Prottay Ray (Super-X, fw11_188)

1. What is a HashMap, explain the internal working of HashMap (in Java). Explain the insertion, retrieval and deletion process with an example, especially in the context of hash collision.

Ans:- HashMap is a popular data structure, which is extensively used a lot of times in real-world programming. It is useful because of it's highly optimal operations with respect to time complexities. Some other data structures viz. HashSet internally use HashMap for their implementation.

HashMap is implemented with the help of Hash tables in Java. Hash tables are a set of key-value pairs in the computer memory, where the key is a particular memory address and the values of the key-value pairs are lists of values of a particular type, which are stored in the memory address pointed by the key. Now there's a reason why we store a list there instead of just one value.

HashMap is called "*hash"* map instead of just map, because the mapping of values to particular keys are decided by some functions called hash functions, which output certain values called "*hashes"* which form the key in the key-value pair data structure. These hash functions take their input those values which are intended to be stored in the data structure. On performing some operations on the value, the hash function generates a hash value which would remain same throughout one execution of the application.

The purpose of HashMap is to reduce the chances of collision as much possible, and always point to a random new memory location of any given value. Although sometimes it is possible that a same hash is assigned to multiple distinct values, which gives rise to collision. Collision simply refers to the situation when multiple distinct values get the same hash.

Insertion

Keeping in mind the possibilities of hash collision, the insertion operation is designed for hashmaps. Whenever we endeavor to insert any value in a hash map corresponding to a specified key some steps happen. The user-specified key is converted to hash and the particular memory location is checked. We maintain a linkedlist in that location. The purpose of keeping linked list to ensure that the value is there even if any hash collision happens. In this way we get two scenarios:

i) When no collision happens – The location is checked and found that there is no value in the locatin directed by the generated hash. A linked list containing only the value to be put in that location is created and is placed in that location.

ii) When collision happens – The location is checked and found that multiple (one or more) values are present in that location. In such case the linked list present there is traversed till its end is reached and then the new intended value is added to the linked list.

Deletion

Keeping in mind the possibilities of hash collision, the deletion operation is designed for hashmaps. Whenever we endeavor to delete any value in a hash map corresponding to a specified key some steps happen. The user-specified key is converted to hash and the particular memory location is checked. We find a linkedlist in that location. The purpose of that linked list was to store the values properly even if any hash collision had happened. In this way we get two scenarios:

i) When no collision happens – The location is checked and found that there is only one value (in the form of a linked list) in that location. That particular value is removed from the one-valued linked list, and the location is again empty. If there was no value in that location then nothing would happen.

ii) When collision happens – The location is checked and found that multiple (one or more) values are present in that location in the form of a linked list. In such case the linked list present there is traversed till its end is reached and the last entry in that linked list is deleted. This is done to ensure that only the last value from the linked list is deleted, the other value(s) present in that location remain preserved.

Retrieval

Keeping in mind the possibilities of hash collision, the retrieval operation is designed for hashmaps. Whenever we endeavor to retrieve any value from a hash map corresponding to a specified key some steps happen. The user-specified key is converted to hash and the particular memory location is checked. We find a linked list in that location. The purpose of that linked list was to store the values properly even if any hash collision had happened. In this way we get two scenarios:

i) When there is no value in the specified location – When the pointed location is visited after hashing and found that no value is present in that location, a null value would be returned in that case, indicating that no value is present in that location.

ii) When no collision happens – The location is checked and found that there is only one value (in the form of a linked list) in that location. That particular value is returned from the one-valued linked list.

iii) When collision happens – The location is checked and found that multiple (one or more) values are present in that location in the form of a linked list. In such case the linked list present there is traversed till its end is reached and the last entry in that linked list is returned. This is done to ensure that only the last value from the linked list is returned, the other value(s) present in that location remain preserved.

Eg :-

```java
import java.util.*;

/**
 * HK
 */
public class Main1 {

    static class Address {

        Long zip;
        String details;

        public Address(long l, String string) {
            zip = l;
            details = string;
        }

        @Override
        public boolean equals(Object obj) {
            // TODO Auto-generated method stub
            if (getClass() != obj.getClass() || ((Address) obj).zip
!= zip || obj == null) {
                return false;
            }
            return true;
        }

        @Override
        public int hashCode() {
            // TODO Auto-generated method stub
            return Objects.hash(zip);
        }

    }
    public static void main(String[] args) {

        Map<Integer, Address> hm = new HashMap<>();

        Set<Address> hSet = new HashSet<>();

        //Adding a value
        hm.put(1, new Address(700120l, "Barrackpore"));
        hm.put(2, new Address(700120l, "Shanti"));
        hm.put(3, new Address(700121l, "Barrackpore"));
        hm.put(4, new Address(700121l, "Shanti"));

        hSet.add(new Address(700120l, "Barrackpore"));
        hSet.add(new Address(700120l, "Shanti"));
        hSet.add(new Address(700121l, "Barrackpore"));
        hSet.add(new Address(700121l, "Shanti"));

        //Retrieving a value
        Address address = hm.get(1);

        System.out.println(address);

        System.out.println(hSet);
    }

}
```

2) What is the general contract between hashCode and equals methods in Java? What happens if the contract is broken, explain with an example?

The general contract between the hashCode() and equals() method is that, whenever we need to implement the hashCode () function, we must implement the equals() method, and vice versa. We should do this because both these functions have inter-related funtions.

Suppose we implement equals() with all only one class field, while hashcode is not defined accordingly or vice versa, then hash map and equality would function inconsistently. This would be inconsistent behaviour for the application as a whole. The live explanation would make it even clearer. Below is the code example :-

```java
import java.util.*;

/**
 * HK
 */
public class Main1 {

    static class Address {

        Long zip;
        String details;

        public Address(long l, String string) {
            zip = l;
            details = string;
        }

        // @Override
        // public boolean equals(Object obj) {
        //     // TODO Auto-generated method stub
        //     if (getClass() != obj.getClass() || ((Address)
obj).zip != zip || obj == null) {
        //         return false;
        //     }
        //     return true;
        // }

        @Override
        public int hashCode() {
            // TODO Auto-generated method stub
            return Objects.hash(zip);
        }

    }



    public static void main(String[] args) {

        Map<Integer, Address> hm = new HashMap<>();
        Set<Address> hSet = new HashSet<>();

        //Adding a value
        hm.put(1, new Address(700120l, "Barrackpore"));
        hm.put(2, new Address(700120l, "Shanti"));
        hm.put(3, new Address(700121l, "Barrackpore"));
        hm.put(4, new Address(700121l, "Shanti"));

        hSet.add(new Address(700120l, "Barrackpore"));
        hSet.add(new Address(700120l, "Shanti"));
        hSet.add(new Address(700121l, "Barrackpore"));
        hSet.add(new Address(700121l, "Shanti"));

        //Retrieving a value
        Address address = hm.get(1);

        System.out.println(address);

        System.out.println("Check equlity of 1 and 2 :- " +
hm.get(1).equals(hm.get(2)));

        for (Address addresss : hSet) {
            System.out.println(addresss);
        }
    }

}
```

3) What is an operating system, discuss its main functions and responsibility.

An operating system is the main software in any computation device which acts as an interface between the hardware, on which the operating system is installed, and the user. Softwares are generally of three kinds: -

i) System Software (eg. Windows 10, Linux)
ii) Application Software (eg. MS Office Word 365)
iii) Utility Software (eg. McAfee antivirus)

Operating System is a system software, and any other software is basically installed within the operating system. The most important functions and responsibilities of the operating system include:-

i) <u>Process Management</u>: The operating system is the primary software responsible for management of all processes. It coordinates the execution time of the processes, and all other major activities to manage the smooth execution of multiple processes is done by the operating system.

ii) <u>Memory Management</u>: The operating system is primarily responsible for the memory management of the computer. Memory is the resource available to all the processes at the same time. It is possible that one process takes up all the memory available to the whole computer and leave no memory for the other processes. All such possibilities of mishap are eliminated by the operating system, and proper and optimized utilization of the memory is ensured by the operating system. Eg. For every process only a particular quota of memory is allocated by operating system.

iii) <u>Context Switching</u>: One of the primary tasks that the operating system does is context-switching to create a multi-tasking environment with a smooth experience. At relevant times the operating system takes away the control to resources from one process, either queues it to the process queue, or kills it, and gives the access to system resources to some other process which was in the waiting state in the queue in the front position.

iv) <u>Inter-Process Communication</u>: Many processes require to communicate among each other thru message passing. This behavior is required for the proper functioning of various softwares. This is facilitated by the operating system.

v) <u>Process Scheduling</u>: Process scheduling is one of the major functions of the operating system. The operating system maintains a queue for all the processes, and applying one of various logics the processes are scheduled and given access to resources. Some of the popular strategies are shortest-job-first, first-come-first-serve, priority, etc.

vi) <u>Resource Management</u>: One of the crucial things that the operating system looks after is the resource management that it has to do. An operating system has a given set of resources available to itself, and many processes are present which can demand resources in unforeseen manners and quantities. To manage that efficiently, the operating system gives resource-access to processes in a proper manner such that a smooth functioning is ensured.

4) Discuss any five pros and cons of using cloud services (such as AWS) over on-prem servers.

Five pros :-

i) For on-prem servers, there's a **huge initial cost** involved in buying the hardwares, which may be a burden for any company to start. On the contrary cloud services can provide well-managed same facilities with zero-investment for hardware setup. You need to pay only for that much amount, which you are utilizing.
ii) For on-prem servers, **scalability is a great challenge**. It is not very easy to quickly scale the on-prem servers with quickly growing needs. Configuring such hardwares in less time with the best functionalities is difficult to achieve. On the contrary, with the help of cloud services we can very easily scale up and down(even auto-scale) our system as and when requirement arises, much more flexibly.
iii) For on-prem servers, it is very difficult, to server customers **globally**. If our application grows, we would need multiple servers across multiple locations across the globe to serve a global customer base. Different countries have different laws and different policies, and it is very difficult to setup multiple on-prems servers globally. Also, it would incur a lot of cost to do so. On the contrary, with the help of cloud services we can spin up multiple servers globally for our application in just a matter of few clicks.
iv) To maintain on-prem servers, incurs a lot of cost always, to have skilled engineers to manage them, and a huge **maintainance** cost is always there. On the contrary, using cloud-services we would require much less budget to maintain the bills and there's no worry to take care of the servers and its extra costs.
v) In case of any data-center failure, a huge loss is faced by the owner. In case of on-prem servers, the company has to bear it, but it is not the case for using cloud-services.

Five cons:-

i) In case of on-prem services the companies have full owner-ship of the data and there's no fear associated of data stealth.
ii) Using on-prem servers we are in possession of our data, and we won't have to store it in any foreign location.
iii) If there's an outage of the cloud-service provider, we would have to face and suffer the outage as well, in case of cloud services. But there's no such fear if we have our own servers. We can maintain and manage our servers in our own way.
iv) For cloud-services we need to adhere to the external policies of cloud-service-provider. But in on-prem we don't have such bindings to foreign providers.
v) We can use any choice of resource and **customize our server configurations as per our choice**. If we are using cloud services, then there's no scope of customizing the servers as per our needs rather we need to choose a configuration from a list of those which the provider provides.

5) Explain any five AWS services. Discuss a few use cases of each of these services.

The five services are as follows :-

i) *EC2*: Amazon EC2 is basically a compute power service, thru which amazon provides us with a virtual computer with our chosen configuration (from a list of configurations which they provide). It is greatly used whenever we want to host any application over the internet.

ii) *S3*: Amazon S3 is yet another popular service provided thru which Amazon provides us with storage where we can store our files over the internet. Each of our files are stored in a dedicated memory area called "bucket" in their global memory storage. The buckets are dedicated shares our Amazon's storage provided to the requesting users, wherein the users drop their files.

iii) *RDS*: Amazon RDS is one of the most famous services provided by AWS. It means Relational Database Service by which Amazon provides us with the database engines of all popular relational databases viz. MySQL, MaraiaDB, etc.

iv) *CloudFront*: Amazon CloudFront is the CDN service provided by Amazon, by which amazon helps us store various copies of our data in multiple edge location globally to serve our customers better over the internet across any geographic location.

v) *SNS*: Amazon SNS is the notification service. With the help of this service we can send notifications like email notifications, sms notifications etc to our clients seamlessly. This is used by applications to send notifications to the clients.

6) What is CAP theorem, discuss a scenario where CAP theorem helps you choose a technology over the other or helps you make a design decision.

CAP Theorem states that, we **cannot** have **tight consistency, tight availability, and partition tolerance** at the same time, for any given database. We can have either tight consistency and partition tolerance with the help of SQL databases. We can have High Availability and partition tolerance with the help of NoSQL databases. While we can have both tight consistency and high availability in case of small datasets. When we have a large user base to query our database, availability is very highly required, and we don't strictly need ACID properties, we generally go with NoSQL databases. Otherwise we go with SQL databases.

SQL Databases: MySQL, MariaDB, MS SQL, etc.

NoSQL Databases: MongoDB, Apache Cassendra, etc.