Interview Questions

1. What is the difference between abstract class and interfere?

Abstract class: An abstract class can have both abstract or concrete methods, it's used when we have related classes sharing common behavior, abstract and use with single inheritance only.

Interface: An interface method is need to implementation for classes that implementing the interface, used when you want to unrelated classes have to follow the same rules. A class can implement multiple interfaces.

2. What are the features in java 8?

- **Lambda expression** It allows to write functional code by passing behavior as parameters.
- **Stream API** Provide functional way to perform data processing with operations like filtering and mapping.
- **Method Reference** A short hand to call methods directly in lambda.
- **Optional class** It's helps to avoid NullPointerException by representing values that may or may not be present.
- **New Date and Time API** Provides a modern, immutable API for handling data and time.
- **Functional interfaces** Functional interfaces are used with lambda expression, and it has only one method in it.

3. What is Singleton and single design pattern?

Singleton class allows only one instance to be created throughout the lifecycle of application it's provide a global point of access to that instance. Features: - lazy initialization, global access, thread safety.

4. What is a constructor?

A constructor is used to initialize objects when they are created. It's a method that runs automatically when the object is created. Its name similar as a class name, it can't return any value not even void.

5. Tell me about oops concept?

Oops concept is based on the objects, which represent real world entity, and oops build on the for key principles: - Encapsulate, Inheritance, Polymorphism, Abstraction.

6. What is the marker interface?

to treat a class in a special way. Moden java replaces marker interfaces with annotations like @Deprecated or @Override.

7. What is HashMap?

HashMap is a data structure that stores data as key value pairs. It's allowing fast access using keys, with unique keys and optional null values. And it's not thread safe by default.

8. What is the difference between method overloading and overriding?

Overriding: - A method overriding has same method name and parameter but in sub class it resolved at runtime.

Overloading: - In method overloading in a class method has same method name but with different parameters in same class, and it resolved at compile time.

9. What is the difference between vector and arraylist?

Vector: - Vector is a thread safe collection and vector doubles in the size when it's full, but vector is slower due to Synchronization.

ArrayList: - ArrayList is not thread safe but faster and it's grown by 1.5 time, and arraylist is a non-synchronized collection.

10. What are the types of inheritance?

a. **Single inheritance: -** A class inherits from only one parents class.
b. **Multilevel inheritance: -** A class inherits from a class that is already a sub class of another class.
c. **Hierarchical inheritance: -** Multiple classes inherit from a single parent class.

11. What are the features in spring boot?

a. **Auto configuration: -** Automatically configures the application based on the dependencies.
b. **Spring boot starter templates: -** pre configured templates for common tasks like web development.
c. **Embedded server: -** Support embedded server like tomcat, eliminating the need for external servers.
d. **Production ready features: -** Offers tools for easy deployment and management in production environments.

12. What is the difference between RequestMapping and GetMapping ?

RequestMapping: - A general annotation used to handle all HTTP methods like get, post, put, delete, offering flexibility in mapping.

GetMapping: - GetMapping is specifically used to handle get request.

13. What is a spring bean lifecycle?

Spring first creates the bean and sets its properties and dependencies. Next, it runs any setup methods like **`@PostConstruct`.** After that, the bean is

ready to use. When the application stops, Spring cleans up the bean by calling **`@PreDestroy`.**

14. What are the basic annotations you have used in spring boot?

Core Annotations

1. **@SpringBootApplication**: Combines `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan` to set up the application.
2. **@Component**: Marks a class as a Spring-managed component.
3. **@Controller**: Marks a class as a Spring MVC controller.
4. **@Service**: Indicates a service class containing business logic.
5. **@Repository**: Marks a class as a data-access layer.

Dependency Injection

6. **@Autowired**: Injects dependencies automatically.
7. **@Qualifier**: Specifies which bean to inject when multiple options exist.

RESTful APIs

8. **@RestController**: Combines `@Controller` and `@ResponseBody` for REST APIs.
9. **@RequestMapping**: Maps HTTP requests to handler methods.
10. **@GetMapping, @PostMapping, @PutMapping, @DeleteMapping**: Simplified versions of `@RequestMapping` for specific HTTP methods.

Validation and Data Binding

11. **@RequestBody**: Binds HTTP request body to a method parameter.

12. **@PathVariable**: Binds a URL path variable to a method parameter.
13. **@RequestParam**: Binds query parameters to method parameters.
14. **@Valid**: Triggers validation on input data.

Database and Transactions

15. **@Entity**: Marks a class as a JPA entity.
16. **@Table**: Specifies the table name for a JPA entity.
17. **@Id**: Specifies the primary key.
18. **@GeneratedValue**: Defines primary key generation strategy.
19. **@Transactional**: Manages transactions.

15. Can we create non web applications?

Yes, Spring Boot can be used to create non-web applications. It is not limited to web-based projects; you can build **standalone, CLI or batch-processing applications** as well.

16. Can you tell me the Http status code that you have used?

2xx (Success)

- **200 OK**: Request succeeded.
- **201 Created**: Resource created successfully.
- **204 No Content**: Request succeeded, but no content returned.

4xx (Client Errors)

- **400 Bad Request**: Invalid request from the client.
- **401 Unauthorized**: Authentication required.
- **403 Forbidden**: Client does not have permission to access the resource.
- **404 Not Found**: Resource not found.
- **405 Method Not Allowed**: HTTP method not supported for the resource.

5xx (Server Errors)

- **500 Internal Server Error**: Unexpected server error.
- **503 Service Unavailable**: Server temporarily unavailable.

17. Do you know about maven and how maven works?

Maven is a build and dependency management tool for Java projects. It uses a `pom.xml` file to manage project configuration, dependencies, and build lifecycle.

How it works:

- **POM**: Defines project details and dependencies.
- **Lifecycle**: A sequence of build phases like `compile`, `test`, `package`, `install`, and `deploy`.
- **Dependencies**: Automatically handles and downloads required libraries from local or remote repositories.
- **Repositories**: Local (on your machine), Central (public), and Remote (custom).

Common Commands:

- `mvn clean`: Clean the build directory.
- `mvn compile`: Compile the source code.
- `mvn test`: Run tests.
- `mvn install`: Install the package locally.
- `mvn deploy`: Deploy the package remotely.

18. How to check maven using the command line?

To check if Maven is installed, just open the Command Prompt and type `mvn -v`. Press Enter, and it will show the Maven version, Java version, and OS details.

19. How to create the maven project?

To create a Maven project, first open the command prompt. Then run:

mvn archetype:generate -DgroupId=your.group.id -DartifactId=your-project-name -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
This will generate a new Maven project with the specified groupId, artifactId, and project name. After the project is created, navigate to the project folder and run:

mvn package

This will compile the code and create the package in the target folder.

20. Do you know about maven repository?

A **Maven repository** is a location where Maven stores and retrieves project dependencies. There are three types:

1. **Local Repository**: Stored on your machine.
2. **Central Repository**: A public, default repository that Maven uses to fetch commonly used libraries.
3. **Remote Repository**: Custom repositories that organizations use to store their own libraries or artifacts.

Maven uses the `pom.xml` file to manage dependencies, and when a project is built, it automatically downloads the necessary libraries from the local or remote repositories.

21. What is the difference between GIT and GITHUB?

- **Git** is the tool you use on your local machine to manage code versions.
- **GitHub** is a cloud service that hosts your Git repositories and helps you collaborate with others.

22. How to change the last commit?

The **-- amend** option in Git is used to modify the last commit. it allows to change the commit message or add new changes to the previous commit to change the commit message we used **git commit --amend** this opens the default editor to modify the commit message. and if you want to change the content of the last commit, we use git add . and then **git commit --amend** this update the last commit with new changes.

23. How to create git repository?

- Go to your project folder in the terminal.
- Type `git init` and hit Enter. This sets up a Git repository in your project.
- Add your files by typing `git add .` to stage everything for tracking.
- Make your first commit with `git commit -m "Initial commit"`.
- If you want to connect to GitHub or GitLab, create a repository there.
- Copy the repository URL and link it by running `git remote add origin <repository-URL>`.
- Push your code with `git push -u origin main`.

24. Explain me about rebase?

Rebase is a Git command used to move your branch's commits on top of another branch to create a cleaner, more linear commit history. To do this, you first switch to your branch using `git checkout your-branch-name`. Then, run `git rebase other-branch-name`. This updates your branch with the changes from the other branch and places your commits on top of them.

25. What is git stash?

Git stash is a command used to temporarily save your changes without committing them. It's helpful when you want to switch branches or work on something else but don't want to lose your current changes.

When you run `git stash`, Git saves your changes in a stack-like structure and reverts your working directory to a clean state. Later, you can reapply the stashed changes using `git stash apply` or remove them completely with `git stash drop`.

It's like putting your work on hold while you focus on something else.

26. Can you tell me what is the difference between git reversing and resetting?

- **Git Revert**:
  - It creates a new commit that undoes changes made by a previous commit.
  - History stays intact, making it safer for shared branches.
  - Example: `git revert <commit-hash>`.
- **Git Reset**:
  - It moves the branch pointer to a different commit, effectively erasing commits from history.
  - Changes can be kept (`--soft`) or discarded (`--hard`).
  - Example: `git reset --hard <commit-hash>`.

In short, *revert* adds a new "undo" commit, while *reset* rewrites history by removing commits.

27. What is the difference between checked and unchecked exceptions?

A **checked exception** is an exception that is checked at compile-time. You must handle it either by using a `try-catch` block or declaring it with `throws` in the method signature. Examples of checked exceptions include `IOException` and `SQLException`.

An **unchecked exception** is an exception that is not checked at compile-time. These exceptions inherit from `RuntimeException`. You don't have to handle them, but you can if needed. Examples of unchecked exceptions are `NullPointerException` and `ArrayIndexOutOfBoundsException`.

28. Tell me about user defined exception?

A **user-defined exception** in Java is an exception that you create to handle specific error conditions in your application. It allows you to define custom behavior for errors that are not covered by standard exceptions.

To create a user-defined exception:

1. **Extend an existing exception class** like `Exception` or `RuntimeException`.
2. **Define a constructor** to initialize the exception with a message.

29. What is the difference between exception and error?

An **exception** is a condition that occurs during the program's execution, which the program can handle. It is usually caused by logical issues such as invalid input or missing files. You can handle exceptions using `try-catch` blocks. Some examples are `IOException`, `SQLException`, and `NullPointerException`.

An **error** is a serious issue that usually cannot be handled by the program. Errors are often caused by system problems, like running out of memory or stack overflow. These issues are beyond the control of the program, and you generally cannot recover from them. Examples of errors include `OutOfMemoryError` and `StackOverflowError`.

30. What is the difference between throw and throws?

throw is used to explicitly **throw an exception** from a method or block of code. You can use it to throw both checked and unchecked exceptions. For example: `throw new Exception ("Error occurred");`.

throws is used in a method declaration to **declare** that a method might throw certain exceptions. It tells the caller to handle the exception. For example:

`public void myMethod() throws IOException,SQLException { }`.

31. What is the difference between primitive and non-primitive datatypes?

Primitive data types are the basic types provided by Java. They store actual values and are not objects. They have a fixed size and are more efficient in memory and performance. Examples include `int`, `char`, `boolean`, and `double`.

Non-primitive data types are objects or classes created by the programmer. They store references to memory locations, not the actual data. They can have methods and be more complex. Examples include `String`, arrays, `ArrayList`, and custom classes.