**JAVA VIVA QUESTIONS**

**Java Basics**

* **Object Class and Methods:** The Object class is the root class for all Java classes. Key methods: toString(), equals(), hashCode(), getClass(), clone(), notify(), notifyAll(), and wait().

**Exception Handling**

* **throw & throws:** throw is used to throw an exception manually. throws is used to declare exceptions a method can throw.
* **Checked vs Unchecked Exceptions:**
  + **Checked:** Exceptions that are checked at compile-time (e.g., IOException).
  + **Unchecked:** Exceptions that occur at runtime (e.g., NullPointerException).
* **Exception Propagation:** The process of passing an exception from a method to the calling method using throws.

**JDK 8 Features**

* **Streams, Lambdas, Optional, New Date-Time API**, default methods in interfaces, etc.

**Collections API**

* Includes classes like List, Set, Map, etc. Common methods: add(), remove(), size(), get(), put(), etc.

**Multithreading**

* **sleep()**: Pauses the thread for a specified time.
* **wait()**: Causes the current thread to wait until notified.
* **join()**: Ensures that one thread waits for another to finish.
* **Synchronization**: Ensures that only one thread accesses a block of code at a time.
* **Thread Life Cycle**: New, Runnable, Blocked, Waiting, Timed Waiting, Terminated.

**Functional Interface and Lambda Expression**

* **Functional Interface**: An interface with a single abstract method (e.g., Runnable, Callable).
* **Lambda Expression**: A short block of code that takes in parameters and returns a value. Example: (x, y) -> x + y.

**Serialization**

* **Serialization**: The process of converting an object into a byte stream to store or transmit it.

**Upcasting, Downcasting, Loose Coupling**

* **Upcasting**: Casting a subclass object to a superclass type.
* **Downcasting**: Casting a superclass object to a subclass type.
* **Loose Coupling**: Reducing dependency between components.

**Bean Life Cycle (Spring)**

* **Spring Bean Life Cycle**: Bean instantiation, initialization, and destruction methods.

**Spring/IOC Container**

* **IOC Container**: Manages beans and their dependencies using Dependency Injection.

**Autowiring and Dependency Injection**

* **Autowiring**: Automatic injection of dependencies by Spring.
  + **Ways**: @Autowired, @Inject, by type, by name, constructor, and setter.
  + **Default Autowire**: By type.

**JPA (Java Persistence API)**

* **@Entity, @Table, @Id**: Used to define an entity, map it to a table, and specify the primary key.
* **JPA Relationships**: @OneToMany, @ManyToOne, @ManyToMany, @OneToOne.
* **JPQL vs SQL**: JPQL is object-oriented, using entity classes, while SQL is database-specific.

**Spring Annotations**

* @Autowired, @Qualifier, @Controller, @Service, @Repository, @Component: Define Spring beans and DI configurations.

**DispatcherServlet (Spring MVC)**

* **DispatcherServlet**: The front controller that handles HTTP requests and maps them to appropriate handlers.
* **Lifecycle**: Initialization, request handling, response generation, destruction.

**Servlet Lifecycle**

* **Lifecycle**: Init, service, destroy.

**JSP Lifecycle**

* **Lifecycle**: Translation, Compilation, Loading, Initialization, Execution, Destruction.

**Inversion of Control (IoC)**

* **IoC**: A design principle where control of object creation and dependency management is transferred to a container (e.g., Spring).

**JPA vs Hibernate vs Spring Data JPA**

* **JPA**: Standard specification for ORM.
* **Hibernate**: A popular JPA implementation.
* **Spring Data JPA**: A Spring project that simplifies JPA-based data access.

**Entity Manager Factory and Entity Manager**

* **EntityManagerFactory**: Creates EntityManager instances.
* **EntityManager**: Manages entities and interacts with the database.

**@Transactional in Spring**

* **@Transactional**: Used to define transaction boundaries (commit/rollback) for database operations.

**Demo Flow**

* **Generalized Exception Handler**: Typically in a @ControllerAdvice class.
* **DispatcherServlet Configuration**: Defined in web.xml or Spring Boot configuration file.

**Spring Framework Key Concepts**

* **IOC Container**: Manages beans and their lifecycle.
* **Dependency Injection**: The process of injecting dependent objects into a class.

**JDBC vs JPA (Hibernate)**

* **JDBC**: Low-level database interaction, manually handling SQL.
* **JPA**: High-level ORM framework, abstracting SQL operations using annotations and entities.

**Multithreading**

* **Synchronization**: Ensures only one thread accesses a block of code at a time.
* **sleep() vs wait()**: sleep() pauses the current thread; wait() causes the thread to wait until notified.