**Java 8 1**

|  |  |
| --- | --- |
| **Intermediate**     * close() * distinct() * filter() * findAny() * findFirst() * flatMap() * limit() * map() * mapToInt() * mapToLong() * mapToDouble() * parallel() * peek() * skip() * sorted() | **Terminal**   * allMatch() * anyMatch() * collect() * count() * forEach() * max() * min() * noneMatch() * reduce() |
| **Comparator**  (compares two different objects)   * Comparing(t1,t2) * comparingInt(x) * comparingLong(x) * comparatingDouble(x) * naturalOrder() * nullsFirst() * nullsLast() * reversedorder() * compare() * reversed() * thenComparing() * thenComparingInt() * thenComparingLong() * thenCompartingDouble() | **Collectors**   * averaging() * collectors.toList() * collectors.toSet() * collectors.toMap() * counting() * grouping() * joining() * summarizing() * comparable   (compares itself to another object)   * compareTo() |
| **Lamda /Funcational Interfaces**   1. **Predicate : (I)**   boolean test (T t);  and();  isEqual();  negate();  or();   1. **Function: (I)**   R apply(T t);  andThen();  compose();  identify(); | 1. **Consumer : (I)**   void accept(T t);  andThen();     1. **Supplier : (I)**   T get(); |

**String java.lang.Object 2**

|  |  |
| --- | --- |
| * charAt() * codePointAt() * before() * count() * offsetByteCodePoint() * compareTo() * compareToIgnoreCase() * concat() * contains() * contentEqualls() * copyValueOf() * startsWith() * endsWith() * equals() * equallsIgnoreCase() * getBytes() * getChars() * indexOf() * lastIndexOf() * intern() * isEmpty() * length() * matches(regex) * regionalMatch() * replace() * replaceAll() * replaceFirst() * split() * subsequence() * substring() * toCharArray() * toLowerCase() * toUpperCase() * trim() * valueOf() | * Basic Methods   + - toString()     - equals()     - hashCode()     - getClass() * Object Life Cycle Methods * Clone() * Finalize() * Thread Methods * Wait() * Notify() * notifyAll()   **Java.Util.Object**     * isNull() * nonNull() * requireNonNull() * deepEqualls() * compare() * requiresNonNullElse()   **Junit-5**   * @ConvertWith * @ValueSource * @MethodSource * @CSVSource * @CSVFileSource |

**Junit-5**

|  |  |
| --- | --- |
| * @Test * @ParameterizedTest * @ReportedTest * @TestFactory * @TestTemplate * @BeforeEach * @AfterEach * @BeforeAll * @AfterAll * @DisplayName * @DisplayNameGenerator | * @Disabled * @TimeOut * @ExtendWith * @ProgramaticExtension * @Tag * @Nested * @BeforeAllCallback * @AfterCallBack * @Container * @DynamicTest * @TestInstance |

**Mockito Rest-Assured 3**

|  |  |
| --- | --- |
| @Mock  @Spy  @Captor  @InjectMocks  @RunWith  @InOrder  @MockitoSettings  @Ignore  @MockStatic  @GlobalSetup  @GlobalTearDown  @ArgumentCaptor | @BaseURI  @BasePath  @RelaxedHttp  @Given  @When  @Then  @And  @Header @Headers  @Param @Params  @Cookie @Cookies  @MultiPart @MultiParts  @FormParam @FormParams  @Body  @Extract  @Matches  @MockMVCRequiredBuilder  @LogDetails  @UnMarshalling  @Name |

**Design Patterns**

|  |  |
| --- | --- |
| **Creational**   * SingleTon * Prototype * Builder * Factory * Abstract Factory | **Structural**   * Adapter * Bridge * Composite * Decarator * Façade * FlyWeight * Proxy |
| **Behavioral**   * State * Visitor * Template * Iterator * Interpreter * Observer * Command * Memento * Mediator * Chain of Responsibility * Strategy |  |

**Collections methods 4**

|  |  |
| --- | --- |
| **Java.util.Arrays** | **Java.util.ArrayList** |
| **Sorting**   * Sort() * parallelSort() * sort(Comparator)   **Searching**   * binarySearch() * binarySerach(Comparator)   **Comparision**   * equals() * deepEqualls()   **Filling**   * fill() * setAll()   **Copying**   * copyOf() * copyOfRange()   **Conversion**   * toString() * asList() * hashCode() * Arrays.Stream() * splitIterator() | **Adding**   * add() * add(index, element) * addAll(collection)   **Accessing**   * get(Index) * set(Index, element) * size() * isEmpty() * contains() * indexOf() * lastIndexOf()   **Remove**   * remove() * clear()   **SubList**   * subList(fromIndex, toIndex)   **Iterator**   * iterator() * forEach() * toArray() * sort |

**Java.util.hashSet**

|  |  |
| --- | --- |
| **Adding**   * add() * addAll()   **Remove**   * remove() * removeAll() * clear()   **Checking**   * contains() * containsAll() * isEmpty() | **Set**   * retainAll()   **Size and Iterator**   * size() * iterator() * forEach() * toArray() |

**Java.util.HashMap java.util.concurrentHashMap 5**

|  |  |
| --- | --- |
| **Adding & Updating**   * put(Key, value) * putIfAbsent(Key, value) * putAll()   **Retrieving**   * get() * containsKey() * containsValue() * getOrDefault(Key, defaultValue)   **Removing**   * remove(key) * remove(Key, value) * clear()   **Size and Clear**   * size() * isEmpty()   **Iterating and views**   * keyset() * values() * entrySet() * forEach(Key, values) | **Adding and Updating**   * put() * putAll() * putIfAbsent()   **Retrieving**   * get() * getOrDefault() * isEmpty() * keys() * contains() * containsKey() * containsValues() * size()   **Calculations**   * compute() * computeIfAbsent() * computeIfPresent() * reduceEntries() * reduceKeys() * reduceValues() * searchEntries() * searchValues() * reduceToInt() * mappingCount()   **Delete**   * remove() * clear()   **Others and Iterations**   * elements() * entrySet() * keyset() * values() * forEach() * merge() * replace() * replaceAll() |

**SQL** mnemonics : GHO is like in between limited distinct **6**

|  |  |  |
| --- | --- | --- |
| **DDL**   * Create * Alter * Drop * Truncate * Rename   **DML**   * Select * Insert * Update * Delete * Merge   **DCL**   * Grant * Revoke   **Transmission**   * Commit * Rollback * SavePoint * SetTransaction   **Query**   * From * Where * GroupBy * Having * OrderBy * Distinct   **Aggregate Functions**   * Average * Count * Sum * Max * Min | **String Functions**   * Replace * Concate * Substring * Instring * Trim * Upper * Lower * Length   **Date & Time Functions**   * SysDate * CurrentDate * CurrentTimestamp * Extract * To\_Date * To\_Char   **Numeric Functions**   * Round * Trunc * Ceil * Floor * Abs * Mod   **Conditional Expressions**   * Case * Decode * NVL * Nullif * Coalsec   **Join Types**   * Inner Join * Left Join * Right Join * Full Join * Cross Join * Natural Join | **SubQuery keywords**   * In * Exists * Any * All * Not In * Some * Not Exits   **SET Functions**   * Union * Intersect * Minus   **SQL Objects**   * Tables * Views * Indexes * Sequences * Synonymous * Procedures * Functions * Triggers * Materialized View * Constraints * Index Organized * Clusters   **Index & Tables**   * Index * Tables * Primary Key * Foreign Key * Constraints * Unique * Default |

**PL/SQL 7**

|  |  |
| --- | --- |
| **Conditional Statements**   * IF ... THEN ... ELSE ... END IF; * CASE ... WHEN ... THEN ... ELSE ... END CASE;   **Loops**   * LOOP ... END LOOP; * WHILE ... LOOP ... END LOOP; * FOR ... LOOP ... END LOOP;   **Exit and Continue**   * EXIT WHEN <condition>; * CONTINUE WHEN <condition>;   **Scalar Types Data Types**   * NUMBER * CHAR * VARCHAR2 * DATE * BOOLEAN   **Composite Data Types**  Records   * TYPE ... IS RECORD (...);   Collections   * Nested Tables: TABLE OF * Varrays: VARRAY(...) OF * Associative Arrays:   INDEX BY BINARY\_INTEGER  **LOBs (Large Objects)**   * BLOB * CLOB * NCLOB * BFILE   **Exception Handling**   * Predefined Exceptions * NO\_DATA\_FOUND * TOO\_MANY\_ROWS * VALUE\_ERROR * ZERO\_DIVIDE | * User-Defined Exceptions * DECLARE ... EXCEPTION; * RAISE <exception\_name>; * Exception Handling Blocks   BEGIN ... EXCEPTION ... WHEN <exception\_name> THEN ... END;  **Cursors**   * **Implicit Cursors**: Automatically created for single SQL statements. * **Explicit Cursors**   + CURSOR cursor\_name IS <query>;   + OPEN cursor\_name;   + FETCH cursor\_name INTO ...;   + CLOSE cursor\_name; * **Cursor Attributes**   + %FOUND   + %NOTFOUND   + %ROWCOUNT   + %ISOPEN * **Cursor FOR Loop**   + FOR record IN cursor\_name LOOP ... END LOOP;   **Built-In Functions**   * **String Functions**   + UPPER,   + LOWER,   + INITCAP,   + SUBSTR,   + INSTR,   + REPLACE,   + CONCAT * **Numeric Functions**   + ROUND   + TRUNC   + MOD   + ABS   + CEIL   + FLOOR   + POWER |

**8**

|  |  |
| --- | --- |
| * **Date Functions**   + SYSDATE   + CURRENT\_DATE   + ADD\_MONTHS,   + MONTHS\_BETWEEN   + LAST\_DAY   + NEXT\_DAY   + EXTRACT * **Conversion Functions**   + TO\_CHAR   + TO\_DATE   + TO\_NUMBER * **Other Utility Functions**   + NVL   + COALESCE   + DECODE   + LEAST,   + GREATEST   **Transaction Control**   * **Savepoints and Rollbacks**   + SAVEPOINT <savepoint\_name>;   + ROLLBACK TO <savepoint\_name>; * **Commit and Rollback**   + COMMIT;   + ROLLBACK;   **Dynamic SQL**   * **Execute Immediate**   + EXECUTE IMMEDIATE <SQL statement>; * **DBMS\_SQL Package**   + DBMS\_SQL.PARSE   + DBMS\_SQL.BIND\_VARIABLE   + DBMS\_SQL.EXECUTE   **Procedures and Functions**   * **Procedure Syntax**   + CREATE OR REPLACE PROCEDURE procedure\_name IS ... BEGIN ... END; * **Function Syntax**   + CREATE OR REPLACE FUNCTION function\_name RETURN return\_type IS ... BEGIN ... RETURN <value>; END; | **Packages**   * **Defining a Package**   + CREATE OR REPLACE PACKAGE package\_name IS ... END;   + CREATE OR REPLACE PACKAGE BODY package\_name IS ... END;   **Triggers**   * **Creating Triggers**   + CREATE OR REPLACE TRIGGER trigger\_name BEFORE/AFTER INSERT/UPDATE/DELETE ON table\_name FOR EACH ROW BEGIN ... END;   **Sequences**   * **Creating and Using Sequences**   + CREATE SEQUENCE sequence\_name START WITH n INCREMENT BY n;   + **Next Value**: sequence\_name.NEXTVAL   **DBMS Packages**   * **DBMS\_OUTPUT**   + DBMS\_OUTPUT.PUT\_LINE(<message>); * **DBMS\_JOB and DBMS\_SCHEDULER**   + Scheduling jobs with DBMS\_JOB.SUBMIT   + DBMS\_SCHEDULER.CREATE\_JOB   **DBMS\_LOCK**: For managing locks  **DBMS\_UTILITY Functions and Descriptions**   1. **GET\_PARAMETER\_VALUE** - Retrieves the value of a specified initialization parameter. 2. **FORMAT\_CALL\_STACK** - Returns the call stack as a formatted string, helpful for debugging and tracking call sequences. 3. **FORMAT\_ERROR\_STACK** - Returns the error stack as a formatted string, useful in exception handling to display error details. 4. **FORMAT\_ERROR\_BACKTRACE** - Provides the backtrace information from the point where an exception was raised. 5. **COMPILE\_SCHEMA** - Compiles all PL/SQL objects within a specified schema, useful after schema modifications. 6. **NAME\_RESOLVE** - Resolves the name of a database object, providing information such as schema, object type, and ID. 7. **GET\_DEPENDENCY** - Retrieves dependency information for a database object, assisting in understanding object relationships. 8. **GET\_TIME** - Returns the current time in 100th of a second, often used to measure elapsed time intervals. 9. **EXECUTE\_DDL\_STATEMENT** - Executes a DDL (Data Definition Language) statement dynamically, allowing for dynamic DDL operations like creating or dropping objects. 10. **ANALYZE\_PART\_OBJECT** - Analyzes statistics for a specific part of an object (e.g., a partitioned table), aiding in performance optimization. |

**9**

**Oracle SQL Performance Tuning Tools and Features**

|  |
| --- |
| **EXPLAIN PLAN** - Generates a query execution plan, showing how Oracle processes a SQL statement. This helps identify performance bottlenecks.  **AUTOTRACE** - Provides query execution statistics and the execution plan for SQL statements, useful for real-time performance insights.  **DBMS\_XPLAN** - A package that formats and displays execution plans from the cursor cache or from a stored plan table, offering a clear view of query execution paths.  **SQL TRACE** - Enables tracing for SQL statements at session or instance level, capturing detailed execution information for analysis.  **TKPROF** - A utility that formats trace files generated by SQL Trace, allowing you to analyze performance metrics like CPU and I/O usage for each statement.  **AWR (Automatic Workload Repository)** - Collects, processes, and maintains performance statistics, providing snapshots and reports for historical analysis.  **ASH (Active Session History)** - Part of the AWR, captures session activity every second, providing near real-time insight into performance bottlenecks.  **Optimizer Hints** - Directives embedded in SQL queries to influence the optimizer's decision-making, such as USE\_NL, USE\_HASH, and INDEX.  **DBMS\_STATS** - Gathers and manages optimizer statistics for tables, indexes, and schemas, ensuring accurate and up-to-date statistics for query optimization.  **SQL Profiles** - Optimization solutions that store alternate execution plans for queries, helping the optimizer make better decisions.  **SQL Baselines** - Stores and enforces specific execution plans for queries, used to stabilize performance by avoiding plan changes.  **Partitioning** - Divides large tables and indexes into smaller, more manageable pieces, improving query performance and manageability.  **Materialized Views** - Pre-computed views that store query results, used to optimize complex queries by avoiding repeated calculations.  **Indexing** - Speeds up data retrieval, with types like B-tree, Bitmap, Function-based, and Unique indexes available for various use cases.  **10**  **Parallel Query** - Executes SQL statements in parallel, splitting workloads across multiple CPUs to improve performance on large datasets.  **Dynamic Sampling** - Allows the optimizer to gather additional statistics at runtime for complex queries, used when table statistics are unavailable or stale.  **In-Memory Column Store** - Stores tables in a compressed columnar format in memory, optimizing analytic queries for performance.  **Adaptive Query Optimization** - Allows the optimizer to make real-time decisions during execution, adapting plans based on runtime conditions.  **Automatic Indexing** - Automatically creates, evaluates, and maintains indexes based on usage patterns to optimize queries without manual intervention.  **Histograms** - Collects frequency distribution data on column values, providing the optimizer with more detailed information for better cardinality estimates. |

**Oracle SQL Reporting Tools and Features**

|  |
| --- |
| **DBMS\_OUTPUT.PUT\_LINE** - Outputs messages from PL/SQL blocks, commonly used for debugging and reporting intermediate values.  **SET SERVEROUTPUT ON** - Enables display of DBMS\_OUTPUT messages in SQL\*Plus, helping to view PL/SQL execution results.  **Oracle Reports** - A robust reporting tool used to create complex reports, which can include charts, graphs, and data from multiple sources.  **Oracle BI Publisher** - A reporting tool that enables users to create, manage, and deliver documents and reports using templates and data models.  **SQL\*Plus** - Command-line interface for running SQL commands and PL/SQL blocks, used to format query results into basic reports.  **Oracle Application Express (APEX)** - Web-based platform for building and deploying applications, often used to create interactive reports and dashboards.  **Analytic Functions** - Functions like RANK, DENSE\_RANK, ROW\_NUMBER, LEAD, and LAG used in reports to provide advanced analytics on data.  **GROUP BY and HAVING Clauses** - SQL clauses that group rows based on columns and apply conditions, used in generating summary reports.  **ROLLUP and CUBE** - Extensions to the GROUP BY clause that generate subtotals and grand totals, useful for multi-level reporting.  **PIVOT and UNPIVOT** - Transforms row data into columns and vice versa, used in reporting to reshape data for different perspectives. **11**  **Oracle Discoverer** - A tool for creating ad hoc queries and reports, allowing users to explore and analyze data interactively.  **Flashback Query** - Allows viewing of historical data by querying a previous version of the table, useful for point-in-time reporting.  **Hierarchical Queries** - Using CONNECT BY and START WITH clauses to represent parent-child relationships, helpful in generating hierarchical reports.  **Oracle Data Visualization** - Provides drag-and-drop visualizations and analytics, offering an interactive way to create reports with charts and graphs.  **JSON and XML Functions** - Functions like JSON\_TABLE, XMLTABLE, and EXTRACTVALUE allow processing and reporting on JSON and XML data |

**Spring Boot Security**

|  |  |
| --- | --- |
| **Starter**  Spring-boot-starter-security  **Configuration**   * Class : WebSecurityConfigurationAdapter * @EnableWebSecurity * @EnableGlobalMethodSecurity   **User Authentication & Authorization**   * AuthenticationProvider * PasswordEncoding * UserRoles And Authorities * Access Control Configuration   **Authenticate Mechanism**   * Form-Based Authentication “ Login Page” * Http Based Authentication * JWT-Authentication * JWT-Tokens for Rest Endpoints   **Session Management**   * Session Fixation Protection * Session Timeout | **Security Headers**   * Content Security Policy (CSP) * Http Strict Transport Security (HSTS)   **Logout Configuration**   * Logout Configuration   **Customization and Extension**   * Custom Files * Authentication Success and Failure Handless   **Testing**   * Integration Testing   **Monitoring and Logging**   * Security Events   **CSRF Protection**   * CSRF Token |

**12**

**Micro-Services Design Patterns**

|  |  |
| --- | --- |
| **Decomposition Patterns**   * By Business Capabilities   (అసలు మనం services విడిగా రాయడానికి గల కారణాలు)   * By Sub Domain (కొన్ని సార్లు కోసం చేస్తాం) * Strangler Pattern   (monolithic application to microservice conversion)   * Side Car Pattern/Service Mesh   (its microservice application but deploys as monolithic application)   * BulkHead Pattern   (if existing functionality looks heavy then we are going to write micro services to overcome existing problem)   * By Transaction   **Database Pattern**   * Database Per service * Shared Database * CQRS * SAGA * Event Sourcing   **Communication among services**   * Synchronous (Rest Template) * Asynchronous /Message Based * Communication Medium   **Integration Pattern**   * API gateway (Zuul Proxy) * Aggregation Pattern   1.Branch Pattern  2.Chained Pattern   * Client-Side UI Composition Patterns | **Deployment Pattern**   * Multiple services per Instance per host * Services instance per host * Service Instance per VM (virtual Machine) * Service Instance per Container * ServerLess * Blue Green * Canary   **Observability Pattern**   * Log Aggregator * Performance Metrics * Distributed Tracing * Health Check   **Cross Cutting Patterns**   * External Configuration * Service Discovery pattern * Circuit breaker Pattern   **Distributed Transactions**  **Distribution Logs Tracing**  **SOLID Principles**   * **Single Responsibility** * **Open Close Principle** * **Liskov substitution** * **Interface Segregation** * **Dependency inverse**   **Garbage Collection**   * Serial Collector * Parallel Collectors /Throughput Collector * Concurrent Mark Sweep Collector (CMS) * G1-Garbage Collector   **Class Loaders**   * Bootstrap Class Loaders * Extension Class Loaders * Application Class Loaders |

**13**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Runtime Area**   * Method Area * Heap Area * Stack Area   -Threads   * PC Registers * Native Method Area   **Execution Engine**   * Interpreters * JIT Compilers * Garbage Collectors * Native method Interface * Native method libraries   **Java 8 Memory Model**   * Heap   -Eden Space  -Survivor Space  -Old Generation Space   * Non-Heap   Meta Space  **Other Memories**   * Code Cache * Thread Stack * Direct Memory * C-Heap   **Meta Space**   * Class Metadata * Strings * Variables   **Singleton Class**   * Single object for entire application * And only once we can create no modifications.  |  |  | | --- | --- | | **C** | **Create -> post** | | **R** | **Read -> Get** | | **U** | **Update -> Put** | | **D** | **Delete -> Delete** |   \* GET \* HEAD  \* POST \* OPTIONS  \* PUT \* TRACE  \* DELETE \* CONNECT  \* PATCH | **Immutable Class** (no modifications after creating)   * No Setters * All Arguments Constructor * Class Final * Deep Clone * Private Fields * Getters only   **java.util.concurrent** geekforgeeks refer   * Executor * ExecutorService * ScheduledExecutorService * Future * CountDownLatch * CycleBarrier * Semaphore * ThreadFactory * BlockingQueue * DelayQueue * Lock * Phaser  |  |  | | --- | --- | | **Information (100-199)**  100 -continue  101 -Switching Protocols  102-Processing  103-Early Hints | **Redirection (300)**  300 -Multiple choices  301 -Moved permanently  302 -Found  303 -see other  304 – Not Modified  307 – Temporary Redirected  308 – Permanently Redirected | | **Success (200-299)**  200 -OK  201 -Created  202 -Accepted  203 -Non authorize Information  204- no content  205 – Reset content  206-Partial Content  207 -Multi status  208 -Already reported  226 – IM used | **Client (400)**  400-Bad Request  401- Un-authorized  402- Payment Required  403 – Forbidden  404 -Not Found  405 -Method Not Allowed  406 -Not Accepted  407- Proxy Authentication Required | | **Server (500)**  500 -Internal Server Error  501 -Not Implemented  502 -Bad Gateway  503 -Service Un-available  504- Gateway Timeout  505- Http version Supported  507-Insuffucuent storage  508 – Not Extended  511- Network Authentication required | 408-Request Timeout  409-Conflict  410-gone  411-Length Required  412- Precondition failed  413-Payload too large  414- URI too long  415-unsupported Media type  416- Range not satisfied  417 -Exception Failed  418-I am teapot  421-Misdirected request  423 – Locked  429 - too many request | |

**Linux 14**

|  |  |  |  |
| --- | --- | --- | --- |
| **CD** | Change Directory | **Sed** | Stream Editor for Filtering and  Transforming Text |
| **LS** | List Directory Content | **Awk** | Text Processing Tool for Pattern  Scanning and Processing |
| **Pwd** | Print working Directory | **Vim/Nano** | Text Editor For Editing Files |
| **mkDir** | Make Directory | **Echo** | Display the Text |
| **rmDir** | Remove Files or Directories | **Less** | Display File content one page at a Time |
| **cp** | Copy Files or Directories | **Diff** | Compare Files Line by Line |
| **mv** | Move or Rename Files or Directories | **Sort** | Sort the Lines of Text Files |
| **touch** | Create Empty Files or Update File Timestamp | **Unique** | Report or Filter out repeated lines in File |
| **cat** | Concatenate and Display File Content | **Wc** | Count Lines , Words, and Characters in File |
| **grep** | Search for Patterns in Files | **Uptime** | Display System uptime and Load |
| **chmod** | Change File Permissions | **IpConfig** | Display and Configure network  Inferface |
| **chown** | Change File Ownership | **Route** | Display or Modify the ip routing Table |
| **tar** | Manipulate Tar Achieve | **Ping** | Send ICMP echo request to network host |
| **gzip** | Compress or Decompose Files | **TraceRoute/**  **TracePath** | Print the Route Package take to network host |
| **unzip** | Extract Files From Zip Archives | **NetState** | Display network connections,  Routing Tables and Interface statics |
| **du** | Display Disk Usage of Files and Directories | **Date** | Display or Set the System date and Time |
| **df** | Display disk Space usage | **Cal** | Display a Calander |
| **find** | Search Files and Directories | **Shutdown** | Shutdown System |
| **ssh** | Secure Shell for Remote Login | **-f : Force**  **-i : Interactive**  **-n : No-Clobber**  **-s : Suffix**  **-t : Target**  **-T : No-Target-Directory**  **-u : Update**  **-v : Verbose**  **-z : Context** | |
| **scp** | Secure Copy for Secure File Transfer |
| **rsync** | Remove File Synchronization |
| **wget** | Download Files from Internet |
| **curl** | Transfer Data with URLs |
| **top** | Display System Process |
| **PS** | Display information about running Process |
| **Kill** | Terminate Process |
| **Man** | Display manual Pages for Commands |
| **History** | Display command History |
| **Tail** | Display The Last part of Files |
| **Head** | Display the First part of the Files |

**Docker 15**

* **Image Management**

|  |  |
| --- | --- |
| **Docker Pull** | Downloads an Image from a repository/Registry (Docker Hub) |
| **Docker Image** | List All Images Available Locally |
| **Docker Search** | Search for Images on Docker Hub |
| **Docker Build** | Creates a new Image from Docker File |
| **Docker Push** | Uploads a Local Image to Repository/Registry |
| **Docker Tag** | Tags an existing Image with a new Name |
| **Docker rmi** | Removes an Image |
| **Docker History** | Shows the History for an Image |

* **Container Management**

|  |  |
| --- | --- |
| **Docker Run** | Creates and Starts a Container from an Image |
| **Docker PS** | Lists all running containers |
| **Docker PS -a** | Lists all containers including stopped ones |
| **Docker Stop** | Stops a running Container |
| **Docker Start** | Starts a stopped Container |
| **Docker re-start** | Restarts a running Container |
| **Docker rm** | Removes a Container |
| **Docker exec** | Execute Command- Executes a command in a Running container |
| **Docker Attach** | Attaches a running container to shell |
| **Docker Logs** | Shows the logs generated by a container |
| **Docker Commit** | Creates a new Image from a Container changes |
| **Docker CP** | Copy Files between container and host |
| **Docker Diff** | Changes shows in container File system |

* **Network Management**

|  |  |
| --- | --- |
| **Docker Network Create** | Creates a new Custom Network |
| **Docker Network List** | Lists all Available networks |
| **Docker Network Inspect** | Shows Details information about a network |
| **Docker Network Connect** | Connects a Container to a Network |
| **Docker Network Disconnect** | Disconnects a Container from a Network |

* **Volume Management**

|  |  |
| --- | --- |
| **Docker Volume Create** | Creates a new Volume |
| **Docker Volume LS** | Lists all Volumes |
| **Docker Volume Inspect** | Shows Detailed Information about volume |
| **Docker Volume PRUNE** | Removes unused Volumes |
| **Docker Volume mount** | Mounts a Volume to a Container |

* **Docker Version**
* **Docker Info**
* **Docker help**
* **Docker Build -t <name> :** Builds an Image From the Current Directory with a specific names
* **Docker run -d :** runs a container detached mode <background>
* **Docker run -p <host-port> :** <container-port> Maps a container port to a host port **16**
* **Docker run –name <name> :** assigns a custom name for a container
* **Docker CP :** Copies Files between Container & host
* **Docker diff :** Shows the changes made to a Container File System.

**Docker Compose File**

* **Services**

|  |  |
| --- | --- |
| **Version** | Specifies the Docker Compose File Format version |
| **Services** | Defines The services that makes up your application |

* **Service Configuration**

|  |  |
| --- | --- |
| **Image** | Specifies The Docker Image to use for the Service. |
| **Build** | Path to Docker File used to build the image within the context of the compose file |
| **Ports** | Maps container ports to host ports ex : 80 :8080 |
| **Volumes** | Mounts volumes from the host or creates named volume persist date |
| **Environment** | Defines Environment variable for services |
| **Network** | Connects the Service to specific Docker Networks |
| **Dependent on** | Specifies services, this service relies on and must be started before itself |

* **Resources**

|  |  |
| --- | --- |
| **Restart** | Policy for restarting container |
| **Command** | Overrides the default command executed by container |
| **Entry Point** | Sets the Entry point for the container |
| **Labels** | Assigns metadata labels to services |

* **Advanced Configurations**

|  |  |
| --- | --- |
| **Secrets** | Secure File stored outside of compose file |
| **Configs** | Shared Configuration file across multiple services |
| **Build** | Multi-stage builds for optimized Images |
| **Health Checks** | Configure heath creates to monitor service availability |
| **Logging** | Define Logging drives and options for centralizing log management |

* **Service Discovery**

|  |  |
| --- | --- |
| **Links** | Establish relationship between services allowing them to discover each other  Hostname and port. |

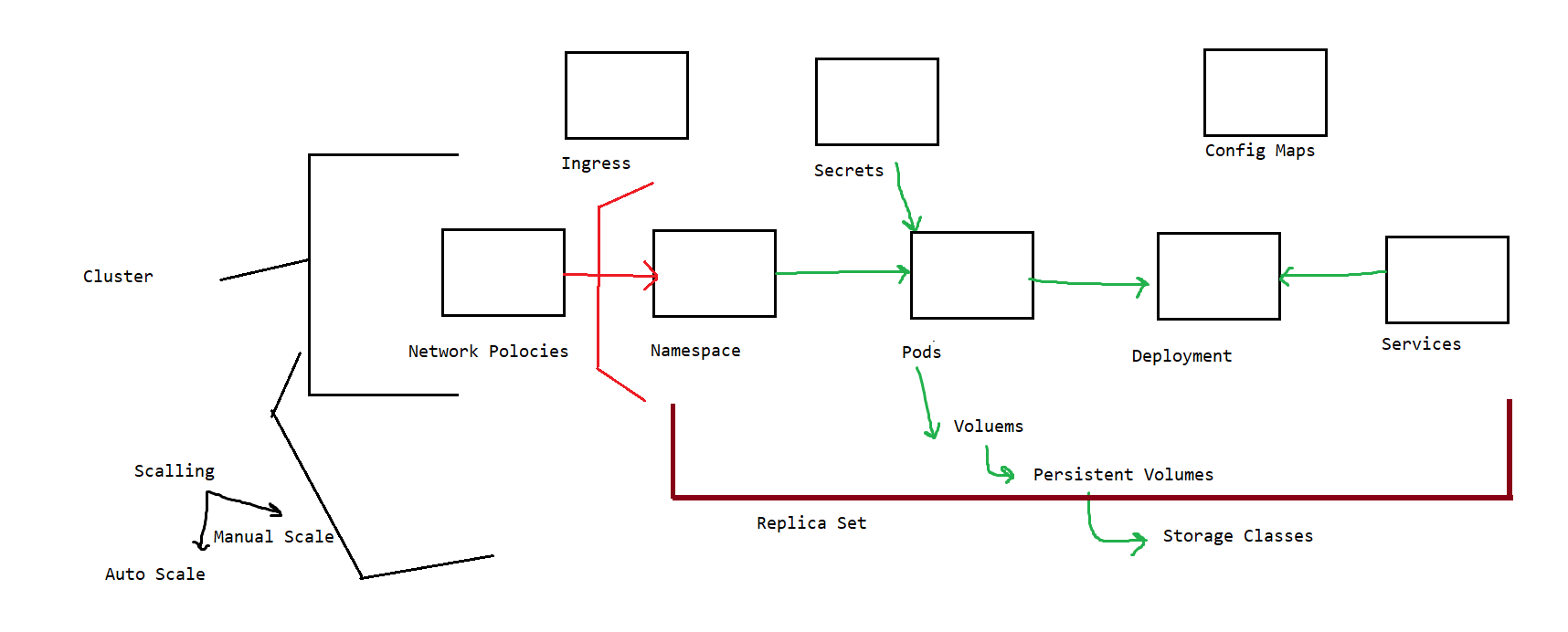
* **Deployment Options**

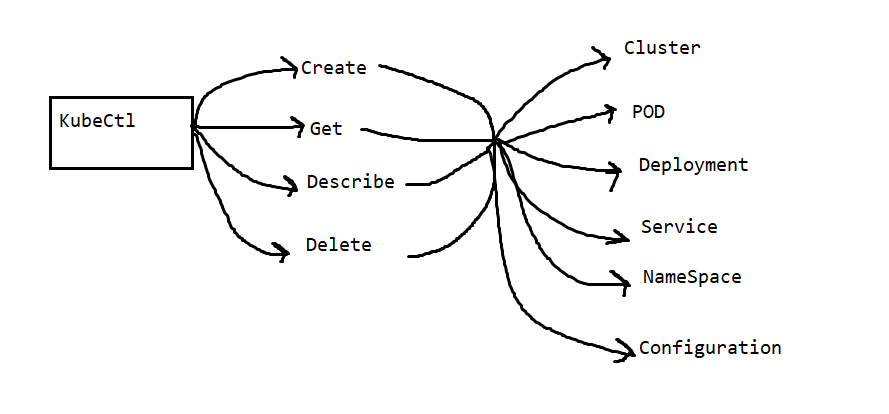
|  |  |
| --- | --- |
| **Docker Compose Up** | Starts all defined services |
| **Docker Compose Down** | Stops and Removes all Containers |
| **Docker Compose Build** | Build images for all Services |
| **Docker Compose Scale** | Scales the number of replicas for a service |
| **Docker -compose Stack** | Manges deployments a Docker Swarn stacks (deprecated) |

**Kubernetes** (KubeCtl -Kubernetes command line interface) **17**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Core Concepts**  Objects   * Pods * Deployments * Services * Replica Sets * Name Space   Storage   * Volumes * Persistent Volumes   Security   * Secrets * ConfigMaps   **API Groups**  Core Resources   * Pods * Services   Applications   * Deployments * ReplicaSets   Networking   * Network Policies * Ingress   Storage   * Persistent Volumes * Storage Classes   Manifest File Sections   * Pods * Deployments * Services * ConfigMaps * Secrets * Ingress   Manifest File Section cum Core Concepts  (short Notes covers all above things)   * Pods * Deployment * Services * ConfigMaps * Secrets * Ingress * Replica Set * NameSpace * Volumes * Persistent Volumes * Storage Classes * Network Policies * Scaling * Cluster | |  |  | | --- | --- | | KubeCtl Create Cluster | Creates a K8s Cluster | | KubeCtl Get Nodes | Lists All nodes in the Cluster | | KubeCtl Descibe Node | Describes a Specific node in Details | | KubeCtl Delete Node | Delets a Node from the cluster |   **Cluster Management**  **POD Management**   |  |  | | --- | --- | | KubeCtl Create POD | Creates a new POD | | KubeCtl Get PODs | List all the PODS in the Cluster | | KubeCtl Describe POD | Describes about the POD | | KubeCtl Delete POD | Deletes the POD from Cluster |   **Deployment Management**   |  |  | | --- | --- | | KubeCtl Create Deployment | Creates a new Deployment | | KubeCtl Get Deployments | Lists all Deployments in the Cluster | | KubeCtl Describe Deployments | Describes a specific Deployment in Details | | KubeCtl Delete Deployments | Deletes the Deployment from the Cluster |   **Service Management**   |  |  | | --- | --- | | KubeCtl Create Service | Creates a new Service | | KubeCtl Get Service | Lists all the services in the cluster | | KubeCtl Descibe Services | Describes the service in the cluster | | KubeCtl Delete Service | Deletes a service from Cluster |   **Namespace Management**   |  |  | | --- | --- | | KubeCtl Create Namespace | Creates a namespace | | KubeCtl Get Namespace | Lists all the namespaces | | KubeCtl Describe Namespace | Describes about namespace | | KubeCtl Delete Namespace | Deletes the namespace |   **Configuration Management**   |  |  | | --- | --- | | KubeCtl Apply -f | Applies a configuration File | | KubeCtl Get ConfigMaps | Lists all configsMaps | | KubeCtl Describe configMaps | Describes about configurations | | KubeCtl Delete ConfigMaps | Deletes a configMaps |   **Scaling**   |  |  | | --- | --- | | KubeCtl Scale Deployment <deployment-name> --replicas = <num-replicas> | Scales a deployment to specified number of replicas | | KubeCtl autoscale deployment <deployment-name> -- | min=<min-replicas> --max =<max-replicas> --cpu-percentage =<cpu-threshould> | |

**18**

****

****

**Jenkins**

|  |  |
| --- | --- |
| **Pipeline Structure**   * Pipeline * Agent * Stages * Stage * Steps * Parallel   **Nodes and Parallel Execution**   * Node   **Error Handling and Flow Control**   * CatchError * CatchErrorBuilderResult * When * Expression * BeforeAgent * AfterAgent | * Retry * Timeout * WaitUntil * WarnError * Input   While(expression){  }CatchError(){  CatchErrorBuilder()  }  **Variables and Parameters**   * Parameters * Parameters Block |

**Environment And Tools 19**

|  |  |
| --- | --- |
| * Environment * Tools * WithCredentials * WithError * WithMaven * WithSonarQubeEnvrnment * WithDockerContainer * WithDockerRegistry * WithDockerServer * WithNode * WithSonarQubeScanner * WithTimeout * WithWorkSpace * DockerFingurePrintForm   **Triggers and Notifications**   * Triggers * EmialText * SlackSend   **Others**   * Library * Error * Load * Lock * MileStone * Options   POST  SUCCESS  FAILURE  UNSTABLE  ABORTED  ALWAYS | **Artifact and Reports**   * UnArchieve * ArchieveArtifact * PublishHTML * PublishBuilderInfo * RecordIssues * GetArtifactServer * FindBuilderScans   **File**   * Clean WS * DeleteDir * ReadFile * WriteFile * ReadTrusted * ReadYml * FileExists   Git & Others   * ChangeLog * Checkout * Stash * Unstash * Validate Declarative Pipeline * Script * Script Approvals * Sleep * Timestamp * GetContext * Build * X-ray scan * Sh (shell) |

**Cucumber BDD :: Gherkin Syntax SetpDefinations Hooks -> Before/After Scenarios /Feature**

|  |  |
| --- | --- |
| **Gherkin**   * Feature * Scenario * Given * When * Then * And * But * Background * Tags | **Annotations**   * @CucumberOptions * @Transpose * @ScenarioScoped * @Features =”FeaturePatch” * @@Glue =”Steps Path” * @Snippet * @Plugin =”Pretty”/html:report.html |

**Feature file 🡪 STEP Defination 🡨 Rest Assured will be used in the STEP definition.**

**HTML 20**

* **Document Structure**

|  |  |
| --- | --- |
| **<html>** | Root element of an HTML document. |
| **<head>** | Contains metadata, links, and scripts. |
| **<title>** | Sets the title of the document. |
| **<base>** | Specifies a base URL for relative URLs. |
| **<body>** | Contains the content of the document. |

* **Metadata**

|  |  |
| --- | --- |
| **<meta>** | Defines metadata, such as character set, viewport settings, and author. |
| **<link>** | Links to external resources like stylesheets. |
| **<style>** | Embeds CSS directly within the HTML. |
| **<script>** | Embeds or links to JavaScript |
| **<noscript>** | Alternative content if JavaScript is disabled. |

* **Text Content**

|  |  |
| --- | --- |
| **<h1> to <h6>** | Headings from largest to smallest |
| **<p>** | Defines a paragraph |
| **<hr>** | Inserts a thematic break (horizontal line). |
| **<pre>** | Preserves whitespace and formatting (preformatted text) |
| **<blockquote>** | Indicates a long quotation. |
| **<b>** | Bold text. |
| **<i>** | Italic text. |
| **<u>** | Underlined text. |
| **<small>** | Smaller text. |
| **<strong>** | Important (bold) text. |
| **<em>** | Emphasized (italic) text. |
| **<mark>** | Highlighted text |
| **<del>** | Deleted (strikethrough) text. |
| **<ins>** | Inserted (underlined) text. |
| **<sup>** | Superscript text. |
| **<sub>** | Subscript text |
| **<br>** | Line break. |
| **<wbr>** | Word break opportunity. |

* **Grouping Content**

|  |  |
| --- | --- |
| **<div>** | Generic container for block content. |
| **<span>** | Generic container for inline content |
| **<figure>** | Groups media elements with optional caption |
| **<figcaption>** | Caption for the <figure> element. |
| **<main>** | Main content of the document |
| **<header>** | Header section. |
| **<footer>** | Footer section. |
| **<section>** | Thematic grouping of content. |
| **<article>** | Independent, self-contained content. |
| **<aside>** | Content related to the main content (e.g., sidebar). |
| **<nav>** | Navigation links section. |
| **<address>** | Contact information. |

* **Lists 21**

|  |  |
| --- | --- |
| **<ul>** | Unordered list. |
| **<ol>** | Ordered list |
| **<li>** | List item. |
| **<dl>** | Description list. |
| **<dt>** | Description term. |
| **<dd>** | Description detail. |

* **Tables**

|  |  |
| --- | --- |
| **<table>** | Table container. |
| **<caption>** | Table caption |
| **<thead>** | Header section of the table. |
| **<tbody>** | Body section of the table |
| **<tfoot>** | Footer section of the table. |
| **<tr>** | Table row. |
| **<tr>** | Table row. |
| **<th>** | Table header cell. |
| **<td>** | Table data cell. |
| **<col>** | Column properties. |
| **<colgroup>** | Group of columns |

* **Forms and Input**

|  |  |
| --- | --- |
| **<form>** | HTML form container |
| **<input>** | Various input fields (text, password, checkbox, etc.). |
| **<textarea>** | Multi-line text input. |
| **<button>** | Clickable button. |
| **<select>** | Dropdown list. |
| **<option>** | Option in a dropdown list. |
| **<optgroup>** | Group of options in a dropdown |
| **<label>** | Label for a form element. |
| **<fieldset>** | Group of related elements. |
| **<legend>** | Caption for <fieldset>. |
| **<legend>** | Caption for <fieldset> |
| **<output>** | Calculation or result output |
| **<datalist>** | Provides a list of options for <input> |
| **<keygen>** | Generates a key-pair (deprecated in HTML5.2) |

* **Interactive Elements**

|  |  |
| --- | --- |
| **<details>** | Expanding/collapsible content. |
| **<summary>** | Summary for <details>. |
| **<dialog>** | Represents a dialog box or window. |
| **<menu>** | Context menu or toolbar (now less commonly used). |

* **Embedded Content 22**

|  |  |
| --- | --- |
| **<img>** | Image. |
| **<iframe>** | Inline frame for embedding content. |
| **<embed>** | Embeds external content. |
| **<object>** | Embeds multimedia or applications. |
| **<param>** | Parameters for <object>. |
| **<video>** | Embeds a video. |
| **<audio>** | Embeds audio content. |
| **<source>** | Media source for <video> and <audio>. |
| **<track>** | Text track for <video> and <audio> (e.g., subtitles). |

**Spring Boot**

* **Core Spring Annotations**

|  |  |
| --- | --- |
| **@SpringBootApplication** | Main entry point for Spring Boot applications, combining |
| **@Configuration** | Marks a class as a source of bean definitions |
| **@EnableAutoConfiguration** |  |
| **@ComponentScan** | Automatically scans for components, configurations, and services in a specified package. |
| **@Bean** | Defines a Spring bean in a configuration class |
| **@Component** | Marks a class as a Spring-managed component. |
| **@Service** | Indicates that a class is a business service layer component. |
| **@Repository** | Indicates that a class is a data repository, with exception translation for persistence errors |
| **@Controller** | Marks a class as a Spring MVC controller. |
| **@RestController** | A specialized version of @Controller that combines @Controller and @ResponseBody. |
| **@RequestMapping** | Maps HTTP requests to handler methods of MVC and REST controllers. |
| **@Autowired** | Automatically injects dependencies by type. |
| **@Qualifier** | Specifies the exact bean to inject when there are multiple candidates. |
| **@Lazy** | Indicates that a bean should be lazily initialized |

* **Spring Boot Configuration Annotations**

|  |  |
| --- | --- |
| **@EnableAutoConfiguration** | Enables automatic configuration of the Spring application based on the classpath. |
| **@PropertySource** | Adds a properties file to the Spring environment. |
| **@Value** | Injects values from properties files |
| **@ConfigurationProperties** | Binds a class to properties with a specific prefix. |
| **@EnableConfigurationProperties** | Enables support for @ConfigurationProperties beans |
| **@Profile** | Defines beans or configurations specific to a particular profile |

* **Web and REST Annotations 23**

|  |  |
| --- | --- |
| **@GetMapping** | Shortcuts for @RequestMapping with specific HTTP methods. |
| **@PostMapping** |
| **@PutMapping** |
| **@DeleteMapping** |
| **@PatchMapping** |
| **@PathVariable** | Binds a method parameter to a URI template variable |
| **@RequestParam** | Binds a method parameter to an HTTP request parameter |
| **@RequestBody** | Binds the body of an HTTP request to a method parameter |
| **@ResponseBody** | Indicates that a method return value should be bound to the web response body |
| **@ResponseStatus** | Specifies the HTTP status code for a method's response |
| **@CrossOrigin** | Configures CORS support for cross-origin requests |

* **Data Access and Persistence Annotations**

|  |  |
| --- | --- |
| **@Entity** | Specifies a class as an entity in JPA |
| **@Table** | Specifies the table associated with an entity. |
| **@Id** | Specifies the primary key of an entity |
| **@GeneratedValue** | Configures automatic generation of primary key values |
| **@Column** | Specifies details of a column in a database table. |
| **@ManyToOne** | Defines various JPA relationships between entities. |
| **@OneToMany** |
| **@OneToOne** |
| **@ManyToMany** |
| **@JoinColumn** | Specifies the joining column for relationships. |
| **@Transactional** | Manages transactional behavior for methods or classes. |
| **@Modifying** | Indicates a modifying query method in Spring Data repositories |
| **@Query** | Defines a custom query in a repository. |

* **Spring Security Annotations**

|  |  |
| --- | --- |
| **@EnableWebSecurity** | Enables Spring Security’s web security support |
| **@Secured** | Defines method-level security with roles |
| **@PreAuthorize** | Method-level security annotation that checks conditions before method execution. |
| **@RolesAllowed** | Specifies roles allowed to access a method |
| **@PermitAll** | Allow or deny access to a specific method. |
| **@DenyAll** |
| **@AuthenticationPrincipal** | Accesses the currently authenticated principal |

* **Validation Annotations (Bean Validation)**

|  |  |
| --- | --- |
| **@NotNull, @Size, @Min, @Max, @Email** | Standard validation constraints |
| **@Valid** | Triggers validation on method parameters |
| **@Pattern** | Specifies a regex pattern for validation |
| **@Positive, @Negative, @NotEmpty** | Additional constraints for validating fields. |

* **Asynchronous Processing Annotations 24**

|  |  |
| --- | --- |
| **@EnableAsync** | Enables asynchronous method execution |
| **@Async** | Marks a method as asynchronous |
| **@Scheduled** | Schedules tasks to be run at a fixed interval. |
| **@EnableScheduling** | Enables Spring’s scheduling support. |

* **Testing Annotations**

|  |  |
| --- | --- |
| **@SpringBootTest** | Sets up an application context for integration testing |
| **@WebMvcTest** | Configures a test with only Spring MVC components. |
| **@DataJpaTest** | Configures a test for JPA components |
| **@MockBean** | Mocks a bean in the application context |
| **@SpyBean** | Spies a bean, allowing partial mocking |
| **@BeforeEach** | Specifies setup and teardown methods for each test |
| **@AfterEach** |

* **Miscellaneous Annotations**

|  |  |
| --- | --- |
| **@EventListener** | Registers a method as an event listener |
| **@ConditionalOnProperty** | Conditionally loads a bean based on a property |
| **@ConditionalOnMissingBean** | Loads a bean only if another bean is absent |
| **@EnableCaching** | Enables caching support. |
| **@Cacheable, @CachePut, @CacheEvict** | Manages caching behaviour. |
| **@Import** | Imports configuration classes. |
| **@Order** | Defines the execution order of components. |
| **@DependsOn** | Specifies dependencies between beans |
| **@Profile** | Activates beans based on the active profile. |

**Data Structures**

**Time Complexity and Space Complexity**

Time Complexity and Space Complexity are measures used to analyse the efficiency of algorithms, focusing on how resource usage grows with input size.

**Time Complexity:** Measures the amount of time an algorithm takes to complete as a function of input size. It's commonly expressed in Big-O notation.

|  |  |
| --- | --- |
| **O(1)** | Constant time, where the time remains the same regardless of input size |
| **O(log n)** | Logarithmic time, often seen in binary search algorithms |
| **O(n)** | Linear time, where time grows directly with input size, common in loops |
| **O(n log n)** | Linearithmic time, typical for efficient sorting algorithms |
| **O(n²)** | Quadratic time, common in nested loops |
| **O(2ⁿ), O(n!)** | Exponential and factorial times, seen in algorithms that try every possibility, like certain recursive algorithms |

**Space Complexity 25**

Measures the amount of memory space an algorithm uses in relation to input size.

|  |  |
| --- | --- |
| **O(1)** | Constant space, where memory usage remains constant regardless of input size |
| **O(n)** | Linear space, where memory usage grows with input size (e.g., storing an array of size n). |
| **O(n²)**, **O(2ⁿ)** | Higher-order space complexities, common in algorithms that use multi-dimensional arrays or complex recursion |

* **Data Structures and Their Subcategories**

**1. Arrays**

* **Static Array** - Fixed size, indexed structure.
* **Dynamic Array (e.g., ArrayList in Java)** - Resizable, provides efficient indexing.
* **Multidimensional Array** - Arrays with more than one dimension, such as matrices.

**2. Linked Lists**

* **Singly Linked List** - Each node points to the next node.
* **Doubly Linked List** - Each node has pointers to both the next and previous nodes.
* **Circular Linked List** - Last node points back to the first, forming a loop.

**3. Stacks**

* **Simple Stack** - Last In, First Out (LIFO) structure.
* **Array-Based Stack** - Stack implemented using arrays.
* **Linked List-Based Stack** - Stack implemented using linked lists.

**4. Queues**

* **Simple Queue** - First In, First Out (FIFO) structure.
* **Circular Queue** - Tail wraps around to the head to utilize empty space.
* **Priority Queue** - Elements are dequeued based on priority, not order.
* **Deque (Double-Ended Queue)** - Allows insertion and removal from both ends.

**5. Hashing**

* **Hash Table** - Key-value pairs with hashing for fast lookups.
* **Hash Map** - Implementation of a hash table, commonly with collision handling.
* **Hash Set** - A set based on hashing for unique elements.

**6. Trees**

* **Binary Tree** - Each node has at most two children.
* **Binary Search Tree (BST)** - Left child < Parent < Right child, for ordered data.
* **AVL Tree** - Self-balancing BST with height balancing.
* **Red-Black Tree** - Balanced BST, used in Java’s TreeMap. **26**
* **Segment Tree** - Tree for range queries.
* **Fenwick Tree (Binary Indexed Tree)** - Efficient for prefix sum queries.
* **Trie (Prefix Tree)** - Stores strings or sequences, common in search applications.

**7. Heaps**

* **Min-Heap** - Parent nodes are less than or equal to their children, minimum at the root.
* **Max-Heap** - Parent nodes are greater than or equal to their children, maximum at the root.
* **Binary Heap** - Implemented as an array, used for priority queues.

**8. Graphs**

* **Undirected Graph** - Graph with bidirectional edges.
* **Directed Graph (Digraph)** - Graph with unidirectional edges.
* **Weighted Graph** - Edges have weights or costs.
* **Unweighted Graph** - All edges are equal.
* **Adjacency Matrix** - 2D array representation.
* **Adjacency List** - List-based representation, efficient for sparse graphs.

**9. Sets**

* **HashSet** - Set with hash-based storage for unique elements.
* **TreeSet** - Ordered set based on a binary search tree.
* **LinkedHashSet** - Set that maintains insertion order.

**10. Maps**

* **HashMap** - Key-value pairs stored with hash-based indexing.
* **TreeMap** - Sorted map based on a red-black tree.
* **LinkedHashMap** - Map that maintains insertion order.

**11. Specialized Data Structures**

* **Bloom Filter** - Probabilistic data structure for checking membership with false positives.
* **Skip List** - List with multiple layers to enable fast searching.
* **Union-Find (Disjoint Set)** - Efficient data structure for disjoint set operations, common in graph problems.
* **LRU Cache (Least Recently Used)** - Cache that discards the least recently accessed items.