Home Assignment

Microimage Mobile Media (Pvt) Ltd

1. **How Hashmaps work internally ?**

It is one of the most popular Collection classes in Java which internally uses HashTable implementation.

HashMap stores map items in its static inner class Node<K,V>. This indicates that each hashMap entry is a Node. HashMap utilizes the hashCode of the key Object internally, and the hash function uses this hashCode to locate the index of the bucket where the new item may be inserted.

HashMap has several buckets, each of which refers to a Singly Linked List, which stores the entries (nodes).

After the hash function has identified the bucket using hashcode, hashCode is used to see if the bucket already contains a key with the same hashCode or not in the bucket.

If a key with the same hashCode already exists, the equals() function is applied to the keys. If the equals function returns true, it implies that a node with the same key already exists, and the value for that key in the entry(node) is overwritten; otherwise, a new node is generated and added to that bucket's Singly Linked List.

If the bucket identified by the hash function does not contain any keys with the same hashCode, the new Node is added to it.

2. **Code conversions in Java**
    i. String to int

To convert String to int, use Integer.parseInt() which returns primitive int in Java.

    ii. String to long

To convert String to long, use Long.parseLong() which returns primitive long in Java.

    iii. String to float

To convert String to float, use Float.parseFloat() which returns primitive float in Java.

    iv. String to Boolean

To convert String to Boolean, use Boolean.parseBoolean () which returns primitive Boolean, either TRUE or FALSE in Java.

    v. String to Date

To convert a value of String into a Date object.

3. Factory Design Pattern (Demonstration)

A Factory Pattern, also known as a Factory Method Pattern, states that you should just define an interface or abstract class for generating objects and leave it up to the subclasses to select which class to instantiate. In other words, subclasses are in charge of creating the class instance.

Java application to demonstrate the Factory Design Pattern (https://github.com/kvdrathnayaka/HomeAssignment_MicroimageMobileMedia/blob/main/factoryPattern.java)

4. Bash Script

A Bash script is a text file with a set of commands in it. A Bash script can contain any command that can be run from the terminal. Any sequence of commands to be performed on the terminal may be written as a Bash script in a text file and executed in that order.

Bash script (https://github.com/kvdrathnayaka/HomeAssignment_MicroimageMobileMedia/blob/main/bashScript.sh) to compile and run the above application.

5. Use of Docker

Docker is an open-source software platform that allows users to develop, deploy, and manage virtualized application containers on a shared operating system using a set of tools. The Docker engine is suitable for lone programmers that want a lightweight, clean environment for testing but do not require sophisticated orchestration.

One underlying operating system is shared by all containers. Containers that migrate across Docker environments with the same OS function without modifications since Docker images contain all the requirements needed to execute code within a container. There can be several usages of Docker, which are,

- Docker can be used as a version control system for your entire app's operating system.
- When you wish to share/collaborate on your app's operating system with a team, use Docker.
- Docker allows you to execute your code in the same environment as your server on your laptop.
- Docker can be used if your program has to go through numerous development phases.

Docker Hub is a software-as-a-service tool that allows users to publish and distribute container-based programs via a centralized location. Trusted Registry, like Hub, is a repository that provides an additional layer of control and ownership over container image storage and delivery.

Docker Engine's swarm mode allows for cluster load balancing. Users may easily scale up container deployments to many hosts by pooling several Docker hosts resources to operate as one.

Compose is a command-line utility for configuring multi-container application services, viewing container statuses, streaming log output, and running single-instance processes.

Content Trust is a security solution that uses user signatures and image metadata to validate the integrity of remote Docker registries.

6. SQL vs NOSQL databases

SQL Databases	NOSQL Databases
Relational databases	Non-relational databases
Have a predefined schema and a structured query language	Have dynamic schemas for unstructured data
Databases are vertically scalable	Databases are horizontally scalable
Databases are table-based	Databases are document, key-value, graph, or wide-column stores
SQL databases are better for multi-row transactions	NoSQL is better for unstructured data like documents or JSON
Best suited for complex queries	Not so good for complex queries
Follow ACID properties (Atomicity, Consistency, Isolation and Durability)	Follows the Brewers CAP theorem (Consistency, Availability and Partition tolerance)

7. Immutable vs mutable

Mutable objects are useful because they allow you to make changes without having to create a new object. However, be aware that if you make an in-place modification to an object, all references to that object will be updated to reflect the change. Example: StringBuilder, java.util.Date.

Immutable refers to a state that cannot be modified after it has been created. When all of an object's fields are immutable, it is said to be immutable. It's the next step after the Unmodifiable object, which is a wrapper for modifiable. It ensures that it can't be changed directly (but it is possibly using backing object). Example: String, Boxed primitive objects like Integer, long etc.

The major benefit of immutable objects is that they are designed to work in a concurrent environment. The most serious issue with concurrency is shared resources that may be modified by any thread. Immutable objects, on the other hand, are read-only, which is a thread-safe process. Any change to an original immutable object results in a copy.

- When utilized for in-place operations, mutable collections are often quicker than immutable collections.
- When there is a performance bottleneck, it is usual to use mutable collections locally within a method or private to a class, but immutable collections elsewhere when speed is less of a concern.

- It's possible to create bugs in which a shared mutable collection is changed unexpectedly, causing you to hunt down which line in a huge codebase is responsible for the undesired update.

8. Horizontal Scaling vs Vertical Scaling

Scaling horizontally and vertically are similar in that they both involve expanding your infrastructure's computer resources. In terms of implementation and performance, there are significant variations between the two. Vertical scaling refers to adding more power to your pool of resources, whereas horizontal scaling refers to adding more units to your pool of resources.

Horizontal scaling requires splitting a sequential piece of logic into smaller chunks so that they may be processed in parallel across several systems, which is one of the key distinctions between the two. Vertical scaling is simpler in many ways since the reasoning does not need to change. Instead, you're simply executing the same code on more powerful computers.

However, there are several additional factors to consider while deciding on the best option.

	Horizontal Scaling	**Vertical Scaling**
Databases	Typically based on data partitioning.	The data is stored on a single node, and scalability is accomplished using multi-core processors.
Downtime	When you add more machines to an existing pool, you're no longer restricted to the capacity of a single unit, allowing you to grow with minimal downtime.	Scaling beyond a single machine's capacity requires downtime and has a hard upper limit.
Concurrency	It's also known as distributed programming since it entails distributing workloads among several workstations across a network.	Multi-threading and in-process message forwarding are frequently used in concurrent programming on multi-core computers.
Message Passing	The lack of a shared address space complicates data sharing in distributed computing. It also increases the cost of sharing, passing, or updating data since copies of the data must be passed.	You may assume the presence of a shared address space in a multi-threaded environment, thus data sharing and message transmission can be accomplished by passing a reference.
Examples	Cassandra, MongoDB, Google Cloud Spanner	MySQL, Amazon RDS

9. How to secure an API endpoint

API endpoints are generally URLs that a server exposes to allow other systems to connect and use its services. API endpoints are points of access into business networks that frequently contain important or sensitive data. As a result, they are a tempting target for attackers.

To increase API endpoint security, minimize your attack surface, and reduce the chance of successful assaults, use the best practices listed below.

- Use API keys to authorize users.

Access to public REST services is controlled via API keys. API keys can be used by public web service providers to rate-limit API calls and prevent denial-of-service(DoS) attacks.

- Ensure that all APIs use HTTPS, even if they appear to be simple.

Your passwords, private keys, and credit card information are easily stolen when you use an API endpoint to communicate over HTTP or any other unprotected protocol. Make HTTPS the only connection method accessible to protect your API.

- To safeguard passwords, use one-way password hashing with robust encryption.

Because all user accounts are at danger in the case of a security compromise, never keep passwords in cleartext. Symmetric encryption techniques should also be avoided. One-way encryption, commonly known as hashing, is the best choice for password security since no one can reverse the password.

- Use rate limitation to prevent unauthorized access and to prevent DoS attacks.

Whatever the volume of requests your API handles, it's a good idea to restrict the number of calls users may make in a particular period of time.

- To avoid code injection attacks, validate inputs.

When validating inputs, make sure the data is in the right format and that any characters that may be harmful code are removed.

- Filter client requests and block geographic areas that you don't wish to see.

Limit client permissions and capabilities to the minimum required to use the API service to reduce security concerns. Ensure that inappropriate requests are rejected by the API with a 405 response code (method not allowed).

Blocking access to your API from any location where you don't do business is one option. Furthermore, if you notice an attack, you may stop it in its tracks by blocking GET/POST requests from that location. Blocking countries/regions from sending GET/POST queries to your API may be the quickest method to stop an ongoing attack.

- XDR(Extended Detection and Response) systems, a new form of security solution that protects APIs holistically, should be considered.

Whereas EDR (Endpoint detection and response) enhanced malware detection above antivirus capabilities, XDR expands EDR's scope to include more widely deployed security solutions. XDR offers a greater range of capabilities than EDR. It makes use of the most up-to-date technology to improve visibility and gather and correlate threat information, as well as analytics and automation to assist identify existing and future threats. XDR secures API endpoints using HTTPS monitoring, API call monitoring, IP address filtering, JSON Web Token (JWT) and Input validation.

10. A tag in Git ?

A tag is similar to a non-changing branch. Tags, unlike branches, have no history of commits once they are established. Tags are identifiers for certain points in Git's history. Tagging is a method of capturing a specific period in time for the purposes of a version release. Annotated tags and lightweight tags are the two sorts of tags. Annotated tags are typically preferable since they contain more useful meta data about the tag.

Annotated tags are tags that hold additional Metadata such as the developer's name, email address, date, and other information. In the Git database, they are stored as a collection of objects.

It is advised to generate an annotated tag when pointing and storing a final version of any project. You can create a light-weight tag if you only want to make a temporary mark or don't want to share information.