So, whenever the bucket will be filled with 12 keys, the threshold will be doubled & the capacity will be increased.

Q Can we override a static class in JAVA?

Ans No, we can't override.

Q What is marker interface?

Ans Blank interface with no implementation inside it.
Serializable & Clonable are marker interface which tells the program about the requirment.

Q Types of collection used?

Ans ArrayList, HashMap, HashSet, TreeSet

Imp Q Internal working of HashMap?

Ans
- Creates a hascode of the key.
- generates the index or calculate the index
- allocate the values & keys to the nodes.

Q Java 8 hashmap vs earlier HashMap?

Ans In Java 8 hashmap, buckets containing a large number of colliding keys will store their entries in a balanced tree instead of the linked list after certain threshold.

This JDK 8 change is applied to HashMap, LinkedHashMap and ConcurrentHashMap.

Q How Concurrent hashmap is different from Hashmap?

Ans Hashmap is unsynchronized & not thread safe while Concurrent hashmap is a thread safe map.

Q HashTable vs concurrent HashMap as both are thread safe? Difference?

Ans. Underlined data structure for concurrent hashmap is HashTable.

- HashTable uses single lock for whole data.
- ConcurrentHashMap uses multiple locks on segment level instead of object level i.e. whole Map.

Q Reverse the string, program?

Q JAVA 8 features?

Q Stream API?

Q What is the way to use the second level cache in hibernate?

Ans By using 3rd party libraries like encache, swarmcache annotation like @cache & @cachable.

Q Why JAVA is platform independent?

Ans Becoz programs are compiled into byte code and that byte code is platform independent. Any machine having JVM can execute the JAVA byte code.

Q What is a singleton class?

Ans A class that can have only one object at a time. After first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created.

Q Is Clonable interface a functional interface?

Ans No, clonable is a marker interface which doesn't have any methods or implementation.

Q Diff. b/w Hashset & Treesets?

Ans
- Hashset is a set where the elements are not sorted or ordered.
- Treeset is a set where the elements are sorted.
- Hashset is faster than a Treeset.

(See geeksforgeeks Example)

Q When to use Comparator or Comparable?

Q Diff. b/w Comparable & Comparator? (Interfaces)

Ans
- used to sort the collection of objects.
- if sorting of objects needs to be based on natural order then use comparable whereas if your sorting needs to be done on attributes of different objects then use comparator in java.
- A comparable object is capable of comparing itself with another object. we use compareTo() method.
- compare() & sort() functions belongs to Comparator interface.

Q 'Try with resources' in JAVA?

- try-with-resource statement is a try statement that declares one or more resource.
- The try-with-resource statement ensures that each resource is closed at the end of the statement execution

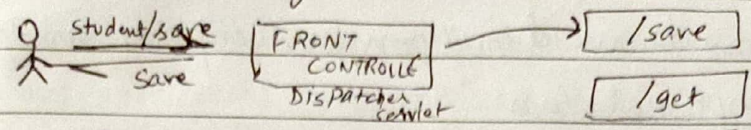
Q Write a program to check weather the string is Palindrom or not?

Ans

spring MVC X

Q. What is DispatcherServlet in Spring MVC?

Ans Acts as a front Controller whose job is to make the right decision that who is the right Controller to handle the request.



Q. What is controller?

Ans A controller is a spring component that processes the request coming from client side.

Q. Annotations used in Spring MVC / SpringBoot?

Ans @ComponentScan @Bean @Autowired

@Secured @SpringBootApplication @RestController

@RequestMapping, etc.

Q. Diff. b/w abstract class & Interface?

Ans
- Abstract class can have abstract & non abstract methods, while Interface can have only abstract methods since Java 8, it can have default & static method also.
- Abstract class achieves partial abstraction while interface achieves fully abstraction.

Q. Difference b/w equals method & == operator?

Ans equals() method for content Comparison whereas == operators checks if both the object points to the same memory location.

Q. What type of garbage collectors are provided & how to choose which Garbage collector to use.

Ans
- Serial Garbage Collector
- Parallel Garbage Collector
- Concurrent MarkSweep
- Garbage first (G1) garbage collector.

Q How do you initialize the Integer?

```
BigInteger big = new BigInteger('1');
```

Q How to avoid null pointer Exception in Java 8?

Ans Optional class

Q What are the various methods in optional class?

Ans ofnullable, ispresent, get()

Q What is functional Interface? / Name some functional Int?

Ans
- an interface that contain only one abstract method.
- Can have any no. of default methods.

Q What is marker interface? Give example. (Cloneable, Serializable)

Ans A marker interface is an interface that has no methods or constants inside it. It provides run-time information about objects so the Compiler and JVM have additional info about the objects.

Q Diff. b/w throw / throws & throwable?

Ans
- throw → throw keyword is used throw an exception explicitly by the programmer within the method implementation
- throws → used with method signature, it is used to declare an exception. It tells the programmer that there might be an exception. so it is better to be ready with try-catch block or proper exception handling code.

Q Foreign Constraints?

- The foreign key constraint is used to prevent actions that would destroy links b/w tables.

• Throwable : Throwable is a superclass of all exceptions and errors.

Q Can we use HashMap in multithreaded environment?

Ans We can, but we should not use HashMap in multithreaded environment instead we can use ConcurrentHashMaps.

• if multiple threads are adding to the same HashMap instance without it being synchronized, it will run into problems.

Q Can we put duplicate elements in Arraylist?

Ans Yes, we can put duplicate elements in the arraylist.

Q "CopyonArraylist" ?

Ans Majorly used where Read operations are frequent and update operations are rare in thread based environment. It is thread safe variant of arraylist.

Q Annotation used while creating RestAPI's ?

Ans @GET @POST @PUT @PATCH @DELETE
@RESTCONTROLLER @Autowired

Q Annotation used in app.properties file to get some values?

Ans @Value, annotation for reading values from an application.

Q Map & flatmap in StreamAPI?

Ans map() can be used where we have to map the elements to a certain function and return the Stream of results.
flatmap(), where we have to flatten or transform out the String.

Q SpringBoot-Dev-tools?

Ans Whenever there is a change in the class path or any file, the server starts automatically on its own.

Q Spring Data / Spring JPA ?

Ans detailed in start of the copy . . .

Q @ComponentScan and @Bean ?

Ans
- @ComponentScan is used when we want to scan a package for beans. It is used with annotation @Configuration
- @Bean, a method level annotation, it tells the method to produce a bean to be managed by Spring Container.

Q what are different design Patterns ?

Ans
- Singleton Design Pattern
- Prototype - cloning of a existing object instead creating new one
- Facade -
- Factory Pattern -
- Builder Pattern - construct a complex object from simple objects using step by step approach.

Q Diff. b/w Heap Memory vs Stack Memory ?

Ans
- Heap memory & stack memory both resides on the RAM.
- Heap memory is the memory allocated to the JVM which is shared by all the resources in the application.
- stack memory allocation is considered to be faster bcoz data is added & removed in last-in-first-out manner.
- Heap memory is also known as dynamic memory allocation.

Q Why String is immutable in Java?

Ans Because string objects are cached in string Pool.

Q How to get values from HashMap? (without iterator)

Ans map.values().toString();

Eg Map<Integer, String> map = new HashMap<>();
map.put(1, 'Amit'); map.put(2, 'Aash');
System.out.println(map.values().toString());

(X) Q method refrence?

Ans Java 8 Feature, is used to refer method of functional interface. Three types are:
① Reference to a static method ② Reference to an instance method.
③ Reference to a constructor

Q @RequestBody vs @QueryParam vs @ResponseBody

Ans @RequestBody, maps the HTTP Request body to Transfer object enabling automatic deserialization. It corresponds to the JSON sent from client side.

@QueryParam, allows to inject individual URI query parameters into your Java Parameters. (used with method parameters)

Q How to create SpringBootApplication in VsCode?

Ans Via Spring Initializer, search for vscode-spring-initializer. Install the extension, with Spring Initializer start generating Maven or Gradle project.

(X) Q Read about Agile methodology? How we follow that in our project.

Ans

Q use of Generic in collections? what is <> in collections?

Ans Generics enables types (classes & Interfaces) to be parameters ~~of~~. Generics shift burden of the type safety from programmer to compiler.

Q @PathVariable vs @ResponseBody

Ans @PathVariable, used to handle template variables in the request URL mapping and set them as method parameters.

@ResponseBody, binds a method return value to the web response body.

Q @Qualifier vs @Autowired @Primary

Ans @Qualifier, eliminate the issue of which bean needs to be injected, together with the name of specific implementation we want to use, we can avoid ambiguity when spring finds multiple beans of the same type

@Primary, which bean to inject when ambiguity is present the bean associated with primary annotation will be given preference

Q why are we using optional class specifically?

Ans. To avoid abnormal termination due to null pointer exception, we use optional class.

- The main advantage is that no more too many null checks and null pointer exceptions.
- It checks for presence of value for particular variable.

Q- How to take XML as input & give HTML response as output?

Ans

Q Dependency injection in JAVA?

Ans

Dependency injection is a technique whereby one object supplies the dependencies of another object.

Three ways of dependency injection:

(1) Constructor injection – dependencies are provided through class constructor.

(2) setter injection – the client exposes the setter method that the injector uses to inject the dependency.

(3) interface injection – inject dependency into any client passed to it.

Responsibilities of Dependency Injection:

(1) Create the objects

(2) Know which class require those objects.

(3) And provide them all those objects.

Q Exception on the whole App level?
How it is handled?

Ans It can be handle via different strategies like:
- Centralized Exception Handling
- Logging and Monitoring
- Notifications and Alerting
- Retry and Circuit Breakers Patterns
- Robust error handling in APIs

Q How to make constructor as final?

Ans Constructor can't be made final.

Q Can we inherit constructors in java?

Ans Constructors cannot be inherited in Java therefore you cannot override constructors.

Q Can you sort Hashmap in Java?

Ans directly we cannot sort hashmap in Java, but if we use TreeMap Constructor and pass the object of hashmap class as an argument. then we can sort hashmap by keys.

Q Why multiple inheritence is not allowed in Java?

Ans (To prevent ambiguity)
Does not support becoz:
(1) every class is a child of object class.
when it inherits from more than one super class, subclass gets the ambiguity to aquire the property of Object class.

Q How to achieve multiple inheritance in Java?

Ans The only way to implement multiple inheritance is to implement multiple interfaces in a class.

In java, one class can implement two or more interfaces.

No ambiguity bcoz all methods declared in interfaces are implemented in class.

Q collection vs Collections?

Ans. • Collection is a interface in java.util.package. Used to represent a group of individual objects.

Eg The List, Queue & Set are sub-interfaces of Collection.

• Collections, is a utility class which has several utility methods to operate on collection like sort(), min(), max().

Q Collections vs StreamAPI?

Ans Collection is a framework to handle group of objects like Set, List, Queue, etc.

StreamAPI is basically used to produce the stream of desired results based on the operation performed on collection.

Date
Page

iterators

Q what is FailFast & Failsafe?

Ans

FailFast is used by Arraylist & LinkedList;
During the iteration, if you are trying to perform any modification operation like adding an element or removing an element or you are trying to modify the existing structure then it gives the exception, ConcurrentModificationException

failsafe, it is thread safe. examples are ConcurrentHashMap & CopyonArrayList, etc.

If you are performing any modification during the iteration so it won't give the concurrent modification exception, it will work fine.

failsafe consuming more memory becoz the modifications are happening on copy not on original list/collection.

Q some Junit annotations?

Ans @Beforeclass → Run once before any of the test methods in the class.
@Afterclass → Run once after all the test in the class have been run
@After → it runs after @Test

@Test → testcase to be executed
@Before → Run some statement before each test case

Q Map with help of StreamAPI?

Ans

Q Why we need serializable interface?

Ans When we need to store a copy of the object and send them to another process which runs on the same system, in the form of bytes or byte stream.

Q Bean lifecycle?

Ans Two imp. bean lifecycle callback methods which are required at the time of bean initialization and its destruction.
We simply declare the <bean> with initmethod and destroy-method parameters.

Q Callable interface?

Ans One feature lacking in Runnable is that we cannot make a thread return result when it terminates i.e. when run() completes.

For supporting this feature, the Callable interface is present in Java.

Its an asynchronous computation, whose value is available via a Future Object.

Q What is Error?

Ans Error is a subclass of the 'Throwable' class that represents serious problems that applications encounter.

- StackOverflowError
- OutOfMemoryError
- NoClassDefinedError
- InternalError

Q Static binding & Dynamic binding?

Ans
- Static binding (Early binding) → compiler determines which method to invoke based on type of object at compile time.
- Method overloading is an example of static binding.
- Dynamic binding, method call is resolved at runtime based on actual object.
- Method overriding is an example of dynamic binding.

Q Types of synchronization?

Ans

Q What is synchronization?

Ans

Q I'll pass employee object to HashMap as key, how to check weather its unique or not?

Ans

Use "Hashmap.containsKey()" method.

Before adding an 'Employee' object as a key to the HashMap, you can use 'containsKey()' method to check if key is already present.

Singleton class

(*) When only one object of a class is created and the same object with shared with all the resources then that class is known as singleton class.

Advantage of Singleton class :

↳ increase in the performance
↳ memory utilization improvement.

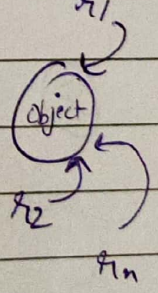
(*) Some examples of singleton classes are :

- Runtime
- Business Delegate
- Service locator

etc.

Eg

```
Runtime r1 = Runtime.getRuntime();
Runtime r2 = Runtime.getRuntime();
    !
    !
Runtime rn = Runtime.getRuntime();
```



(*) We have two approaches to create singleton classes :-

Private constructor used to create singleton class.

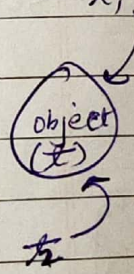
Approach 1

```
class Test
{
//private static variable
private static Test t = new Test();
//private constructor
private Test()
{

}
//public factory method
public static Test getTest(){
    return t;
}
}
```

So, now whenever somebody asks for Test.

```
Test t1 = Test.getTest();
Test t2 = Test.getTest();
```



(*) Runtime class is also based on the same approach

One cannot create the child class of singleton class becox it has private constructor which restricts any class to be get inherited into child class.

Approach 2 (Recommended)

Don't create the object initially, only create object when it is required during the first time.

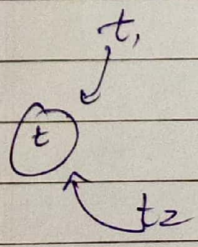
```
class Test {
private ~~public~~ static Test t = null;

    private Test()
    {
    }
public static Test getTest()
    {
        if(t == null)
        {
            t = new Test();
        }
        return t;
    }
}
```

So when the request comes for the first time then only object gets created.

```
Test t1 = Test.getTest();

Test t2 = Test.getTest();
```



Hence, Test class is singleton class.

Spring Boot

(1) Thymeleaf

- It is an open source Java library.
- a server-side Java template engine.

JSP is more or less similar to HTML. But it is not completely compatible with HTML like Thymeleaf.
We can display a thymeleaf template file normally in the browser while the JSP file does not.

(2) Spring Boot Key Components

- Spring Boot auto-configuration.
- Spring Boot CLI.
- Spring Boot starter POMs.
- Spring Boot Actuators.

(3) Starter dependency of Spring Boot module —

- Data JPA starter
- Web starter
- Security starter
- Email starter
- thymeleaf starter
- Test starter

(*) Profiles

While developing the application we deal with multiple environments such as Dev, QA, Prod and each environment requires a different configuration.

To make this easy & clean, Spring has the provision of profiles to keep the separate configuration of environments.

(*) Enable debugging log

- we can start the application with --debug switch
- we can set the logging.level.root = debug property in application.property file.

(*) Deploy SpringBoot Application as Jar and war

- Spring tackles this problem by providing spring-boot-maven-plugin, to package a web application as an executable Jar.
- to build a war, we change the packaging element to war.

Eg <packaging> war </packaging>

(x) Spring Boot Dev-tools

- is a set of tools making the development process easier.
- Applications using DevTools restart whenever a file on the classpath changes.

(x) @Service Vs @Component @Bean

@service annotation is used in your service layer and annotates classes that perform service tasks or run Business logic.

@Component is an annotation that allows Spring to automatically detect our custom beans!
- it is a class level annotation.

@Bean is also an annotation that Spring uses to gather beans at runtime, but it is not used at the class level.

We annotates methods with @Bean so that Spring can store the methods result as a spring bean.

When we want to share single instance of a class across the whole application.

Used at method level.

Imp

(*) JPA vs Hibernate

↳ • JPA (Java Persistance API) is a specification of Java which is used to access, manage and persist data between Java object and relational database.

• it is a bridge b/w Java objects and relational database systems.

• Being a specification, JPA doesn't perform any implementation by itself.

So ORM (Object Relational Mapping) tools like Hibernate, Toplink implements JPA specifications for data persistance.

↳ Hibernate

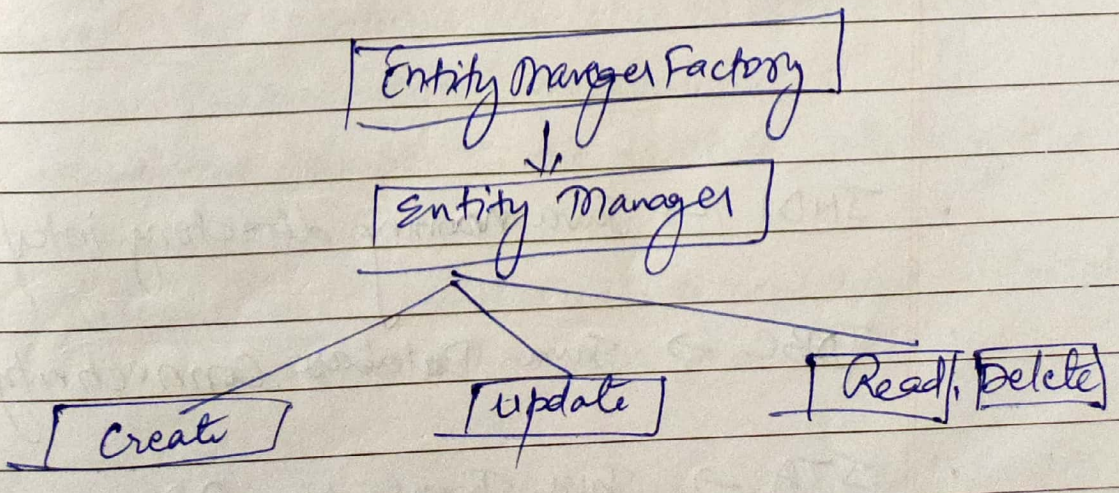
• A hibernate is a Java framework which is used to store the Java objects in the relational database systems.

• it is a open-source, lightweight ORM tool.

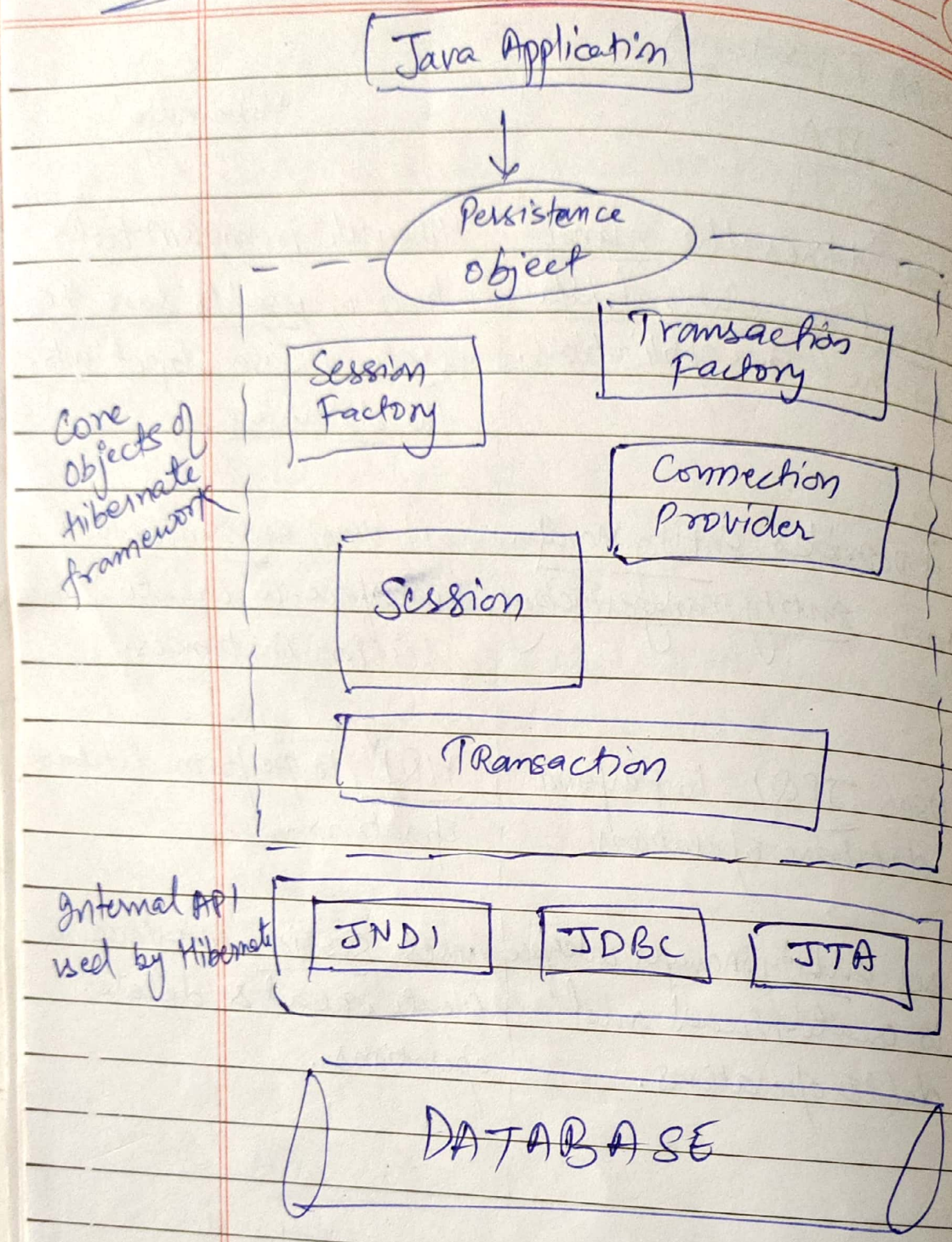
• it is an implementation of JPA, it follows common standards provided by the JPA.

(JPA Repository)

JPA	Hibernate
JPA defines the management of relational data in the Java applications.	Hibernate is an ORM tool which is used to save the state of Java Object into the database.
it provides Entity Manager from Entity ManagerFactory	it uses sessionFactory interface to create session instances.
Uses JPQL to perform database operations	HQL to perform database operations.
uses Entity Manager interface to create, read and delete operations.	uses Session interface to create, read & delete operations.



Hibernate Architecture



- JNDI → Java naming directory interface
- JDBC → Java Database Connectivity.
- JTA → Java Transaction API

Session (mandatory)

- it holds the first level cache of data.
- it provides an interface b/w the application and data stored in the database.
- The org.hibernate.session interface provides methods to insert, update and delete the object.

SessionFactory

- it holds the second level cache (optional) of data.
- it provides factory methods to get the object of the session.

(*) **Save()**
- Returns generated ID after saving, its return type is serializable.
- save the changes to the database outside of the transactions.
- session.save() will create a new row in the table for a detached object.

(*) **Persist()**
- It does not return generated ID after saving. Its return type is void.
- Does not save the changes to the database outside of the transaction.
- Stores object in Database.

@Entity

defines that a class can be mapped to a table.

@Id

corresponds to the primary key of the objects table.

@GeneratedValue

@Id will only declare the primary key, it will not insert generated value, if we use @generatedvalue then it will generate the value of the field.

⭐ first level cache Vs Second level cache

first level cache	Second level cache
• first level cache is a session level cache & it is always associated with session level object.	• Second level cache is session factory level cache and it is available across all sessions.
• Enabled by default.	• Not enabled by default (optional)
• available for a session.	• available across all session.

- First level cache is used for minimizing DB interaction by caching the state of the object.

Instead of updating after every modification done in the transaction, it updates the transaction only at the end of the transaction.

- Second level cache

While running the transactions, it loads the object at the session factory level, so that those objects will be available to the entire application.

@ Column(name = "Employee_Name")
- Used to give column names in the table from Java class/POJO.

@ Transient
- temporary storage of data & the column will not be created in database.

@ Embeddable
- used to insert object into another table.
- it will insert one class object into another class.

Database Mapping Relationship

- @ OneToOne
- @ OneToMany
- @ ManyToOne
- @ ManyToMany

@OneToOne

Eg: one laptop is assigned to one student.

FETCH Techniques

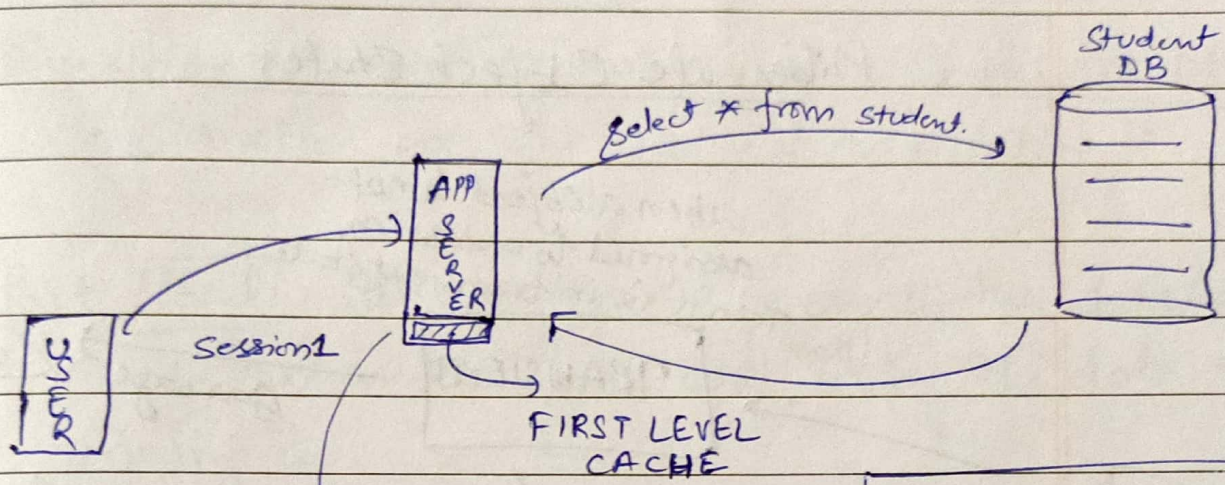
- EAGER
- LAZY

- it will return everything.

(fetch = FetchType.EAGER) needs to be mentioned and then get function will bring everything.

- get is an example of lazy fetch technique.
- By default, it is lazy.
- it will only fetch data when you need it.
- if we dont need to get all the dependent data then we use lazy.

Hibernate Caching



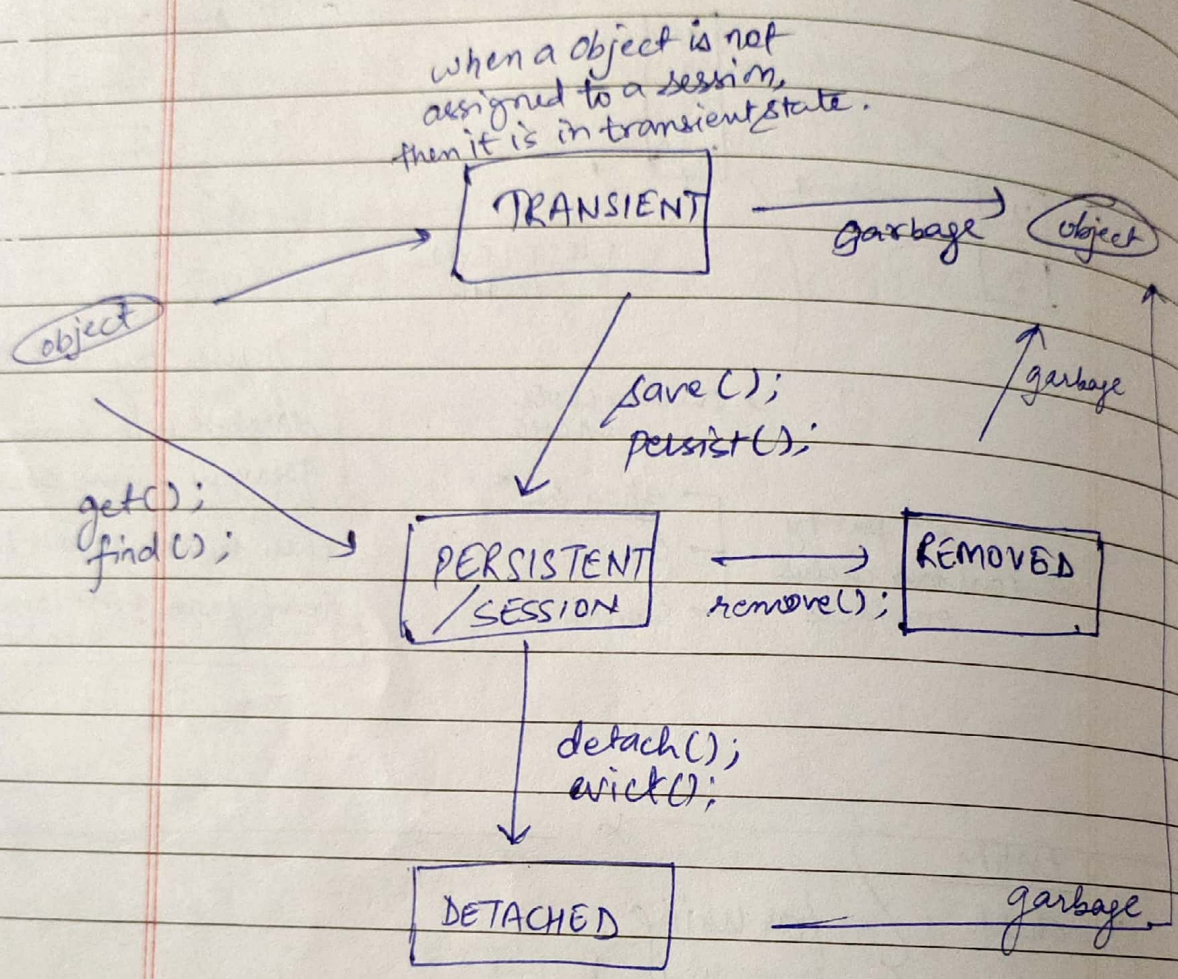
3rd party second level cache providers
- ehcache *
- OS
- Swarm

if user try to hit the database with same query in same session, then the result will be read from First level cache.

Entity
- @cachable
- @cache

for using second level cache.

Hibernate Persistence Life cycle
OR
Hibernate Object States



session.evict(s);

- it detaches the object from persistent state to detached state, if we don't want to close the session.

- Else if we close the session all the persistent state objects are detached automatically.

Get Vs load

- Everytime we use `get` method, it will hit the DB no matter we are using it or printing it.

- But `load` will not fire the query to the database until and unless you are not using the data or printing it.

load only fires query to the DB when the result is been used somewhere.

get will fire the query to DB, eventhough the result is not used anywhere.

- get gives you the object.
- Load gives the `proxy` object (its a kind of fake object)

- Normally we use get method widely.
- load is used when we need proxy objects or fake objects

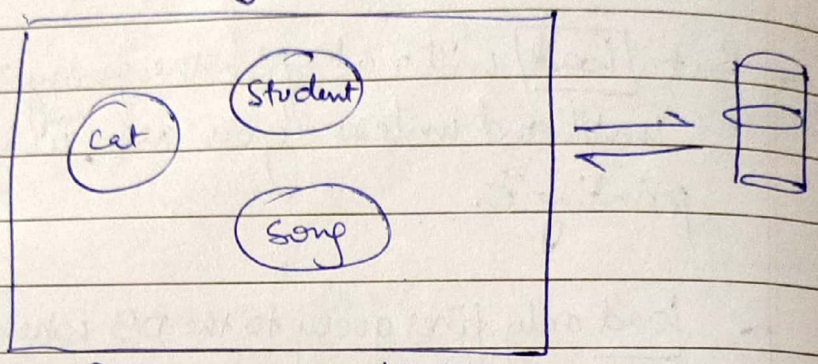
Exceptions

- When record not found, get method won't throw any exception if the record is not there, it will return you a null.

but, load method it returns object not found Exception when no record found.

Persistent Context. (session)

Whenever we open a session, the blocks is created in the memory.



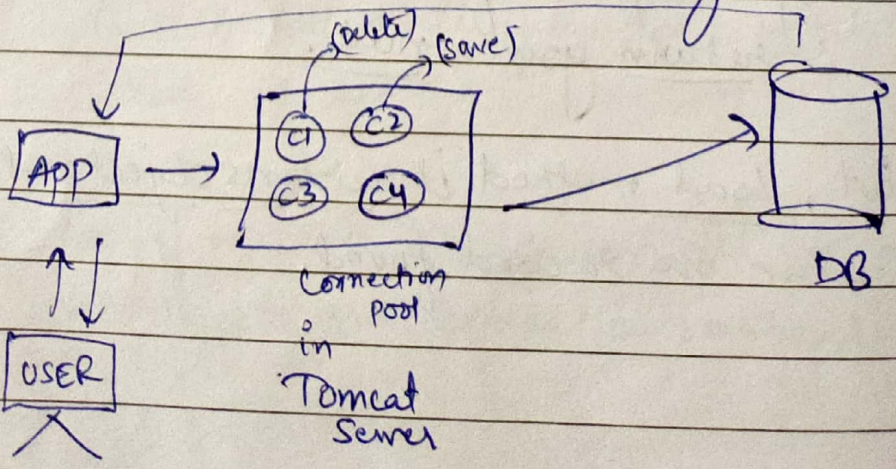
Persistent Context

- it stores all the state of objects.
- it also have session in it which communicate to the database.
- it is a first level cache.

Connection Pool

(Spring Boot uses by default Hikari connection pool)

- used for reusability of sessions.
- maintained in server side always.



- DataSource maintains the connection pooling.
- With spring DriverManager maintains the connection pool.

(*) @GeneratedValue

@GeneratedValue(Strategy = GenerationType. AUTO
IDENTITY
SEQUENCE
TABLE)

Auto : default method used by hibernate.

IDENTITY : my DB should generate the Primary key & that will be used by Hibernate.

SEQUENCE : to generate a sequence of primary key.

TABLE : to generate a primary key, hibernate is maintaing a separate table internally which uses SEQUENCE concept.

Date
Page

(4) ArrayList vs Linked List

ArrayList	Linked List
• Internally uses a dynamic array to store the elements.	• Internally uses a doubly linked list to store the elements.
• manipulation is slow	• manipulation is faster.
• Better for storing & accessing data.	• better for manipulating data.

(5) STREAM API — Java 8 feature

- it is used to process collection of objects.
- it takes input from collections, Array & I/input O/output streams.
- they only provide results as per the pipelined methods.

Intermediate Operations:

1) map: the map method is used to return a stream consisting of the results of applying the given function to the elements of this stream.

Eg

```
List.add();
.sort();
List number = Arrays.asList(2,3,4,5);
List square = number.stream().map(x -> x*x).
              collect(Collectors.toList());
```

2) filter :

the filter method is used to select elements as per the condition passed as an argument.

Eg

```
List names = Arrays.asList("Reflection", "Collection");
List result = names.stream().filter(s -> s.startsWith
                ('c').
              collect(Collectors.toList());
```

3) Sorted :

the sorted method is used to sort the stream.

```
List names = Arrays.asList("Reflection", "Collection");
List result = names.stream().sorted().collect
              (collectors.toList());
```

Terminal Operations :

1) Collect : The collect method is used to return the result of the intermediate operations performed on the stream.

2) ForEach : Is used to iterate through every element of the stream.

3) Reduce : is used to reduce the elements of a stream to a single value.

Principles of JAVA

- Inheritance
- Polymorphism
- Encapsulation
- Abstraction

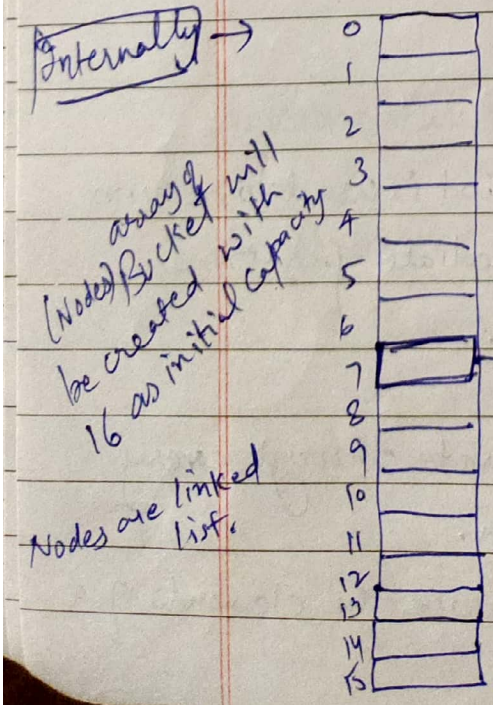
SOLID Principles of Java

Single Responsibility Principle (SRP)
Open-closed principle (OCP)
Liskov Substitution principle (LSP)
Inteyace Segregation Principle (ISP)
Dependency Inversion Principle (DIP)

HASHMAP

```
Map<String, Integer> map = new HashMap<>();
map.put("abc", 1);
```

Internally →



- hashcode = hash(abc) // 1122
- calculate index = index(hashcode) // 7
- allocate the values to the node or linked list