Core Java

Q1. What is polymorphism? How polymorphism can be achieved in java?
In object-oriented programming concept, polymorphism refers to the ability of an object to take on many forms.
In java, polymorphic behaviour is achieved though overriding parent class method with child class method. If one child class object is assigned to parent class reference, and the method is invoked on the parent class reference, then child class method will be called at runtime.

Q2. What are the java access modifiers?
There are 4 access modifiers available in java:
1. Private : can only be accessible from the class it is declared
2. Default or no modifier or package private : can be accessible from any class of the same package
3. Protected : can be accessed from any class of the package or subclass on the class
4. Public : can be accessed from anywhere

Q3. While iterating a list using for-each loop, can we remove element from the list? If not, what can be used instead of for-each loop?
No, if we try to remove element, it will throw ConcurrentModificationException. To avoid this, we have to first get the iterator of the list and then loop though the iterator. In this case, we can be safely remove element while iterating.

Q4. In java exception handling, can we have finally block just after try block without having a catch block?
Yes, we can omit catch block if we want choose not to handle exception from try block (leave it to the caller to handle), but want to release resources or doing clean-ups in finally block in case exception occurs.

Q5. What is annotation? Give few examples of commonly used annotations.
Annotation is metadata for java code. It provides some additional information which could be used by compiler, jvm or processing logic.
Some commonly used annotations are @Override, @Deprecated, @Test, @Bean.

Q6. How can you make an object immutable?
An object can be made immutable by having:
- No setter method
- Make fields private and final
- Subclass should not override methods.
- Don't provide methods to modify mutable objects.

Q7. Why insertion is faster in LinkedIn compared to ArrayList?
As linked list is structured as connected nodes, to insert new data all we need to do is move pointer from previous node and add pointer to next node. But as ArrayList is backed by an Array, to insert value, we have to shift all values from the index to the right. So it is slower than LinkedList.

Q8. Can we assign List of String into a List of Object variable?
For example : List<Object> var = new ArrayList<String>;
No, because list of Object can hold any type of Object. But actual list object is restricted to hold only String type. So it will give compile time error.

Q9. How make a class singleton?
To make a class singleton, it's constructor needs to be private, it should have one public static getter method to provide single instance of the class. The instance of the class can be created eagerly as private static instance variable or lazily in static getter method.

Q10. When you try to insert element in HashMap, when hash collision occurs and how is it handled?
By definition, hash collision means having same hash value for 2 different objects. When we try to insert element into HashMap, it calls hashcode() method of key to get hash value, based on which it finds the bucket location where the value needs to be stored. Now, if 2 different key objects have same hashcode, they will point to same bucket position. Then HashMap call equals() method to check if the key objects are equal or not. In this case, hashmap uses LinkedList structure for the bucket, and the new value is stored in the next available place in the LinkedList.
Q11. What is Decorator pattern? Give one example of decorator pattern.
Decorator pattern is a design pattern which allows us to add new functionality to an existing object without altering its structure. Decorator pattern is created by using wrapper class, which provides additional method to the wrapped object.
<Example>

Q12. What in lambda expression in Java 8?
Lambda expression is gateway of functional programming in java. This is a concise way of representing Single Method Interface (SAM), also called as Functional Interface. Lambda expression spares us from creating verbose implementation of anonymous inner class.

Multi-threading and Concurrency Interview Questions

Basic

What is the difference between calling wait() and sleep() method in Java multi-threading?

Though both wait and sleep introduce some form of pause in Java application, they are the tool for different needs. Wait method is used for inter thread communication, it relinquishes lock if waiting for a condition is true and wait for notification when due to an action of another thread waiting condition becomes false. On the other hand sleep() method is just to relinquish CPU or stop execution of current thread for specified time duration. Calling sleep method doesn't release the lock held by current thread

What is the difference between Runnable and Callable in Java?

Both Runnable and Callable represent task which is intended to be executed in a separate thread. Runnable is there from JDK 1.0 while Callable was added on JDK 1.5. Main difference between these two is that Callable's call() method can return value and throw Exception, which was not possible with Runnable's run() method. Callable return Future object, which can hold the result of computation.

There are three threads T1, T2, and T3? How do you ensure sequence T1, T2, T3 in Java?

Sequencing in multi-threading can be achieved by different means but you can simply use the join() method of thread class to start a thread when another one has finished its execution. To ensure three threads execute you need to start the last one first e.g. T3 and then call join methods in reverse order e.g. T3 calls T2. join and T2 calls T1.join, these ways T1 will finish first and T3 will finish last.

Similarity and Difference between CyclicBarrier and CountDownLatch in Java?

1.**CyclicBarrier** and **CountDownLatch** **are similar because** they wait for specified number of thread to reach certain point and make count/parties equal to 0. But,for completing wait in CountDownLatch specified number of threads must call **countDown()** method in java and for completing wait in CyclicBarrier specified number of threads must call **await()** method.

2.Constructor’s >

CountDownLatch(int ***count***) CountDownLatch is initialized with given ***count***. ***count*** specifies the number of events that must occur before latch is released.	**CyclicBarrier(**int **parties)** New CyclicBarrier is created where **parties** number of thread wait for each other to reach common barrier point, when all threads have reached common barrier point, **parties** number of waiting threads are released.

3.CyclicBarrier can be **awaited repeatedly**, but **CountDownLatch** can't be awaited repeatedly. i.e. once count has become 0 cyclicBarrier can be used again but CountDownLatch cannot be used again in java.

4.**CyclicBarrier** can be used to trigger event, but **CountDownLatch** can't be used to launch event. i.e. once count has become 0 cyclicBarrier can trigger event but CountDownLatch can't in java.

Medium

Given two threads , ThreadA and ThreadB, how are you going to make sure that ThreadB starts executing after 60 seconds ThreadA has started executing. Note: ThreadA for full execution takes roughly around 3 minutes.

public final synchronized void join(long millis): This java thread join method is used to wait for the thread on which it's called to be dead or wait for specified milliseconds.

```
ThreadA.start();
try {
ThreadA.join(60000);
} catch (InterruptedException e) {
e.printStackTrace();
}
TheadB.start();
```

What is the difference between synchronized and volatile keyword in Java

1. The volatile keyword in Java is a field modifier while synchronized modifies code blocks and methods.

2. Synchronized obtains and releases the lock on monitor's Java volatile keyword doesn't require that.

3. Threads in Java can be blocked for waiting for any monitor in case of synchronized, that is not the case with the volatile keyword in Java.

4. Synchronized method affects performance more than a volatile keyword in Java.

5. Since volatile keyword in Java only synchronizes the value of one variable between Thread memory and "main" memory while synchronized synchronizes the value of all variable between thread memory and "main" memory and locks and releases a monitor to boot. Due to this reason synchronized keyword in Java is likely to have more overhead than volatile.

6. You can not synchronize on the null object but your volatile variable in Java could be null.

7. From Java 5 writing into a volatile field has the same memory effect as a monitor release, and reading from a volatile field has the same memory effect as a monitor acquire

What is executor framework in java?

Executor and ExecutorService are used for following purposes >

- creating thread in java,
- starting threads in java,
- managing whole life cycle of Threads in java.

Executor creates pool of threads and manages life cycle of all threads in it.
In Executor framework, Executor interface and ExecutorService class are most prominently used in java.
Executor interface defines very important execute() method which executes command in java.
ExecutorService interface extends Executor interface.
An Executor interface provides following type of methods >

•methods for managing termination and

•methods that can produce a Future for tracking progress of tasks in java.

An Executor that provides methods to manage termination and methods that can produce a Future for tracking progress of one or more asynchronous tasks.

What are differences between execute() and submit() method of executor framework in java?

execute() method	**submit()** method
execute() method is defined in *Executor* interface in java.	**submit()** method is defined in *ExecutorService* interface in java.
It can be used for executing **runnable task in java in java**.	It can be used for executing **runnable task** or **callable task**, submitted callable returns future and Future's get method will return the task's result in java.

Signature of execute method is > *void **execute**(Runnable **task**)*	submit method has 3 forms > *<T> Future<T> **submit**(Callable<T> **task**)* Submits a callable ***task*** for execution. Method **returns** a Future which represents pending results of the task. Once task is completed Future's get method will return the task's result. *<T> Future<T> **submit**(Runnable **task**, T **result**)* Submits a Runnable ***task*** for execution. Method **returns** a Future which represents that task. Once task is completed Future's get method will return ***result***. *Future<?> **submit**(Runnable **task**)* Submits a Runnable ***task*** for execution. Method **returns** a Future which represents that task. Once task is completed Future's get method will return null.

Tough

How to stop a thread in Java?

It's easy to start a thread in Java because you have a start() method but it's difficult to stop the thread because there is no working stop() method. Well, there was a stop() method in Thread class, when Java was first released but that was deprecated later. In today's Java version, You can stop a thread by using a boolean volatile variable. Threads in Java start execution from run() method and stop, when it comes out of run() method, either normally or due to any exception. You can leverage this property to stop the thread. All you need to do is create a boolean variable. Your thread should check its value every iteration and comes out of the loop and subsequently from run() method if Exit is true

What is Semaphore ?Can you design a semaphore ?

A counting semaphore. Conceptually, a semaphore maintains a set of permits. Each acquire() blocks if necessary until a permit is available, and then takes it. Each release() adds a permit, potentially releasing a blocking acquirer. However, no actual permit objects are used; the Semaphore just keeps a count of the number available and acts accordingly.Semaphores are often used to restrict the number of threads than can access some (physical or logical) resource.

1.1) Custom Semaphore's constructors in java >

•SemaphoreCustom (int permits)

permits is the initial number of permits available.
This value can be negative, in which case releases must occur before any acquires will be granted, permits is number of threads that can access shared resource at a time.
If permits is 1, then only one threads that can access shared resource at a time.

CODE >

```
 public SemaphoreCustom(int permits) {
       this.permits=permits;
 }
```

1.2) Custom Semaphore's acquire() method :

•void acquire( ) throws InterruptedException

Acquires a permit if one is available and decrements the number of available permits by 1.
If no permit is available then the current thread waits until one of the following things happen >
>some other thread calls release() method on this semaphore or,
>some other thread interrupts the current thread.

CODE >

```
 public synchronized void acquire() throws InterruptedException {
       //Acquires a permit, if permits is greater than 0 decrements
       //the number of available permits by 1.
       if(permits > 0){
             permits--;
       }

       //permit is not available wait, when thread
       //is notified it decrements the permits by 1
       else{
             this.wait();
             permits--;
       }
  }
```

1.3) Custom Semaphore’s release() method :

•void release( )

Releases a permit and increases the number of available permits by 1.
For releasing lock by calling release() method it’s not mandatory that thread must have acquired permit by calling acquire() method.

CODE >

```
  public synchronized void release() {
       //increases the number of available permits by 1.
       permits++;

       //If permits are greater than 0, notify waiting threads.
       if(permits > 0)
            this.notifyAll();
  }
```

What is CountDownLatch? Can you design a Custom CountDownLatch in java**?**

A CountDownLatch is initialized with a given **count** .
count specifies the number of events that must occur before latch is released.
Every time a event happens **count** is reduced by 1. Once count reaches 0 latch is released.

1.1) Custom CountDownLatch’s **constructor** in java >

•CountDownLatch(int count)

CountDownLatch is initialized with given count.
count specifies the number of events that must occur before latch is released.

CODE >

```
  public CountDownLatchCustom(int count) {
       this.count=count;
  }
```

1.2) Custom CountDownLatch’s await() method in java:

•void await( ) throws InterruptedException

Causes the current thread to wait until one of the following things happens-

•latch count has down to reached 0, or

•unless the thread is interrupted.

CODE >

```
  public synchronized void await() throws InterruptedException {
        //If count is greater than 0, thread waits.
        if(count>0)
            this.wait();
  }
```

1.3) Custom CountDownLatch's countDown() method in java:

- void countDown( )

Reduces latch count by 1.
If count reaches 0, all waiting threads are released.

CODE >

```
  public synchronized void countDown() {
        //decrement the count by 1.
        count--;

        //If count is equal to 0, notify all waiting threads.
        if(count == 0)
            this.notifyAll();
  }
```

There are two methods printA inside class A and printB inside class B. The method printA is a static method and print B is a non static method. Both methods are synchronized. There are exact 100 million threads fired on both printA and printB each. Which method execution will take less time?

```
Public class A {
        public static void synchronized printA(){
                …///…
        }
}
public class B {
        public void synchronized printB(){
                ….////…..
        }
}
```

There exists only one copy of each method per class, be the method static or non-static. A thread needs to acquire the monitor of an appropriate object before entering a synchronized method which it releases when the thread returns from the method. In case of static synchronization the thread acquires the monitor of the class object, the locking is at class level so no other thread can execute it until the current thread releases the lock. In case of non-static synchronisation the thread acquires the monitor of that particular instance on

which the call was made so other threads using different instances are not locked. They acquire monitors of there respective instances but continue to execute, as no two thread use the same instance. The non-static part will work faster i.e printB(). In case of the method being static, this monitor is class-specific, so essentially every thread tries to acquire a single monitor.
In case the method is non static like printB() each of the objects have their own monitors for the method. Since every thread is the

Spring Core:

1)What do you understand by Dependency Injection?
Dependency Injection design pattern allows us to remove the hard-coded dependencies and make our application loosely coupled, extendable and maintainable. We can implement dependency injection pattern to move the dependency resolution from compile-time to runtime.
Some of the benefits of using Dependency Injection are: Separation of Concerns, Boilerplate Code reduction, Configurable components and easy unit testing.

2)What are the common implementations of the ApplicationContext?
The three commonly used implementation of 'Application Context' are:
FileSystemXmlApplicationContext: This container loads the definitions of the beans froman XML file. Here we need to provide the full path of the XML bean configuration file to the constructor.
ClassPathXmlApplicationContext: This container loads the definitions of the beans from an XML file. Here we do not need to provide the full path of the XML file but we need to set CLASSPATH properly because this container will look bean configuration XML file in CLASSPATH.
WebXmlApplicationContext: This container loads the XML file with definitions of all beans from within a web application.

3)How do you implement caching in Spring framework? How do you remove cache?
Use @EnableCaching at class level,then
Add the annotation on the method we would need to cache results,for instance

```
@Cacheable("employee")
public Employee findEmployee(int empId) {
...
}
```

To remove cache, forinstance if the Employee is deleted from a dao method,then
We need to have @CacheEvict in the delete method.

4)What are the types of IOC container in spring?
There are two types of IOC containers in spring framework.
BeanFactory
ApplicationContext

4a)What is the difference between Bean Factory and ApplicationContext?
Following are some of the differences:
Application contexts provide a means for resolving text messages, including support for i18nof those messages.
Application contexts provide a generic way to load file resources, such as images.
Application contexts can publish events to beans that are registered as listeners.
Certain operations on the container or beans in the container, which have to be handled in aprogrammatic fashion with a bean factory, can be handled declaratively in an application context.
The application context implements MessageSource, an interface used to obtain localized messages, with the actual implementation being pluggable.

5)What bean scopes does Spring support? Explain them.

The Spring Framework supports following five scopes, three of which are available only if we usea web-aware pplicationContext.
singleton: This scopes the bean definition to a single instance per Spring IoC container.
prototype: This scopes a single bean definition to have any number of object instances.
request: This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.
session: This scopes a bean definition to an HTTP session. Only valid in the context of a webaware Spring ApplicationContext.
global-session: This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.
For experienced:
5a)What is the disadvantage of Spring singleton scope? How do you ensure thread safety?
It is not thread safe, and hence cannot be used in multithreaded environement,
we need to change the bean scope from singleton scope to prototype to ensure the thread safety.

5b)What is the reason for singleton bean not providing thread safety?
The default scope of Spring bean is singleton, so there will be only one instance per context. That means that all the having a class level variable that any thread can update will lead to inconsistent data. Hence in default mode spring beans are not thread-safe.
However we can change spring bean scope to request, prototype or session to achieve thread-safety at the cost of performance. It's a design decision and based on the project requirements.

Spring JDBC:

What template does Spring JDBC provide to access database?
1. JdbcTemplate
2. SimpleJdbcTemplate
3. NamedParameterJdbcTemplate

7)What are the advantages of JdbcTemplate in spring?
Less code: By using the JdbcTemplate class, we don't need to create connection,statement,start transaction,commit transaction and close connection to execute different queries. We can execute the query directly.
So what we have to do is just define the connection parameters and specify the SQL statement to be executed and do the required work for each iteration while fetching data from the database.

For experienced:
7a)Can you explain more about dbcTemplate Class?
The JDBC Template class executes SQL queries, updates statements, stores procedure calls, performs iteration over ResultSets, and extracts returned parameter values. It also catches JDBC exceptions and translates them to the generic, more informative, exception hierarchy defined in the org.springframework.dao package.
Instances of the JdbcTemplate class are threadsafe once configured. So we can configure a single instance of a JdbcTemplate and then safely inject this shared reference into multiple DAOs.
A common practice when using the JDBC Template class is to configure a DataSource in our Spring configuration file, and then dependency-inject that shared DataSource bean into our DAO classes, and the JdbcTemplate is created in the setter for the DataSource.
Instances of the JdbcTemplate class are threadsafe once configured. This is important because it means that we can configure a single instance of a JdbcTemplateand then safely inject this shared reference into multiple DAOs (or repositories). The JdbcTemplate is stateful, in that it maintains a reference to a DataSource, but this state is not conversational state.
Eg) int countOfEmpNamedJoe = jdbcTemplate.queryForObject("select count(*) from Employee where first_name = ?", Integer.class, "Henry");
8)How can you fetch records by spring JdbcTemplate?

We can fetch records from the database by the query method of JdbcTemplate. There are two interfaces to do this:

1. ResultSetExtractor
2. RowMapper

For experienced:

8a) What is the difference between ResultSetExtractor and RowMapper

- ResultSetExtractor interface can be used to fetch records from the database. It accepts a ResultSet and returns the list.
- RowMapper interface allows to map a row of the relations with the instance of user-defined class. It iterates the ResultSet internally and adds it into the collection. So we don't need to write a lot of code to fetch the records as ResultSetExtractor
- RowMapper saves a lot of code becuase it internally adds the data of ResultSet into the collection.

RowMapper is a higher level interface than ResultSetExtractor. We would use the latter if you want to deal with the entire ResultSet, and translate that to some sort of returned object, whereas RowMapper pre-supposes that each row in the ResultSet will be mapped to a returned object of some sort. The callback will happen once for each row.

Spring MVC:

9)What is DispatcherServlet and ContextLoaderListener?

Spring's web MVC framework is, like many other web MVC frameworks, request-driven, designed around a central Servlet that handles all the HTTP requests and responses. Spring's DispatcherServlet however, does more than just that. It is completely integrated with the Spring IoC container so it allows we to use every feature that Spring has.

After receiving an HTTP request, DispatcherServlet consults the HandlerMapping (configuration files) to call the appropriate Controller. The Controller takes the request and calls the appropriate service methods and set model data and then returns view name to the DispatcherServlet. The DispatcherServlet will take help from ViewResolver to pickup the defined view for the request. Once view is finalized, The DispatcherServlet passes the model data to the view which is finally rendered on the browser.

10)Can we have multiple Spring configuration files?

Yes We can have multiple spring context files. There are two ways to make spring read and configure them.

a) Specify all files in web.xml file using contextConfigLocation init parameter.

```
<servlet>
    <servlet-name>spring</servlet-name>
    <servlet-class>
      org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>
        WEB-INF/spring-dao-hibernate.xml,
        WEB-INF/spring-services.xml,
        WEB-INF/spring-security.xml
      </param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>spring</servlet-name>
    <url-pattern>/</url-pattern>
```

```
   </servlet-mapping>
```
b) OR, we can import them into existing configuration file we have already configured.
```
<beans>
   <import resource="spring-dao-hibernate.xml"/>
   <import resource="spring-services.xml"/>
   <import resource="spring-security.xml"/>
        ... //
 </beans>
```

11)What is the ViewResolver class? What is the use of it and what is commonly used implementation?
ViewResolver is an interface to be implemented by objects that can resolve views by name. There are plenty of ways using which we can resolve view names. These ways are supported by various in-built implementations of this interface. Most commonly used implementation is InternalResourceViewResolver class. It defines prefix and suffix properties to resolve the view component

Spring Batch:

12)Explain the spring batch framework architecture.
Job is an entity that encapsulates the entire batch process and acts as a container for steps. A job combines multiple steps that belong logically together in a flow and allows for configuration of properties global to all steps such as restartability.
A job is wired using an XML configuration or java based configuration and it is referred as Job configuration. The job configuration contains:
- The name of the job.
- Definition and ordering of steps.
- Whether or not the job is restartable.

JobInstance represents a logical job run. Consider a batch job that runs every night, there is a logical JobInstance running everyday.
JobParameters are set of parameters used to start a batch Job. They also distinguish JobInstance from another.

13. What is tasklet in Spring batch framework?
The Tasklet is an interface which performs any single task such as setup resource, running a sql update, cleaning up resources etc.

14) List out some of the practical usage scenario of Spring Batch framework.
- Reading large number of records from a database, file, queue or any other medium, process it and store the processed records into medium, for example, database.
- Concurrent and massively parallel processing.
- Staged, enterprise message-driven processing.
- Sequential processing of dependent steps.
- Whole-batch transaction.
- Scheduled and repeated processing.

Technical advantages of using Spring Batch Framework from a Developer perspective.
- Batch framework leverages Spring programming model thus allows developers to concentrate on the business logic or the business procedure and framework facilitates the infrastructure.
- Clear separation of concerns between the infrastructure, the batch execution environment, the batch application and the different steps/proceses within a batch application.
- Provides common scenario based, core execution services as interfaces that the applications can implement and in addition to that framework provides its default implementation that the developers could use or partially override based on their business logic.
- Easily configurable and extendable services across different layers.
- Provides a simple deployment model built using Maven.

What are the important features of Spring Batch?
- Restorability: Restart a batch program from where it failed.

- Different Readers and Writers : Provides great support to read from text files, csv, JMS, JDBC, Hibernate, iBatis etc. It can write to JMS, JDBC, Hibernate, files and many more.
- Chunk Processing : If we have 1 Million records to process, these can be processed in configurable chunks (1000 at a time or 10000 at a time).
- Easy to implement proper transaction management even when using chunk processing.
- Easy to implement parallel processing. With simple configuration, different steps can be run in parallel.

Spring Boot:

15)What is Spring Boot?
First of all Spring Boot is not a framework, it is a way to ease to create stand-alone application with minimal or zero configurations. It is approach to develop spring based application with very less configuration. It provides defaults for code and annotation configuration to quick start new spring projects within no time.

For experienced:
They should explain more in details, the advantages of Spring boot like below
It leverages existing spring projects as well as Third party projects to develop production ready applications.
It provides a set of Starter Pom's build files which one can use to add required dependencies and also facilitate auto configuration.
It avoids writing lots of boilerplate Code, Annotations and XML Configuration.
It provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.
And about the starter pom:
Starter POMs are a set of convenient dependency descriptors that we can include in our application. We get a one-stop-shop for all the Spring and related technology that we need, without having to hunt through sample code and copy paste loads of dependency descriptors. For example, if we want to get started using Spring and JPA for database access, just include the spring-boot-starter-data-jpa dependency in our project, and we are good to go.

16) What is the use of @EnableAutoConfiguration and @SpringBootApplication? When do we use these?
@EnableAutoConfiguration annotation on a Spring Java configuration class causes Spring Boot to automatically create beans it thinks we need and usually based on classpath contents, it can easily override

@SpringBootApplication is the convenience annotation to be used in Spring Boot application and this is equivalent to having @Configuration, @EnableAutoConfiguration and @ComponentScan in the main class.

17) How to reload any changes on Spring Boot without having to restart server?
Include following maven dependency in the application.

```
<dependency>
 <groupId>org.springframework</groupId>
 <artifactId>springloaded</artifactId>
 <version>1.2.6.RELEASE</version>
</dependency>
```

And for the Automatic restart
Applications that use spring-boot-devtools will automatically restart whenever files on the classpath change. This can be a useful feature when working in an IDE as it gives a very fast feedback loop for code changes. 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>
```

This can be achieved using DEV Tools. With this dependency any changes we save, the embedded tomcat will restart. Spring Boot has a Developer tools (DevTools) module which helps to improve the productivity of developers. One of the key challenge for the Java developers is to auto deploy the file changes to server and auto restart the server. Developers can reload changes on Spring Boot without having to restart my server. This will eliminates the need for manually deploying the changes every time. The module DevTools does exactly what is needed for the developers.

Note: This module will be disabled in the production environment.

Hibernate questions:

Easy :

1. What are the three states of a Hibernate Persistence object?
   - Transient
   - Persistent
   - Detached

2. What are the core interfaces of Hibernate framework?
   i. Session
   ii. SessionFactory
   iii. Configuration
   iv. Transaction
   v. Query and Criteria interface.

3. 3 3. What is the difference between load and get methods?
   i. load() will throw an exception if an object with id passed to them is not found, but get() will return null. Another important difference is that load can return proxy without hitting the database unless required (when you access any attribute other than id) but get() always go to the database, so sometimes using load() can be faster than the get() method. It makes sense to use the load() method, if you know the object exists but get() method if you are not sure about object's existence.

Medium :

1. Explain Criteria API.
   - It is a simplified API provided by Hibernate to retrieve entities by composing Criterion objects. For e.g List employees = session.createCriteria(Employee.class).add(Restrictions.like("name", "c%")).list();

2. In Hibernate, how can I map a class to be immutable?
   - By marking the class as { mutable="false" }. The default value is always true.

3. What are the different types of caches available in Hibernate?
   First level cache
   - This is associated with the Session object. This is used by default on a per transaction basis. This is mainly used to reduce the number of SQL queries it needs to generate within a given transaction.

   Second level cache
   - This is associated with SessionFactoryObject.

Tough:

1. What is N+1 problem in Hibernate? Explain
   - The N+1 SELECT problem is a result of lazy loading and load on demand fetching strategy. In this case, Hibernate ends up executing N+1 SQL queries to populate a collection of N elements. For example, if you have a List of N Items where each Item has a dependency on a collection of Bid object. Now if you want to find the highest bid for each item then Hibernate will fire 1 query to load all items and N subsequent queries to load Bid for each item. So in order to find the highest bid for each item your application end up firing N+1 queries.
2. Explain light object mapping.
   - The entities are represented as classes that are mapped manually to relational tables. The code is hidden from the business logic using specific design patterns. This approach is good for applications with a less number of entities or applications with common, metadata-driven data models.
3. Explain the working of Second-Level cache.
   - Hibernate will first try to retrieve objects from the session and if it is unable to, it retrieves from the second-level cache. If again this also fails, the objects are retrieved from the database. The initialize() method which populates a proxy object, will attempt to hit the second-level cache before going to the database.

Database PL/SQL

##Easy Set##

1) What is DDL/DML/DCL? // Atleast DDL and DML should known
Ans::
DDL:: Data definition language(DDL) allows you to CREATE, ALTER and DELETE database objects such as schema, tables, view, sequence etc.
DML:: Data manipulation language makes user able to access and manipulate data. It is used to perform following operations.
Insert data into database, Retrieve data from the database,
Update data in the database, Delete data from the database
DCL:: Data control language allows you to control access to the database. It includes two commands GRANT and REVOKE.
GRANT: to grant specific user to perform specific task.
REVOKE: to cancel previously denied or granted permissions.

2) If there are 10 records in the Emp table and 5 records in the Dept table, how many rows will be displayed in the result of the following SQL query, Explain your answer:

Select * From Emp, Dept
Ans:: The query will result in 50 rows as a “cartesian product” or “cross join”, which is the default whenever the ‘where’ clause is omitted.

3) What is a key difference between Delete, Truncate and Drop?
Ans::
1.DELETE command is used to remove rows from a table. A WHERE clause can be used to only remove some rows. If no WHERE condition is specified, all rows will be removed. After performing a DELETE operation you need to COMMIT or ROLLBACK the transaction to make the change permanent or to undo it. Note that this operation will cause all DELETE triggers on the table to fire.
2.TRUNCATE removes all rows from a table. The operation cannot be rolled back and no triggers will be fired. As such, TRUCATE is faster and doesn't use as much undo space as a DELETE.

3.DROP command removes a table from the database. All the tables' rows, indexes and privileges will also be removed. No DML triggers will be fired. The operation cannot be rolled back.
NOTE:: DROP and TRUNCATE are DDL commands, whereas DELETE is a DML command. Therefore DELETE operations can be rolled back (undone), while DROP and TRUNCATE operations cannot be rolled back.

4) Explain different JOINs.
Ans::
1.INNER JOIN:: The INNER JOIN selects all rows from both participating tables as long as there is a match between the columns. A SQL INNER JOIN is same as JOIN clause, combining rows from two or more tables
2.LEFT JOIN (or LEFT OUTER JOIN):: Returns all rows from the left table, and the matched rows from the right table; i.e., the results will contain all records from the left table, even if the JOIN condition doesn't find any matching records in the right table. This means that if the ON clause doesn't match any records in the right table, the JOIN will still return a row in the result for that record in the left table, but with NULL in each column from the right table.
3.RIGHT JOIN (or RIGHT OUTER JOIN):: Returns all rows from the right table, and the matched rows from the left table. This is the exact opposite of a LEFT JOIN; i.e., the results will contain all records from the right table, even if the JOIN condition doesn't find any matching records in the left table. This means that if the ON clause doesn't match any records in the left table, the JOIN will still return a row in the result for that record in the right table, but with NULL in each column from the left table.
4.FULL JOIN (or FULL OUTER JOIN):: Returns all rows for which there is a match in EITHER of the tables. Conceptually, a FULL JOIN combines the effect of applying both a LEFT JOIN and a RIGHT JOIN; i.e., its result set is equivalent to performing a UNION of the results of left and right outer queries.
5.CROSS JOIN:: Returns all records where each row from the first table is combined with each row from the second table (i.e., returns the Cartesian product of the sets of rows from the joined tables). Note that a CROSS JOIN can either be specified using the CROSS JOIN syntax ("explicit join notation") or (b) listing the tables in the FROM clause separated by commas without using a WHERE clause to supply join criteria ("implicit join notation").

6.SELF JOIN:: A self join is a join in which a table is joined with itself (which is also called Unary relationships), especially when the table has a FOREIGN KEY which references its own PRIMARY KEY. To join a table itself means that each row of the table is combined with itself and with every other row of the table.

##Moderate Set##
1) What is the difference between UNION and JOIN?
Ans:: Joins and Unions can be used to combine data from one or more tables. The difference lies in how the data is combined.
In simple terms, joins combine data into new columns. If two tables are joined together, then the data from the first table is shown in one set of column alongside the second table's column in the same row.
Unions combine data into new rows. If two tables are "unioned" together, then the data from the first table is in one set of rows, and the data from the second table in another set. The rows are in the same result.

2) What is the difference between UNION and UNION ALL?
Ans:: UNION merges the contents of two structurally-compatible tables into a single combined table. The difference between UNION and UNION ALL is that UNION will omit duplicate records whereas UNION ALL will include duplicate records.
It is important to note that the performance of UNION ALL will typically be better than UNION, since UNION requires the server to do the additional work of removing any duplicates. So, in cases where is is certain that there will not be any duplicates, or where having duplicates is not a problem, use of UNION ALL would be recommended for performance reasons.

3) What is the difference between procedure and function?
Ans::

1.Function must return a value but in Stored Procedure it is optional(Procedure can return zero or n values)
2.Functions can be called from Procedure whereas Procedures cannot be called from Function.
3.Procedure allows SELECT as well as DML(INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it.
4.Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section whereas Function can be.

##Tough Set##
1) What is index?
Ans:: Index is used to increase the performance and allow faster retrieval of records from the table. An index creates an entry for each value and it will be faster to retrieve data.
There are three types of Indexes in SQL: Unique Index, Clustered Index, NonClustered Index

Unique Index:: This indexing does not allow the field to have duplicate values if the column is unique indexed. Unique index can be applied automatically when primary key is defined.
Clustered Index:: The clustered index is used to reorder the physical order of the table and search based on the key values. Each table can have only one clustered index.
NonClustered Index:: NonClustered Index does not alter the physical order of the table and maintains logical order of data.

2) What is the difference between clustered and non clustered index in SQL?
Ans::
There are mainly two type of indexes in SQL, Clustered index and non clustered index. The differences between these two indexes is very important from SQL performance perspective.
1.One table can have only one clustered index but it can have many non clustered index.
2.Clustered index determines how data is stored physically in table. Actually clustered index stores data in cluster, related data is stored together so it makes simple to retrieve data.
3.Reading from a clustered index is much faster than reading from non clustered index from the same table.
4.Clustered index sort and store data rows in the table or view based on their key value, while non cluster have a structure separate from the data row.

3) What is the difference between the WHERE and HAVING clauses?
Ans::
When GROUP BY is not used, the WHERE and HAVING clauses are essentially equivalent. However, when GROUP BY is used:
1.The WHERE clause is used to filter records from a result. The filtering occurs before any groupings are made.
2.The HAVING clause is used to filter values from a group (i.e., to check conditions after aggregation into groups has been performed).

UNIX

1) What is chmod?
Ans:: chmod is the command and system call which is used to change the access permissions to file system objects (files and directories).
2) Which command is use to check the current directory?
Ans:: pwd (Print Working Directory)
3) In a file word UNIX is appearing many times? How will you count number?
Ans:: grep -c "Unix" filename

Java Questions

1. Why serializable is an interface.
2. How ConcurrentHashMap internally works.
3. How to design custom ConcurrentHashMap .
4. Which data structure you will use to create custom ConcurrentHashMap.
5. Why their is a need of synchronization.
6. Static synchronize method vs non static synchronize method.
7. How locking works in synchronization.
8. What is the use of volatile.
9. Does it requires to synchronize volatile getter and setter methods.
10. What is thread Call-Stack.
11. Let say two singleton object are already persisted into the files and we want to deserialize them.
12. Restful Web-Service paradigm.

13 Remove duplicates from an ArrayList
14 Immutability
15 ArrayList vs LinkedList – when to use what
16 Difference between Executor.submit vs execute –
17 Spring Scope – Prototype vs Singleton
18 Collections – Failsafe vs Failfast
19 Exception handling in multi-threaded applications
20 Overloading

Also revise the below topics from geeksforgeeks.org website

1. OOPS Concepts
   a. A.P.I.E.

2. Core Java

   a. Abstract Class
   b. Inner class / static inner class
   c. Immutable class
   d. Interface
   e. Marker Interface
   f. Exception handling
   g. Garbage Collection
   h. Class loader/ static and dynamic loading
   i. Comparable / Comparator Interface
   j. String pool / String Buffer/ String Builder
   k. Constructor chaining / in case of abstract class/ interface
   l. This and super keywords
   m. Serialization
   n. Iterator / List Iterator
   o. Rules of overloading and overriding
   p. Reflection
   q. Exception Hierarchy and finally Method, ConcurrentModificationException
   r. Pass by value / pass by reference
   s. Locking(class level vs instance level)
   t. Synchronization
   u. Locking mechanisms
   v. Serializable & Externalizable and cloanable Interfaces
   w. Finalize/clone method of object class
3. Collection Framework(internals)

a. Array / Array List
b. Linked List
c. Vector (rarely)
d. Hash Map
e. hash Table
f. Linked Hash Map
g. Tree Map
h. Sorted Map
i. WeakHashMap
j. All kinds of Sets/ Hash Set/Tree Set
k. LinkedHashSet
l. Stack / Queue/ Priority Queue/ Blocking Queue
m. Condition Interface
n. Producer – consumer , race condition
o. Fail safe and fail fast iterator
p. CopyOnWriteArrayList
q. ConcurrentSkipListMap
r. ConcurrentHashMap
s. Collections.unmodifiableCollection()

4. Design Patterns / Sorting Algorithms
    a. Singletons
    b. Visitor
    c. Template
    d. Decorator
    e. Strategy
    f. Observer
    g. Façade /session Façade
    h. Factory /Abstract Factory
    i. DAO

5. Spring Core
    a. Bean Factory
    b. Application Context
    c. Bean Life Cycle
    d. Init / destroy methods
    e. Bean Listeners
    f. Processors
    g. Scopes
    h. Loading mechanisms
    i. IOC

6. Database (SQL/PLSQL)
    a. DDL
    b. DML
    c. Delete/truncate/Drop
    d. Union / Union All
    e. Index/ clustered- non clustered index (including implementations at DS level)
    f. Procedure
    g. Group by/ having
    h. Count(*) Max , Avg, etc
    i. Join (types of joins)
    j. Primary Kay / Unique Key
    k. Isolation levels
    l. ACID properties

7. Java Performance Tuning

    a. GC algorithm names only
    b. Heap memory settings
    c. strong, soft, weak and Phantom reference
    d. Stack and Heap Concept

8. Analytical/Logical /Scenario Based questions.
    a. LRU dictionary or Cache
    b. ATM/Library/HR dept design
    c. Parking allocation
    d. Find most frequently used word from text file
    e. Sorting 10 MB file using 1 MB memory
    f. 1 billion cellphone numbers to finds duplicates
    g. Find duplicate number in Integer Array
    h. Identify palindrome
    i. Fibonacci series printing using recursive
    j. Calculate factorial using recursive and iterative
    k. Implement single elevator , double elevator
    l. Simulate DVD renting system
    m. etc

Sample questions below:

Question Set 1

1. Design a stack that supports getMin() in O(1) time and O(1) extra space.

2. Program for n'th node from the end of a Linked List

3. Semaphore in java 8, print odd and even number using semaphore

4. How ArrayList works internally in Java 8

5. find second largest number in array without sorting in java

6. Sort an array of 0s, 1s and 2s

7. Reverse a linked list

8. Garbage collection algorithms

9. Implement two stacks in an array

10. Producer-Consumer solution using threads in Java

Question Set 2

1. Implement database connection pooling using semaphore

2. Countdown latch/cyclic barrier -explain, difference between cyclic barrier and countdown latch

3. How HashMap works internally in Java 8

4. Function to check if a singly linked list is palindrome

5. Atomic variable -How it works internally

6. Difference between Callable and Runnable

7. Detect and Remove Loop in a Linked List

8. CopyOnWriteArrayList implementation

9. Find first unique character in a String

10. Implement Multithreading application which demonstrates deadlocks and how to avoid deadlocks.

Question Set 3:

1. Find position of an element in a sorted array of infinite numbers

2. How ConcurrentHashMap works internally in Java 8

3. BlockingQueue-Expalin, implement own ArrayBlockingQueue

4. ReentrantLock implementation

5. Intersection point of two Linked Lists.

6. Creating custom exceptions

7. Design a vending machine

8. Java Reference- Soft, Weak, Strong and Phantom

9. Sort an array of 0s, and 1s

10. Different and best approach for Singleton Pattern

Queue Set 4:

1. Search an element in a sorted and rotated array

2. How TreeSet works internally in Java 8

3. UnModifiable collection own implementation

4. Java 8 new features

5. largest-sum-contiguous-subarray

6. Tree traversal with implementation [preorder, postorder, inorder and mirror]

7. Design multi-level parking system

8. Map sort by value

9. Design Principle

10. find the middle element in a linked list

11. Implement StringPool -Flyweight Design Pattern

Real time Interview Questions

1. What is JVM,JRE,JDK
2. What is heap and stack
3. Define the storage-

   Emp e = new Emp(), and int a = 8;
   Where e will be stored and where a will be stored
4. Will my program work if I have jre not jdk
5. How do you compile java class
6. Difference between Array list and Linked List
7. Why Array List is preferred for accessing elements instead of Linked list, and what is time complexity of get method of array list
8. Which data structure implemented by array list and how it stores element
9. Is java is pass by value or pass by reference
10. Internal implementation of hash map.
11. What is hash collision?
12. Program.

```
Employee e = new Employee();
     public void a(){
     e.setName("Shubham");

      public void b(e){
      e.setName("Karim");

          }
     System.out.println(e.getName());

     }
```

13. Program -→ What is output of below code

    final Employee e = new Employee(); Employee e1 = new Employee(); ------e=b ;

14. Program -→ What is output of below code
    final int a=6; b=7; a=b;

15. Program→

Scenario 1—Will it work or not

```
    Employee e = new Employee();
         HashMap<Employee, String> hm = new HashMap<Employee,String>() ;
         TreeMap<Employee, String> tm = new TreeMap<Employee, String>();

         hm.put(e, "employee 1");
         tm.put(e, "employee 2");
```

Scenario 2- Will it work if Employee class doesn’t override equals and hashcode?

```
    Employee e = new Employee();
    Employee e1 = new Employee();
         HashMap<Employee, String> hm = new HashMap<Employee,String>() ;
         TreeMap<Employee, String> tm = new TreeMap<Employee, String>();

         hm.put(e, "employee 1");
```

```
                tm.put(e, "employee 2");
                hm.put(e1, "employee 1");
                tm.put(e1, "employee 2");
```

Scenario 2- Will it work if Employee class doesn't override equals and hashcode?
Employee e = new Employee();

```
                HashMap<Employee, String> hm = new HashMap<Employee,String>() ;
                TreeMap<Employee, String> tm = new TreeMap<Employee, String>();

                hm.put(e, "employee 1");
                tm.put(e, "employee 2");
        hm.put(e, "employee 3");
                tm.put(e, "employee 4");
```

16. What is interface and why variables are default static and final in interface?
17. How we iterate hashmap? Explain the code for iterating hashmap
18. What is volatile, ThreadLocal
19. Difference between synchronization and volatile
20. Where is stack memory located

Ishwar Shah

1.) Difference between JDK and JRE.
2.) Internal working of hashmap(deeply) and TreeMap (Deeply).
3.) Difference between Arraylist and LinkedList (Deeply).
4.) Complete Hierarchy of Throwable Class(Deeply).
5.) Understanding of wait and notify methods(deeply).
6.) synchronized keyword , volatile keyword.
7.) Stack and HEAP memories.
8.) Comparable and Comparator (which methods we need to override and signature of it).
9.) Semaphore and Reentrantlock.
10.) HashTable vs Concurrent Hashmap.
11.) Time Complexity for all collections.
12.) Write a program to reverse a linkedList.
13) Write program which perform same as splitwise application (Expense management application).
14) if we return from both try and finally , which will be return and what warning we will get.
15.) Can we override methods with same signature but in child class it is throwing exception.