic dependencies. If you don't want to implement deep copy yourselves then you can go for serialization. It does implements deep copy implicitly and gracefully handling cyclic dependencies.

QUESTION 142

Question : Can we have interfaces with no defined methods in java?

Response:
The interfaces with no defined methods act like markers.
They just tell the compiler that the objects of the classes implementing the interfaces with no defined methods need to be treated differently.

Marker interfaces are also known as “tag” interfaces.

QUESTION 143

Question : What is the difference between “==” and equals() method?

Response:
The == (double equals) returns true, if the variable reference points to the same object in memory. This is called “shallow comparison”.

The equals() method calls the user implemented equals() method, which compares the object attribute values. The equals() method provides “deep comparison” by checking if two objects are logically equal as opposed to the shallow comparison provided by the operator ==.

If equals() method does not exist in a user supplied class then the inherited Object class's equals() method will be called which evaluates if the references point to the same object in memory. In this case, the object.equals() works just like the "==" operator.

QUESTION 144

Question : How can you create an immutable class in java?

Response:
Here are the steps to create immutable class:

Declare the class as final, we can not extend the final class.

public final class MyTestImmutable { ... }

Declare all fields as final. Final fields can not be changed once its assigned.

private final int salary;

Do not provide any method which can change the state of the object, for example the setter methods which changes the values of the instance variables.

The “this” reference is not allowed to escape during construction from the immutable class and the immutable class should have exclusive access to fields that contain references to mutable objects like arrays, collections and mutable classes like Date etc by:
Declaring the mutable references as private.
Not returning or exposing the mutable references to the caller.

QUESTION 145

Question : What are access modifiers in java?

Response:
public:
A class or interface may be accessed from outside the package. Constructors, inner classes, methods and field variables may be accessed wherever their class is accessed.

protected:
Accessed by other classes in the same package or any subclasses of same package or different package.

private:
Accessed only within the class in which they are declared.

no modifier (default modifier):
Accessed only with in the class.

-->>and the in the same package ??

QUESTION 146

Question : Can we have private constructor in java?

Response:
Private constructor is used if you do not want other classes to instantiate the object.

Private constructors are used in singleton design pattern, factory method design pattern.

QUESTION 147

Question : Why do we need generics in java?

Response:
Code that uses generics has many benefits over non-generic code:

1) Stronger type checks at compile time: A Java compiler applies strong type checking to generic code and issues errors if the code violates type safety. Fixing compile-time errors is easier than fixing runtime errors,

which can be difficult to find.

2) Elimination of casts: If you use generics, then explicit type casting is not required.

3) Enabling programmers to implement generic algorithms: By using generics, programmers can implement generic algorithms that work on collections of different types, can be customized, and are type safe and easier to read.

QUESTION 148

Question : What is the difference between a product and a project?

Response:
A project is an endeavor with a clear definition of what needs to be delivered and the date when it needs to be delivered.
Actually, it may seem that a product is a project, but it is not. Because, there is no clear definition of what needs to be delivered and there is no clear definition of the the date when it needs to be delivered.

The product backlog is a collection of all possible ideas and additions to a product. The stories in the product backlog have no particular priority or schedule, although you can sort and organize them in hierarchies in the breakdown view by drag & drop. Thus, if you wish to prioritize your stories but don't want to create projects or iterations, just use the breakdown view!

Products are developed in projects.

QUESTION 149

Question : How does substring() method works on a string?

Response:
String in java is a sequence of characters.
String is more like a utility class which works on that character sequence. This character sequence is maintained as a array called value[], for example

```
private final char value[];
```

String internally defines two private variables called offset and count to manage the char array. The declarations can be as shown below:

```
/** The offset is the first index of the storage that is used. */
private final int offset;
```

/** The count is the number of characters in the String. */
private final int count;

Everytime we create a substring from any string object, substring() method assigns the new values of offset and count variables.

The internal char array is unchanged. This is a possible source of memory leak if substring() method is used without care.

QUESTION 150

Question : What is the difference between a Java Library and a framework?

Response:
A library is a collection of class definitions and its implementations.
The main benifits of creating library is simply code reuse.
A simple example is one of the other developer written code for sending emails. If you think it is a generic module.
Most of the places this code can be reusable. If we can make it a library (jar), we can include this library in our code, and call those methods.
The classes and methods normally define specific operations in a domain specific area.

In framework, all the control flow is already defined, and there is a bunch of predefined places that you should fill out with your code.
We use framework to develope applications.
A framework defines a skeleton where the application defines its own features to fill out the skeleton.
In this way, your code will be called by the framework when appropriately.
The benefit is that developers do not need to worry about if a design is good or not, but just about implementing domain specific functions.

QUESTION 151

Question : In Java, if we do not specify any value for local variables, then what will be the default value of the local variables?

Response:
Java does not initialize local variables with any default value. So
these variables will be just null by default.

QUESTION 152

Question : Can we write main method as public void static instead of public static void?

Response:
No, you cannot write it like this. Any method has to first specify the modifiers and then the return value. The order of modifiers can change.

We can write static public void main() instead of public static void main().

QUESTION 153

Question : Do you think 'main' used for main method is a keyword in Java?

Response:
No, main is just a name of method. There can be multiple methods with same name main in a class file. It is not a keyword in Java