1. **How hashing technique works internally?**

**Which data structure is used to implement HashMap? -> Array and LinkedList**

**Why the key in HashMap cannot be duplicated?**

[**https://www.javatpoint.com/working-of-hashmap-in-java**](https://www.javatpoint.com/working-of-hashmap-in-java)

[**https://www.geeksforgeeks.org/internal-working-of-hashmap-java**](https://www.geeksforgeeks.org/internal-working-of-hashmap-java)

1. **How HashSet works internally do avoid duplicates?**

[**https://www.geeksforgeeks.org/internal-working-of-sethashset-in-java/**](https://www.geeksforgeeks.org/internal-working-of-sethashset-in-java/)

1. **What improvements of Linked HashMap were introduced in Java 8?**

[**https://www.nagarro.com/en/blog/post/24/performance-improvement-for-hashmap-in-java-8**](https://www.nagarro.com/en/blog/post/24/performance-improvement-for-hashmap-in-java-8)

1. **Internal working and data structure used in LinkedHashMap, TreeMap and TreeSet.**

[**https://anmolsehgal.medium.com/java-linkedhashmap-internal-implementation-44e2e2893036**](https://anmolsehgal.medium.com/java-linkedhashmap-internal-implementation-44e2e2893036)

[**https://www.geeksforgeeks.org/internal-working-of-treemap-in-java/**](https://www.geeksforgeeks.org/internal-working-of-treemap-in-java/)

[**https://www.javatpoint.com/how-treemap-works-internally-in-java**](https://www.javatpoint.com/how-treemap-works-internally-in-java)

[**https://www.javatpoint.com/how-treeset-works-internally-in-java**](https://www.javatpoint.com/how-treeset-works-internally-in-java)

[**https://www.geeksforgeeks.org/treeset-in-java-with-examples/**](https://www.geeksforgeeks.org/treeset-in-java-with-examples/)

1. **What are advantages and disadvantages of immutability? Which design pattern implements it or which design pattern is immutable?**

[**https://www.baeldung.com/java-immutable-object**](https://www.baeldung.com/java-immutable-object)

[**https://dzone.com/articles/the-importance-of-immutability-in-java**](https://dzone.com/articles/the-importance-of-immutability-in-java)

[**https://lkumarjain.blogspot.com/2016/02/immutable-design-pattern.html**](https://lkumarjain.blogspot.com/2016/02/immutable-design-pattern.html)

**Design pattern:**

[**https://www.sitepoint.com/flyweight-design-pattern-immutability-perfect-match**](https://www.sitepoint.com/flyweight-design-pattern-immutability-perfect-match)

[**https://www.davideguida.com/immutable-builder-pattern/**](https://www.davideguida.com/immutable-builder-pattern/)

1. **Serialization and what is difference between custom made Serial UID and JVM given Serial UID?**

[**https://www.geeksforgeeks.org/serialization-in-java/**](https://www.geeksforgeeks.org/serialization-in-java/)

[**https://www.geeksforgeeks.org/serialversionuid-in-java/**](https://www.geeksforgeeks.org/serialversionuid-in-java/)

[**https://java2blog.com/java-serialization-interview-questions-and-answers/**](https://java2blog.com/java-serialization-interview-questions-and-answers/)

[**https://stackoverflow.com/questions/45761399/serialversionuid-without-l-xxxxxxxxxxxxxxxxxl-vs-1l-positive-vs-negative**](https://stackoverflow.com/questions/45761399/serialversionuid-without-l-xxxxxxxxxxxxxxxxxl-vs-1l-positive-vs-negative)

1. **How to enable annotations in Spring? What we define before using he annotations?**

[**https://www.tutorialspoint.com/spring/spring\_annotation\_based\_configuration.htm**](https://www.tutorialspoint.com/spring/spring_annotation_based_configuration.htm)

[**https://www.journaldev.com/2696/spring-interview-questions-and-answers?utm\_source=website&utm\_medium=menubar&utm\_campaign=Top-Menu-Bar**](https://www.journaldev.com/2696/spring-interview-questions-and-answers?utm_source=website&utm_medium=menubar&utm_campaign=Top-Menu-Bar)

1. **Callable in multithreading and call().**

[**https://www.geeksforgeeks.org/callable-future-java/**](https://www.geeksforgeeks.org/callable-future-java/)

[**https://dzone.com/articles/java-callable-future-understanding**](https://dzone.com/articles/java-callable-future-understanding)

1. **Executor Framework**

[**https://stackabuse.com/concurrency-in-java-the-executor-framework**](https://stackabuse.com/concurrency-in-java-the-executor-framework)

[**https://www.geeksforgeeks.org/java-util-concurrent-executor-interface-with-examples/**](https://www.geeksforgeeks.org/java-util-concurrent-executor-interface-with-examples/)

[**https://www.geeksforgeeks.org/java-util-concurrent-executorservice-interface-with-examples/**](https://www.geeksforgeeks.org/java-util-concurrent-executorservice-interface-with-examples/)

1. **Difference between primary key and unique key (Oracle database).**

Primary key will not accept NULL values whereas Unique key can accept one NULL value.

A table can have only primary key whereas there can be multiple unique key on a table.

A Clustered index automatically created when a primary key is defined whereas Unique key generates the non-clustered index

Primary Key is column which uniquely identifies each row where as unique key is a constraint which can be imposed on several columns which are not primary key to identify a group of tuples or rows.

1. **Difference between delete and truncate command.**

Truncate (DDL command) has auto commit it can’t be rollbacked and delete (DML command) can be rollbacked if not committed explicitly.

Truncate doesn’t has where clause and used to delete whole data of a table and need ALTER permission on table to do so.

Delete has a where clause and deletes specified rows only and one need DELETE permission on table.

1. **What are triggers?**

**Triggers** are the **SQL** statements that are automatically executed when there is any change in the database. The **triggers** are executed in response to certain events (INSERT, UPDATE or DELETE) in a particular table.

1. **Difference between function and procedures in database.**

Function only returns values but procedures can accept values and return as well.

1. **Heap memory vs stack memory?**

[**https://www.journaldev.com/4098/java-heap-space-vs-stack-memory**](https://www.journaldev.com/4098/java-heap-space-vs-stack-memory)

1. **Linkedlist vs arraylist. Time Complexity of both while performing different operations?**

[**https://www.baeldung.com/java-arraylist-linkedlist**](https://www.baeldung.com/java-arraylist-linkedlist)

[**https://www.geeksforgeeks.org/arraylist-vs-linkedlist-java/**](https://www.geeksforgeeks.org/arraylist-vs-linkedlist-java/)

1. **Spring bean life cycle and Bean Scopes?**

[**https://www.geeksforgeeks.org/bean-life-cycle-in-java-spring/**](https://www.geeksforgeeks.org/bean-life-cycle-in-java-spring/)

[**https://www.journaldev.com/21039/spring-bean-scopes**](https://www.journaldev.com/21039/spring-bean-scopes)

1. **What is memory leak in Java?**

[**https://www.baeldung.com/java-memory-leaks**](https://www.baeldung.com/java-memory-leaks)

1. **What are SOLID principles?**

[**https://www.javatpoint.com/solid-principles-java**](https://www.javatpoint.com/solid-principles-java)

[**https://www.baeldung.com/solid-principles**](https://www.baeldung.com/solid-principles)

1. **Test Pyramids or testing types or agile testing pyramids? Mockito & Junit?**

[**https://martinfowler.com/articles/practical-test-pyramid.html**](https://martinfowler.com/articles/practical-test-pyramid.html)

[**https://career.guru99.com/top-11-junit-interview-questions/**](https://career.guru99.com/top-11-junit-interview-questions/)

[**https://www.journaldev.com/20834/junit5-tutorial**](https://www.journaldev.com/20834/junit5-tutorial)

[**https://www.journaldev.com/21816/mockito-tutorial**](https://www.journaldev.com/21816/mockito-tutorial)

1. **Difference between JAR, WAR and EAR.**

[**https://pediaa.com/what-is-the-difference-between-jar-war-and-ear/**](https://pediaa.com/what-is-the-difference-between-jar-war-and-ear/)

1. **Difference between ListIterator and Iterator.**

**ListIterator has hasPrevious() method can traverse both ways-backwards and forward.**

1. **Code review best practices.**

[**https://javarevisited.blogspot.com/2011/09/code-review-checklist-best-practice.html**](https://javarevisited.blogspot.com/2011/09/code-review-checklist-best-practice.html)

1. **FIRST principles of unit testing.**

[**https://medium.com/@tasdikrahman/f-i-r-s-t-principles-of-testing-1a497acda8d6**](https://medium.com/@tasdikrahman/f-i-r-s-t-principles-of-testing-1a497acda8d6)

1. **Java data types.**

[**https://www.geeksforgeeks.org/data-types-in-java/**](https://www.geeksforgeeks.org/data-types-in-java/)

[**https://www.protechtraining.com/content/java\_fundamentals\_tutorial-data\_types**](https://www.protechtraining.com/content/java_fundamentals_tutorial-data_types)

1. **What is dependency injection in Java ?**

[**https://www.vogella.com/tutorials/DependencyInjection/article.html**](https://www.vogella.com/tutorials/DependencyInjection/article.html)

[**https://www.freecodecamp.org/news/a-quick-intro-to-dependency-injection-what-it-is-and-when-to-use-it-7578c84fa88f/**](https://www.freecodecamp.org/news/a-quick-intro-to-dependency-injection-what-it-is-and-when-to-use-it-7578c84fa88f/)

1. **When we override hashCode() method, we have to override equals() method as well. Is the vise-versa true?**

it is generally necessary to override the hashCode method whenever this method is overridden, so as to maintain the general contract for the hashCode method, which states that equal objects must have equal hash codes.

1. **Spring dependency injection and which way is better? Constructor or setter way?**

**Setter way is better**

Setter Injection and Constructor Injection have their own advantages and disadvantages. The good thing about Spring is that it doesn't restrict you to use either Setter Injection or Constructor Injection and you are free to use both of them in one Spring configuration file. Use Setter injection when a number of dependencies are more or you need readability. Use Constructor Injection when Object must be created with all of its dependency.

1. **Difference between Spring and Spring Boot.**
2. **What is polymorphism? Types of polymorphism – Runtime polymorphism and compile time polymorphism?**
3. **What is marker interface?**

Used to indicate JVM that this is special interface and has some specific behavior like Serializable and Cloneable interfaces.

1. **Java Overriding: widening and narrowing of access modifiers and exception handling for the same.**

[**https://www.geeksforgeeks.org/exception-handling-with-method-overriding-in-java/**](https://www.geeksforgeeks.org/exception-handling-with-method-overriding-in-java/)

[**https://www.benchresources.net/java-overriding-widening-and-narrowing-for-access-modifier-return-type-and-exception-handling/**](https://www.benchresources.net/java-overriding-widening-and-narrowing-for-access-modifier-return-type-and-exception-handling/)

1. **Why do we use? in generics?**

Because it could be any class or Object which is cleared at run time.

1. **How can we create standalone project in Java?**

Just by using public static void main(String[] args) method.

1. **Explain the hierarchy of exceptions?**

[**https://www.geeksforgeeks.org/exceptions-in-java/**](https://www.geeksforgeeks.org/exceptions-in-java/)

1. **Difference between Composition and Inheritance?**

[**https://www.geeksforgeeks.org/difference-between-inheritance-and-composition-in-java/**](https://www.geeksforgeeks.org/difference-between-inheritance-and-composition-in-java/)

1. **What are primitive and non-primitive data types?**

[**https://www.w3schools.com/java/java\_data\_types.asp#:~:text=Primitive%20types%20are%20predefined%20(already,operations%2C%20while%20primitive%20types%20cannot.**](https://www.w3schools.com/java/java_data_types.asp#:~:text=Primitive%20types%20are%20predefined%20(already,operations%2C%20while%20primitive%20types%20cannot.)

1. **Advantages and disadvantages of test-driven development (TDD)?**

[**https://www.geeksforgeeks.org/advantages-and-disadvantages-of-test-driven-development-tdd/**](https://www.geeksforgeeks.org/advantages-and-disadvantages-of-test-driven-development-tdd/)

1. **What steps should be taken to check if application is high memory usage?**

[**https://www.vogella.com/tutorials/JavaPerformance/article.html**](https://www.vogella.com/tutorials/JavaPerformance/article.html)

1. **What are some unix commands?**

[**unixtutorial.org/basic-unix-commands**](unixtutorial.org/basic-unix-commands)

[**http://mally.stanford.edu/~sr/computing/basic-unix.html**](http://mally.stanford.edu/~sr/computing/basic-unix.html)

1. **What is the way to avoid writing constructors and setters for a class?**

we can use Lombok Library where getter and setter and constructors can be created using annotations.

[**https://medium.com/javarevisited/stop-writing-getters-setters-and-constructors-in-java-e400325cd5c4**](https://medium.com/javarevisited/stop-writing-getters-setters-and-constructors-in-java-e400325cd5c4)

Or can use a design pattern- Builder Design Pattern (for immutability as well)

[**https://dzone.com/articles/the-builder-pattern-for-class-with-many-constructo**](https://dzone.com/articles/the-builder-pattern-for-class-with-many-constructo)