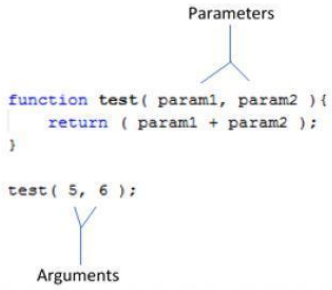
GENERAL

Difference between compile time and run time?
The time period in which the developer is compiling or converting the code into executable that is called as Compile time whereas the time in which the code is being run is called as Run time.

More information including runtime and compile time error:
https://www.javatpoint.com/compile-time-vs-runtime

Difference between parameters and arguments?
Parameter is the variable in the method definition whereas arguments are the data you pass into the method's parameter when the method is called.



Difference between function and method?
Functions are defined outside of classes, while methods are defined inside of and part of classes.

Difference between array and list?
Array has a fixed size and it's static while list has no fixed size and it's dynamic.

More information about pros and cons, speed, etc.:

https://www.javatpoint.com/difference-between-array-and-arraylist

Difference between HashMap and HashSet?

- HashSet doesn't allow duplicate values. HashMap store key, value pairs and it does not allow duplicate keys. If key is duplicate then old key is replaced with new value.
- HashMap internally uses hashing to store or add objects, HashSet internally uses HashMap object to store or add the objects.
- HashSet is slower than HashMap.

https://www.javatpoint.com/working-of-hashmap-in-java
https://www.javatpoint.com/working-of-hashset-in-java

What is the difference between HTTP and HTTPS?
HTTP (Hyper Text Transfer Protocol) is a request-response protocol which transfers data between the clients and servers. HTTPS (Hyper Text Transfer Protocol Secure) is a protocol that uses encrypted HTTP connection to keep the transferring data secure.

What is the difference between GET and POST?
GET and POST are both HTTP request methods. GET is used for requesting a data from a source. While POST is used for submitting data to a source for processing.

What is difference between Process and Thread?
Threads are used for small tasks, whereas processes are used for more 'heavyweight' tasks – basically the execution of applications. Another difference between a thread and a process is that threads within the same process share the same address space, whereas different processes do not.

https://www.tutorialspoint.com/difference-between-process-and-thread

What is enumeration?
An enumeration is used in any programming language to define a constant set of values. For example, the days of the week can be defined as an enumeration and used anywhere in the program.

What is generics?
Generics is a concept in programming in which the type of a method or a class is generalized.
Write a sample code for generic class and generic parameter.
https://www.javatpoint.com/generics-in-java

Is String mutable? (Mutable-> can be changed, Immutable -> cannot be changed)
The string is immutable, if you try to alter their values, another object gets created, whereas StringBuffer and StringBuilder are mutable so they can change their values.

https://pediaa.com/what-is-the-difference-between-string-literal-and-string-object-in-java/

https://www.javatpoint.com/immutable-string

https://www.javatpoint.com/difference-between-stringbuffer-and-stringbuilder

What Is an API?
An API (Application programming interface) is a collection of subroutine definitions, protocols, and tools for building a software. It interprets the data and sends the required information to you
https://www.youtube.com/watch?v=s7wmiS2mSXY

What is Deadlock?
It is a situation when a resource is being shared by two or more than two processes, and these processes are preventing each other from sharing the resource.
https://www.javatpoint.com/deadlock-in-java

You have a cake and a knife and you have to cut the cake into 8 equal pieces?
Cut the cake halfway its height, then cut the cake in 2 perpendicular lines.

OOP BASICS

What is an object?
An object is an instance of a class. It has its own state, behavior, and identity.
What is a class?
It is a blueprint that defines what an object should look like. Where you list all the fields and methods what an object should have.

Difference between object-based and object-oriented programming?
Object-based Programming is almost the same methodology as the Object Oriented Programming but it does not support Inheritance or Polymorphism. JavaScript is one of the examples of Object-based programming.
https://www.w3schools.in/java-questions-answers/difference-object-oriented-programming-language-object-based-programming-language/

What are main concepts/pillars of OOP?
1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

What is 'this' operator?
'this' pointer refers to the current object of a class.
https://www.javatpoint.com/this-keyword

What is 'final' keyword?
Protects methods or variables or classes to be over ridden or modified.
https://www.javatpoint.com/final-keyword

What is 'static' keyword?
A variable or a method can be accessed without requiring an instantiation of the class to which it belongs. However it can be changed but will changed for every object and that class.
https://www.javatpoint.com/static-keyword-in-java

What are constructors?
Constructors are special type of methods that are used for creating & initializing an object of a class.
1. Default Constructor (Automatically created if no constructor created.
2. Parameterized Constructor
https://www.javatpoint.com/java-constructor

What is the difference between Aggregation and Composition?
The composition is a strong association between two entities. For Example Car and Engine. It Car is destroyed, Engine is of no use. As compared to this, entities are independent of each other in Aggregation. For Example, Class, and Student. If Class is destroyed, Student can still exist.

https://www.tutorialspoint.com/Association-Composition-and-Aggregation-in-Java

ENCAPSULATION

What is Encapsulation?

Each class is just like a capsule that contains everything it needs and nothing more. Concept also used for information hiding. Like hiding an attribute through access modifier and accessing it through methods (getter and setter)

Why we use getter and setter?

If you have an attribute that is not visible from the outside of a class, bundle it with methods that provide read or write access to it, then you can hide specific information and control access to the internal state of the object or specify some constraints while accessing those attributes.

What are access modifiers?

Access modifiers are used to define the visibility of classes, methods, and attributes. Each of them specifies a different level of accessibility, and you can only use one modifier per class, method or attribute. As a rule of thumb, you should always use the most restrictive modifier that still allows you to implement your business logic.

What are different access modifiers?

1. Private: It can only be accessed within your class. Subclasses or any other classes within the same or a different package can't access this attribute or method.

2. No modifier: It can be accessed within your same class and from all classes within the same package. That's why it's often called package-private.

3. Protected: It can be accessed within your class, by all classes within the same package, and by all subclasses within the same or other packages.

4. Public: It can be accessed within your current class and by all other classes.

Modifier	Class	Package	Subclass	Other Classes
Private	Yes	No	No	No
No modifier	Yes	Yes	No	No
Protected	Yes	Yes	Yes	No
Public	Yes	Yes	Yes	Yes

https://www.javatpoint.com/encapsulation

https://www.javatpoint.com/access-modifiers

ABSTRACTION

What is Abstraction?

Handle complexity by hiding unnecessary details from the user. That enables the user to implement more complex logic on top of the provided abstraction without understanding or even thinking about all the hidden complexity.

Example: TV, Laptop, Careem Application etc

.

How abstraction can be achieved?

Abstraction is achieved by using interface and abstract class. Interface give 100% abstraction and abstract class give 0-100%.

ABSTRACT CLASS

An abstract class is a class that contains abstract methods. It is an incomplete class and you cannot create instance of abstract class. If you want to use it, you need to make it complete or concrete by extending it.

```
abstract class TV{
public abstract void changeVolume();
public abstract void changeChannel();
public void Range(){
       System.out.println("This is my range"); }
}

class Samsung extends TV{
public void changeVolume(){
       System.out.println("Volume in Samsung"); }
public void changeChannel(){
       System.out.println("Channel in Samsung"); }
}
Example: Car, TV etc.
```

What is a concrete class?

A class is called concrete if it does not contain any abstract method and implements all abstract method inherited from abstract class or interface it has implemented or extended.

What is an abstract method?

A method that is declared as abstract and does not have implementation is known as abstract method. An abstract method can only be in abstract class.

INTERFACE

What is an Interface?

It lists the method that needs to be implemented by a class. Defines what a class should do, not how to do it.

```
interface Car{
void turnLeft();
void turnRight();
void changeOil();
void changeGear();
}
```

Why do we need interfaces when abstract class can do the functionality of interfaces?

We can extend only one Class but can implement multiple interfaces. When your class has to extend another concrete class you cannot use Abstract class, that time you have to use an interface

https://www.javatpoint.com/difference-between-abstract-class-and-interface

https://www.javatpoint.com/interface-in-java

https://www.javatpoint.com/abstract-class-in-java

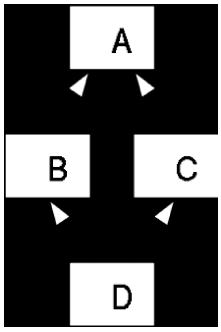
INHERITANCE

What is inheritance?
Passing down traits or characteristics from a parent to their child class.

What are different types of inheritance?

- Single Inheritance: Single Inheritance is when a class inherits only one base class.
- Multiple Inheritance: Multiple Inheritance is when a class inherits more than one base class.
- Multilevel Inheritance: Multilevel Inheritance is when a class inherits from a derived class making that derived class a base class for a new class.

What is problem with Inheritance?
The "diamond problem" (sometimes referred to as the "deadly diamond of death") is an ambiguity that arises when two classes B and C inherit from A, and class D inherits from both B and C. If there is a method in A that B and C have overridden, and D does not override it, then which version of the method does D inherit.



https://www.javatpoint.com/inheritance-in-java

https://www.javatpoint.com/aggregation-in-java

https://stackoverflow.com/questions/269496/inheritance-vs-aggregation

POLYMORPHISM

What is Polymorphism?
Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object. Any Java object that can pass more than one IS-A test is considered to be polymorphic.

```
class Animal{}
interface Vegtarian{}
class Cow extends Animal implements Vegtarian{}

Cow c = new Cow();
Animal a = c;
Vegtarian v = c;
Object o = c;
```

What are different types of Polymorphism?
1. Method Overloading (Compile time Polymorphism, Static Binding, Early Binding)
2. Method Overriding (Run time Polymorphism, Dynamic Binding, Late Binding, Dynamic Dispatch)

- **Method Overloading**: If a class has multiple methods having same name but different in parameters, it is known as Method Overloading. Change the number of arguments or change the data-type. (Within the same class)
- **Method Overriding:** Method Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. (Between base-class and child-class)

Can we overload Main Method?
Yes it is perfectly fine to have another main method in same class with different signature.

Can we override Main Method?
No because main is a static method and static method cannot be overridden in Java.

Why static methods are not overridden?
Cause overridden concept only applies to objects. If you declare static method with the same name & signature in the child class then the static method of the super class will be hidden that is called as method hiding.

What is virtual method?
A virtual method is a method that can be redefined in derived classes.

What is binding?
Binding is the linking between method call and method definition.

What is static binding?
Binding which can be resolved at the compile time by the compiler. Binding happens before a program actually runs. It is also called as Early Binding. Example: Method Overloading

What is dynamic binding?
Binding which cannot be resolved at the compile time by the compiler. Binding happens during runtime. It is also called as Late Binding. Example: Method Overriding

What is the difference between abstraction and encapsulation?
Encapsulation is hiding the internal details of the object while abstraction is showing the essential features and hiding the internal complexity.

https://www.javatpoint.com/runtime-polymorphism-in-java

https://www.javatpoint.com/method-overriding-in-java

https://www.javatpoint.com/access-modifiers#accessoverriding

https://www.javatpoint.com/method-overloading-vs-method-overriding-in-java

https://www.javatpoint.com/super-keyword

JAVA

Difference between Primitive and Non-Primitive Data Types?

Primitive Data Types A primitive data type specifies the size and type of variable values, and it has no additional methods.

Non Primitive Data Types Non-primitive data types are called reference types because they refer to objects.

- Non-primitive types can be used to call methods to perform certain operations, while primitive types cannot.
- A primitive type has always a value, while non-primitive types can be null.
- The size of a primitive type depends on the data type, while non-primitive types have all the same size.

https://www.w3schools.com/java/java_data_types.asp

What is Recursion?
Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.

```
public class Main {
  public static void main(String[] args) {
    int result = sum(10);
    System.out.println(result);
  }
  public static int sum(int k) {
    if (k > 0) {
      return k + sum(k - 1);
    } else {
      return 0;
    }
  }
}
```

https://www.w3schools.com/java/java_recursion.asp

https://www.javatpoint.com/recursion-in-java (for factorial and Fibonacci program)

Difference between Procedural and Object Oriented Programming?
Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

How many types of Modifiers in Java?

Access Modifiers are the modifiers which are used to restrict the visibility of a class, field, method, and constructor.
Non-Access Modifiers: These types of modifiers are used to control a variety of things, such as inheritance capabilities, whether a method can be overridden in a subclass, etc.

https://stackabuse.com/non-access-modifiers-in-java/

Can we have a private constructor in Java?
Yes, we can have a private constructor in Java.

- The private constructor is used when you do not want to create the object of that class.
- We cannot create a subclass of that class.
- It is also used in singleton design pattern, Factory method design pattern. A singleton class is one which limits the number of objects creation to one. Using private constructor we can ensure that no more than one object can be created at a time.

https://www.tutorialspoint.com/can-we-declare-a-constructor-as-private-in-java

Which modifiers can be used with a class?
Public, abstract and final

Can we declare an abstract method as private?
No, an abstract method cannot be private. They must be declared as public, protected or default so that they can be further modified.

Can a method or a class be final and abstract at the same time?
No, it is not possible. A class or a method cannot be final or abstract at the same time because the final method or final class cannot be further modified whereas an abstract class or an abstract method must be modified further.

What are non-access modifiers in Java?
There are four non-access modifiers in Java. They are as follows:

- **Static**: A variable or a method can be accessed without requiring an instantiation of the class to which it belongs. However it can be changed but will changed for every object and that class.
- **Final:** Final is a keyword which is used to restrict the users. In other words, It is used to restrict further modification of a class, field or method. If a class is declared as ‘final’, the class cannot be sub-classed.
- **Abstract:** Abstract is a keyword that is used with a class or a method. An abstract class or abstract method is used for further modification. If a class is declared as ‘abstract’, the class cannot be instantiated.

- **Synchronization:** Synchronization is the capability to control the access of multiple threads to any shared resource. Synchronization is better option where we want to allow only one thread to access the shared resource.

What is the default access modifier for Interface and Enumeration?
The public is the default access modifier. No other access modifier is allowed for them.

Explain visibility control in Java?
Visibility control in Java is implemented by the access modifier.

What are enums?
An enum is a special "class" that represents a group of constants (unchangeable variables, like final variables). To create an enum, use the enum keyword, and separate the constants with a comma. Note that they should be in uppercase letters

```
enum Level {
  LOW,
  MEDIUM,
  HIGH
}
```

https://www.javatpoint.com/enum-in-java

COLLECTIONS

Difference between an Array and ArrayList?
The difference between a built-in array and an ArrayList in Java, is that the size of an array cannot be modified https://www.javatpoint.com/difference-between-array-and-arraylist

Difference between an ArrayList and LinkedList?

Sr. No.	Key	ArrayList	LinkedList
1	Internal Implementation	ArrayList internally uses a dynamic array to store its elements.	LinkedList uses Doubly Linked List to store its elements.
2	Manipulation	ArrayList is slow as array manipulation is slower.	LinkedList is faster being node based as not much bit shifting required.
3	Implementation	ArrayList implements only List.	LinkedList implements List as well as Queue. It can acts as a queue as well.
4	Access	ArrayList is faster in storing and accessing data.	LinkedList is faster in manipulation of data.

https://www.javatpoint.com/java-arraylist

https://www.javatpoint.com/java-linkedlist

https://www.javatpoint.com/difference-between-arraylist-and-linkedlist

Difference between an Iterator and ListIterator?

BASIS FOR COMPARISON	ITERATOR	LISTITERATOR
Basic	Iterator can traverse the elements in a collection only in forward direction.	ListIterator can traverse the elements in a collection in forward as well as the backwards direction.
Add	Iterator is unable to add elements to a collection.	ListIteror can add elements to a collection.
Modify	Iterator can not modify the elements in a collection.	ListIterator can modify the elements in a collection using set().
Traverse	Iterator can traverse Map, List and Set.	ListIterator can traverse List objects only.
Index	Iterator has no method to obtain an index of the element in a collection.	Using ListIterator, you can obtain an index of the element in a collection.

Difference between List and Set and Map?

- **Duplicate Objects:** The main difference between List and Set interface in Java is that List allows duplicates while Set doesn't allow duplicates. While a Map holds two objects per Entry e.g. a key and a value and It may contain duplicate values but keys are always unique.
- **Order:** List is an ordered collection, List maintains insertion order or element. Set is an unordered collection, you get no guarantee on which order element will be stored.
- **Null Element:** The list allows null elements and you can have many null objects in a List because it also allowed duplicates. Set just allow one null element as there is no duplicate permitted while in Map you can have null values and at most one null key.

List: ArrayList, LinkedList, Vector
Set: HashSet, LinkedHashSet, TreeSet
Map: HashMap, LinkedHashMap, HashTable, TreeMap

https://www.geeksforgeeks.org/difference-between-list-set-and-map-in-java/

Difference between HashMap and TreeMap?

- **HashMap** maintains no order, but TreeMap maintains ascending order.
- **HashMap** is implemented by hash table whereas TreeMap is implemented by a Tree structure.
- **HashMap can** be sorted by Key or value whereas TreeMap can be sorted by Key.

https://www.geeksforgeeks.org/differences-treemap-hashmap-linkedhashmap-java/

Difference between HashSet and TreeSet?

- **HashSet** gives better performance (faster) than TreeSet for the operations like add, remove, contains, size etc.
- **HashSet** does not maintain any order of elements while TreeSet elements are sorted in ascending order by default.
- If you want a sorted Set then it is better to add elements to HashSet and then convert it into TreeSet rather than creating a TreeSet and adding elements to it.

https://www.javatpoint.com/working-of-hashset-in-java

https://www.javatpoint.com/java-linkedhashset

https://www.javatpoint.com/java-treeset

Difference between HashSet and HashMap?

- HashSet contains only values whereas HashMap includes the entry (key, value).
- HashSet can be iterated, but HashMap needs to convert into Set to be iterated.
- HashSet implements Set interface whereas HashMap implements the Map interface
- HashSet cannot have any duplicate value whereas HashMap can contain duplicate values with unique keys.

https://www.javatpoint.com/working-of-hashmap-in-java
https://www.javatpoint.com/working-of-hashset-in-java

Difference between HashMap and HashTable?

- HashMap is not thread safe and unsynchronized whereas HashTable is thread safe and synchronized.
- HashMap is faster because it's unsynchronized.
- HashMap works with single thread whereas HashTable works with Multiple Threads.
- HashMap allows one null key whereas HashTable doesn't allow any null key.

https://www.javatpoint.com/difference-between-hashmap-and-hashtable

What is hashCode() method?

- The hashCode method is an inbuilt method that returns the integer hashed value of the input value.
- If two or more objects are equal according to the equals method, then their hashes should be equal too.
- If two or more objects are not equal according to the equals method, then their hashes can be equal or unequal.

What is equals() method?

The equals method is used to check whether two objects are the same or not. It needs to be overridden if we want to check the objects based on the property.
For example, Employee is a class that has 3 data members: id, name, and salary. However, we want to check the equality of employee object by the salary. Then, we need to override the equals() method.

Sort the numbers in ascending order?

```
for(int i=0; i<a.length; i++){
    for(int j=i+1; j<a.length; j++){
        if(a[i]>a[j]){
            int temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
}
```

Find the second largest number?

```
int a[] = {10,2,16,3,11,29,33,2,4,50,11};
int max = a[0];
int smax = a[0];

for(int i=1; i<a.length; i++){
    if(a[i]>max){
        smax = max;
        max = a[i];
    }
}
```

Find the missing number?

```java
int a[] = {5,3,2,1,7,6};
int N = 7;
int idealSum = (N * (N+1)) / 2;
int sum = 0;

for(int num : a){
    sum+=num;
}

System.out.println("Missing Number is " + (idealSum - sum));
```

NOTE: Have to do other coding practices too and also add here the recursive function codes.

DBMS / MySQL

What is Stored Procedure?

Stored Procedures are collection of SQL Statements that are stored in the Database. It is a prepared SQL code that you can save so that the code can be used over again and again.

```
DELIMITER $$

CREATE PROCEDURE getCityName(IN CITY_ID INT, OUT CITY_NAME VARCHAR(255))
BEGIN
   SELECT NAME INTO CITY_NAME
   FROM city
   WHERE ID = CITY_ID;
END $$

DELIMITER ;
```

Difference between Function and Stored Procedure?

Stored Procedure	**Function**
Supports both input and output parameters.	Supports only input parameter.
There is no way to call procedures from the Select and Where statements.	You can call function from a Select statement
Not necessary to return any value.	Must return a value.
Can return multiple values.	Returns only one value.
All database operations are allowed like insert, update delete etc.	Only Select is allowed
Stored procedures can call the function.	The function cannot call a stored procedure.

https://bestinterviewquestion.medium.com/difference-between-stored-procedure-and-function-in-mysql-52f845d70b05

https://www.mysqltutorial.org/mysql-stored-function

Function example?

Example 1

```
CREATE FUNCTION expiry_date (expiry_month VARCHAR(100), expiry_year VARCHAR(100))
RETURNS VARCHAR(100) DETERMINISTIC
RETURN CONCAT(expiry_month, '/', expiry_year);
```

Example 2:

```
DELIMITER $$
CREATE FUNCTION get_card_type(card_type_id INT)
RETURNS VARCHAR(100) DETERMINISTIC
BEGIN
    DECLARE card_type VARCHAR(100);
    IF card_type_id = 1 THEN
        SET card_type = 'AMEX';
    ELSEIF card_type_id = 2 THEN
        SET card_type = 'DISCOVER';
    ELSEIF card_type_id = 3 THEN
        SET card_type = 'MASTERCARD';
    ELSEIF card_type_id = 4 THEN
        SET card_type = 'VISA';
    ELSEIF card_type_id = 4 THEN
        SET card_type = 'DINERS';
    END IF;
    RETURN (card_type);
END$$
DELIMITER ;
```

What are Triggers?

A trigger is a set of actions that are run automatically when a specified operation (INSERT, UPDATE, or DELETE statement) is performed on a specified table. Triggers are useful for tasks such as enforcing business rules, validating input data, and keeping an audit trail.

```
DROP TRIGGER IF EXISTS tr_ins_char;
DROP TRIGGER IF EXISTS tr_up_char;

CREATE TRIGGER tr_ins_char
BEFORE INSERT ON characters
FOR EACH ROW
SET NEW.character_name = UPPER(NEW.character_name);

-- OLD

CREATE TRIGGER tr_up_char
BEFORE UPDATE ON characters
FOR EACH ROW
 SET NEW.character_name =
LOWER(NEW.character_name);
```

https://www.mysqltutorial.org/create-the-first-trigger-in-mysql.aspx/

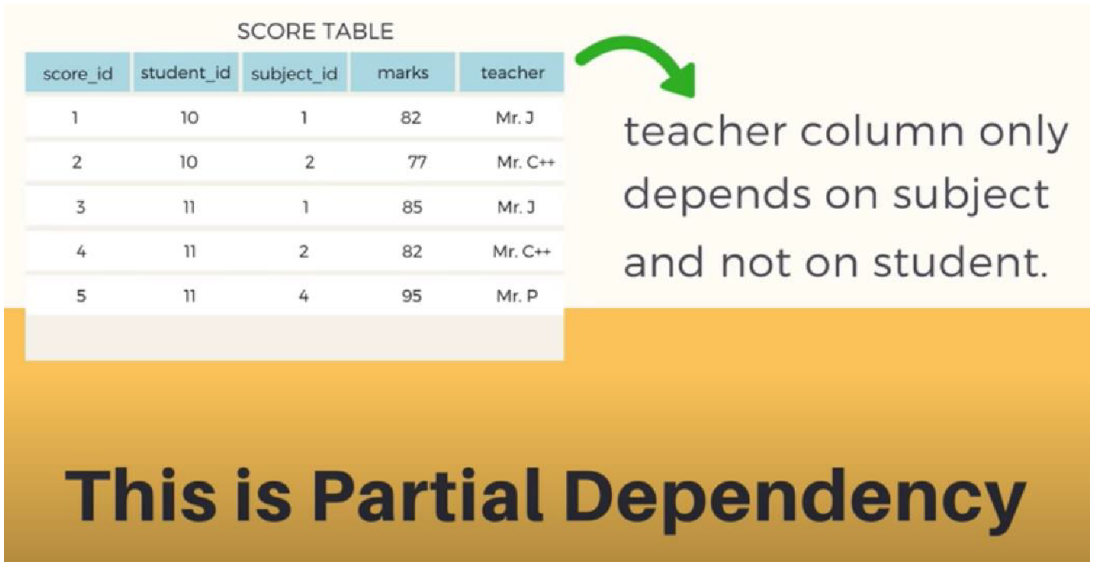
What is Normalization?
Normalization is the technique of organizing the data into multiple table to minimize data Redundancy and inconsistency. Data Redundancy is repetition of similar data at multiple places.

Issues due to Data Redundancy?
1. Insertion Anomaly: Inserting redundant data for every new row.
2. Deletion Anomaly: Loss of a related dataset when some other dataset is deleted.
3. Updation Anomaly

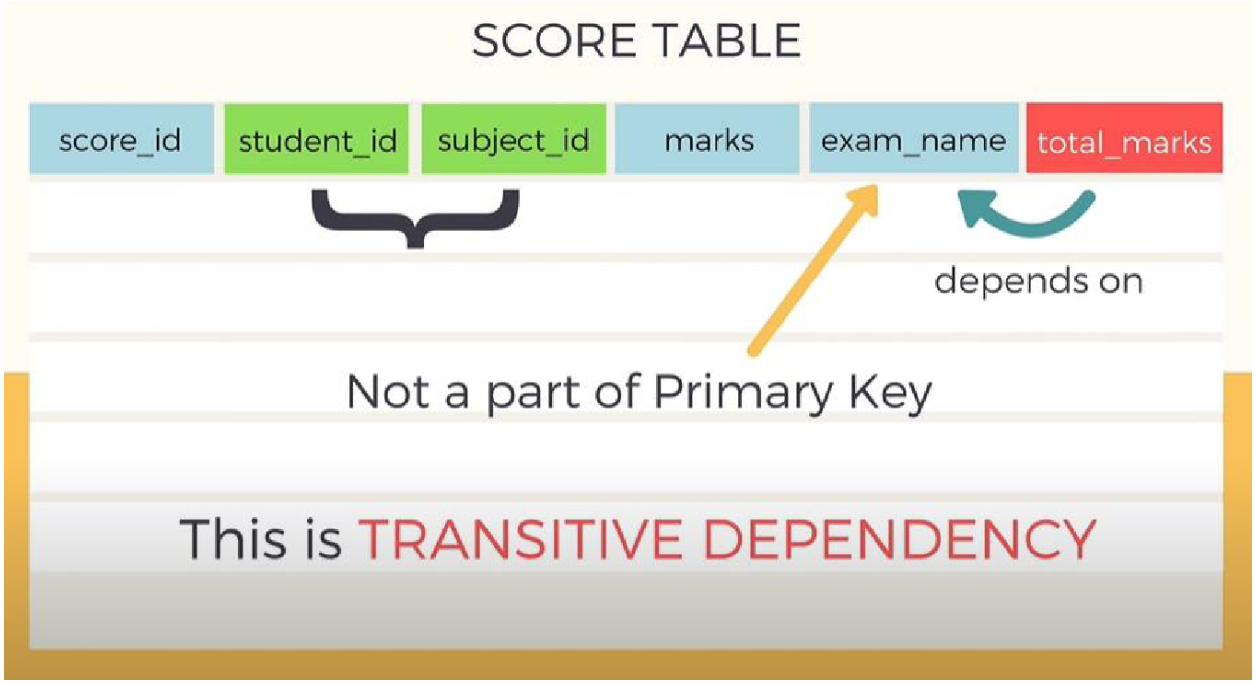
Advantages of Normalization?
- Minimization of redundant data.
- Data anomalies are resolved.
- Overall size of the database is reduced.
- Fewer indexes per table ensures faster execution of commands.
- Makes data more reusable and meaningful.

What are the types of Normalization?
1NF:
1. Each column should contain atomic values.
2. Column should contain values that are of the same type.
3. Each column should have a unique name

2NF: It should not have partial dependency.
Partial Dependency is when an attribute depends on part of the primary key, not the complete primary key (Composite Key)



3NF: It should not have transitive dependency.
Transitive dependency is when an attribute depends on a non-primary attribute.



https://www.w3schools.in/dbms/database-normalization/

https://www.javatpoint.com/dbms-normalization

JAVA 8

New features in Java 8?

- forEach() method in Iterable Interface.
- Default and static methods in Interface.
- Functional Interface and Lambda Expressions.
- Java Stream API
- Java Time API

Changes in Interface in Java 8?

- Before java 8, interface in Java can only have abstract methods. All the methods of interfaces are public & abstract by default.
- Java 8 allows the interfaces to have default and static methods.
- The reason we have default methods in interfaces is to allow the developers to add new methods to the interfaces without affecting the classes that implements these interfaces.
- Defender or Default methods are introduced to maintain backward compatibility

For example, if several classes such as A, B, C and D implements an interface TestInterface then if we add a new method to the TestInterface, we have to change the code in all the classes (A, B, C and D) that implements this interface.
In this example we have only four classes that implements the interface which we want to change but imagine if there are hundreds of classes implementing an interface then it would be almost impossible to change the code in all those classes.
This is why in java 8, we have a new concept default methods. These methods can be added to any existing interface and we do not need to implement these methods in the implementation classes mandatorily, thus we can add these default methods to existing interfaces without breaking the code.

Static methods in interfaces are similar to the default methods except that we cannot override these methods in the classes that implements these interfaces.

```java
public interface TestInterface {

    void existingMethod();

    default void newMethod(){
        System.out.println("New Method");
    }

    static void staticMethod(){
        System.out.println("Static Method");
    }

}
```

```java
public class ClassA implements TestInterface {

    @Override
    public void existingMethod() {
        System.out.println("Existing Method");

    }

}
```

https://dzone.com/articles/interface-default-methods-java

https://www.tutorialspoint.com/default-method-vs-static-method-in-an-interface-in-java

What to do if problem with Multiple Inheritance arises in Interface in Java 8?
Then simply need to provide the implementation of that method in the implementation class

How forEach() is implemented in Java 8?
https://www.javatpoint.com/java-8-foreach

What is Lambda Expression?
Lambda expression is a short block of code which takes in parameters and returns a value. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method.

```
ArrayList<Double> amountList = new ArrayList<>();
amountList.add(50.5);
amountList.add(1000.0);
amountList.add(255.0);

amountList.forEach(
    (i) -> System.out.println("Number: " + i)
);
```

https://www.javatpoint.com/java-lambda-expressions

What is a functional interface?
A functional interface is an interface that contains only one abstract method. From Java 8 onwards, lambda expressions can be used to represent the instance of a functional interface. A functional interface can have any number of default methods.

```
@FunctionalInterface
public interface TestInterface {

    String getLast4Digits(String cardNumber);

}
```

```
TestInterface obj = (cardNumber) -> cardNumber.substring(cardNumber.length()-4);
obj.getLast4Digits("5555555555554444");
```

Java 8 Stream API and Collectors?
https://www.javatpoint.com/java-8-stream

https://www.javatpoint.com/java-8-collectors

https://www.tutorialspoint.com/java8/java8_streams.htm

Difference between Aggregation and Composition?
Association refers to the relationship between multiple objects. It refers to how objects are related to each other and how they are using each other's functionality. Composition and aggregation are two types of association.

Aggregation is a weak association. An association is said to be aggregation if both Objects can exist independently. For example, a Team object and a Player object. The team contains multiple players but a player can exist without a team.

```
public class Person {

    private final Address address;

    public Person(String firstName, String lastName, Address address){
        this.address = address;
    }

}
```

Composition is the strong type of association. An association is said to composition if one object cannot exist without the other object. Consider the case of a Human having a heart. Here Human object contains the heart and heart cannot exist without Human.

```
public class Car {

    private final Engine engine;

    public Car(){
        this.engine = new Engine();
    }

}
```

https://stackoverflow.com/questions/11881552/implementation-difference-between-aggregation-and-composition-in-java

What are different String Operations?
charAt() method returns a char value at the given index number.

compareTo() method compares the given string with current string lexicographically. It returns positive number, negative number or 0.

contains() method searches the sequence of characters in this string. It returns true if sequence of char values are found in this string otherwise returns false.

endsWith() method checks if this string ends with given suffix. It returns true if this string ends with given suffix else returns false.

indexOf() method returns index of given character value or substring. If it is not found, it returns -1. The index counter starts from zero.

substring() method returns a part of the string.

trim() method eliminates leading and trailing spaces.

toCharArray() method converts this string into character array. It returns a newly created character array.

toLowerCase() method returns the string in lowercase letter.

toUpperCase() method returns the string in uppercase letter.

replace() method returns a string replacing all the old char to new char.

https://www.javatpoint.com/java-string

WEBSERVICES

What are Web Services and its types?
Web service is a collection of standards or protocols for exchanging data between two devices or applications. There are mainly two types of web services.

SOAP web service	REST web service
SOAP is a protocol	REST is an architectural style
SOAP stands for Simple Object Access Protocol	REST stands for Representational State Transfer
SOAP uses services interfaces to expose the business logic.	REST uses URI to expose the business logic.
SOAP requires more bandwidth and resources.	REST doesn't need much bandwidth and resource.
SOAP only works with XML formats.	REST work with plain text, XML, HTML and JSON.
SOAP cannot make use of REST.	REST can make use of SOAP.

What is REST?
REST is the acronym for Representational State Transfer. REST is an architectural style for developing applications that can be accessed over the network.

REST is a stateless client-server architecture where web services are resources and can be identified by their URIs.

Client applications can use HTTP GET/POST methods to invoke Restful web services. REST doesn't specify any specific protocol to use, but in almost all cases it's used over HTTP/HTTPS.

Advantages of REST?

- Learning curve is easy since it works on HTTP protocol
- Supports multiple technologies for data transfer such as text, XML, JSON, image etc.
- No contract defined between server and client, so loosely coupled implementation.
- REST is a lightweight protocol
- REST methods can be tested easily over browser.

What is purpose of a URI in REST based web services?
URI stands for Uniform Resource Identifier. Each resource in REST architecture is identified by its URI. Purpose of an URI is to locate a resource(s) on the server hosting the web service.

What are the HTTP methods supported by REST?
GET, POST, PUT DELETE

What is the purpose of HTTP Status Code?
HTTP Status code are standard codes and refers to predefined status of task done at server. For example, HTTP Status 404 states that requested resource is not present on server.

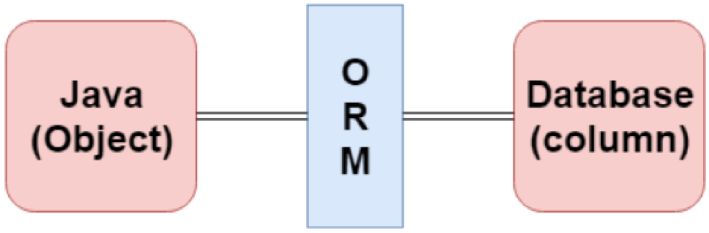
- 200 - OK, shows success.
- 400 - BAD REQUEST, states that invalid input is provided e.g. validation error, missing data.
- 401 - FORBIDDEN, states that user is not having access to method being used for example, delete access without admin rights.
- 404 - NOT FOUND, states that method is not available.
- 500 - INTERNAL SERVER ERROR, states that server has thrown some exception while executing the method.

https://www.studytonight.com/rest-web-service/types-of-webservices

HIBERNATE FRAMEWORK

What is ORM?

Object Relational Mapping (ORM) is a functionality which is used to develop and maintain a relationship between an object and relational database by mapping an object state to database column. It is capable to handle various database operations easily such as inserting, updating, deleting etc.



What is Hibernate and why we use it?

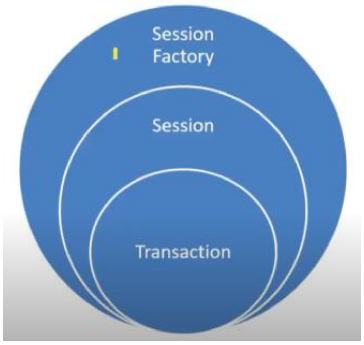
Hibernate is an ORM framework / tool used to map Java objects and the database tables.
It provides JPA Implementation hence we can use JPA annotations as well as xml configurations to achieve the mapping.

Why Hibernate?

- **Eliminates Boiler-Code or Repeat Code:** Hibernate eliminates all the boiler-code that comes with the JDBC.
- **Supports HQL:** It supports HQL which is more Object Oriented.
- **Supports Caching**: Hibernate supports caching for better performance.
- **Object Mapping**: In JDBC, you need to write code to map the object model's data representation to a relational model and its corresponding schema. Hibernate itself maps java classes to database tables using xml or by using annotations.

Important Interface used in the Hibernate?

- **SessionFactory (org.hibernate.SessionFactory)** – Instance of this is used to retrieve Session objects for database operations. We need to initialize that once and can cache it to reuse it again and again. Its like one SessionFactory object per database connection. Like 1 for mysql, 1 for oracle.
- **Session (org.hibernate.Session)** – Its factory for transaction, it's used for connecting application with persistant store like hibernate framework / DB. It is used to get a physical connection with the database. It also provides methods for CRUD operations.
- **Transaction (org.hibernate.Transaction).** – This specifies single / atomic units of work



Code for Hibernate CRUD?

SAVE

```
public Integer savePaymentRequest(PaymentRequestDTO paymentRequest, SessionFactory sessionFactory){
    Session session = sessionFactory.openSession();
    Transaction tx = null;
    Integer requestId = null;

    try {

        tx = session.beginTransaction();
        requestId = (Integer) session.save(paymentRequest);
        tx.commit();

    } catch (HibernateException ex) {
        if(tx!=null){
            tx.rollback();
        }
    } finally {
        session.close();
    }

    return requestId;
}
```

READ

```
public ArrayList<PaymentRequestDTO> getPaymentRequests(SessionFactory sessionFactory){
    Session session = sessionFactory.openSession();
    ArrayList<PaymentRequestDTO> paymentRequestList = null;
    Transaction tx = null;
    try {

        List list = session.createCriteria(PaymentRequestDTO.class).setCacheable(true).list();
        paymentRequestList = (ArrayList<PaymentRequestDTO>) list;
        tx.commit();

    } catch(HibernateException ex){
        if(tx!=null){
            tx.rollback();
        }
    } finally {
        session.close();
    }
    return paymentRequestList;
}
```

UPDATE

```
public void updatePaymentRequest(PaymentRequestDTO paymentRequest, SessionFactory sessionFactory){
    Session session = sessionFactory.openSession();
    Transaction tx = null;

    try {

        tx = session.beginTransaction();
        session.update(paymentRequest);
        tx.commit();

    } catch (HibernateException ex) {
        if(tx!=null){
            tx.rollback();
        }
    } finally {
        session.close();
    }
}
```

DELETE

```
public void deletePaymentRequest(Integer requestId, SessionFactory sessionFactory){
    Session session = sessionFactory.openSession();
    Transaction tx = null;

    try {

        tx = session.beginTransaction();
        PaymentRequestDTO paymentRequest = session.get(PaymentRequestDTO.class, requestId);
        session.delete(paymentRequest);
        tx.commit();

    } catch (HibernateException ex) {
        if(tx!=null){
            tx.rollback();
        }
    } finally {
        session.close();
    }
}
```

```java
public ArrayList<PaymentRequestDTO> getAllPaymentRequestbyReferenceTypeId(Integer referenceTypeId, Session session) {
    ArrayList<PaymentRequestDTO> paymentRequestList = null;
    boolean isNew = false;
    Transaction tx = null;

    try {
        if (session == null) {
            session = HibernateSessionFactory.getSession();
            tx = session.getTransaction();
            // If no active transaction associated with Session, then begin new
            if(tx==null || !tx.isActive()){
                tx = session.beginTransaction();
            }
            isNew = true;
        }

        Criteria paymentRequestCriteria = session.createCriteria(PaymentRequestDTO.class);
        paymentRequestCriteria.add(Restrictions.eq("referenceTypeId", referenceTypeId));

        List list = paymentRequestCriteria.setCacheable(true).list();

        if(list != null && list.size() > 0){
            paymentRequestList = (ArrayList<PaymentRequestDTO>) list;
        }

        if(isNew && tx.isActive() && !HibernateUtil.wasCommitted(tx)){
            tx.commit();
        }

    } catch(HibernateException ex){
        if(isNew){
            tx.rollback();
        }
    }

    return paymentRequestList;
}
```

NOTE: See the HQL example in POAAssociationService.java

https://mkyong.com/hibernate/hibernate-cascade-example-save-update-delete-and-delete-orphan/

https://howtodoinjava.com/hibernate/hibernate-jpa-cascade-types/

https://www.tutorialspoint.com/hibernate/hibernate_query_language.htm

```
<hibernate-mapping>
    <class name="com.sibisoft.northstar.util.pps.eft.dto.CustomerAccountDTO" table="pps_eft_customer_account">

        <id name="customerAccountId" type="java.lang.Integer">
            <column name="customer_account_id" />
            <generator class="identity" />
        </id>

        <property name="accountType" type="java.lang.Integer">
            <column name="account_type" not-null="false" />
        </property>

        <property name="accountHolderName" type="java.lang.String">
            <column name="account_holder_name" not-null="false" />
        </property>

        <property name="ccNumber" type="java.lang.String">
            <column name="cc_number" not-null="false" />
        </property>

        <property name="expiryYear" type="java.lang.String">
            <column name="expiry_year" not-null="false" />
        </property>

        <property name="expiryMonth" type="java.lang.String">
            <column name="expiry_month" not-null="false" />
        </property>

        <!-- One Customer Account can/will be used for multiple Payment Requests -->
        <set name="paymentRequests" lazy="true" inverse="true" cascade="all" order-by="request_id asc">
          <key column="customer_account_id" />
          <one-to-many class="com.sibisoft.northstar.util.pps.eft.dto.PaymentRequestDTO" />
        </set>

    </class>
</hibernate-mapping>
```

```java
@Entity(name = "ForeignKeyAssoEntity")
@Table(name = "Employee", uniqueConstraints = {
@UniqueConstraint(columnNames = "ID"),
@UniqueConstraint(columnNames = "EMAIL") })
public class EmployeeEntity implements Serializable {

    private static final long serialVersionUID = -1798070786993154676L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID", unique = true, nullable = false)
    private Integer employeeId;

    @Column(name = "EMAIL", unique = true, nullable = false, length = 100)
    private String email;

    @Column(name = "FIRST_NAME", unique = false, nullable = false, length = 100)
    private String firstName;

    @Column(name = "LAST_NAME", unique = false, nullable = false, length = 100)
    private String lastName;

    @OneToMany(cascade=CascadeType.ALL)
    @JoinColumn(name="EMPLOYEE_ID")
    private Set<AccountEntity> accounts;

    //Getters and setters
}
```

SPRING

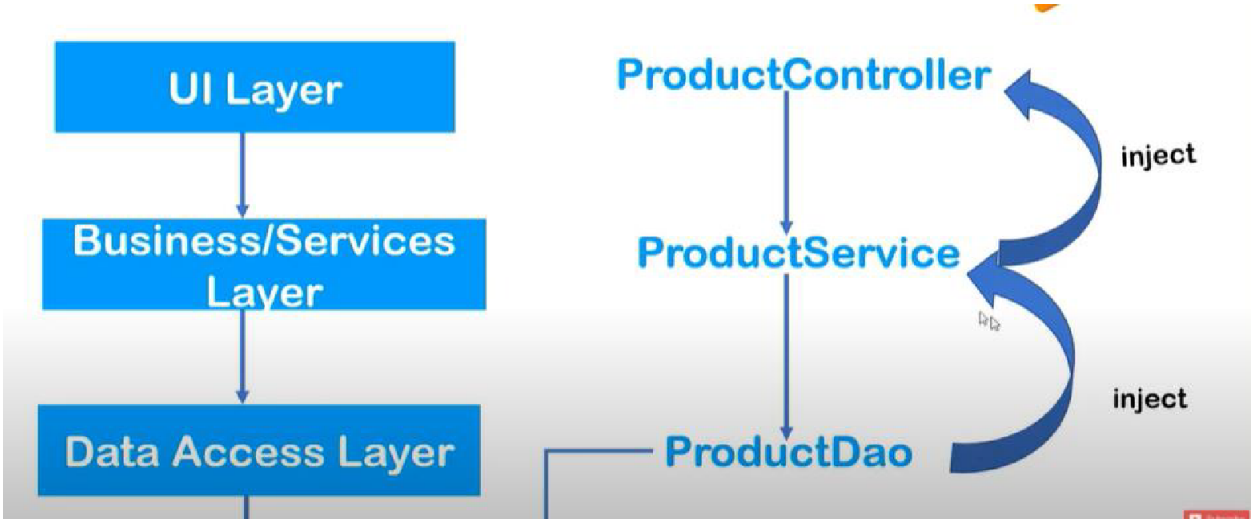
What is MVC?
MVC is an architectural pattern that separates an application into three main logical components.

- Model
- View
- Controller

What is the Spring Framework?

- Spring Framework is an application development framework for development Java Enterprise Applications.
- It is a Dependency Injection Framework to make Java Application loosely coupled.
- It provides various modules that makes the easy development of Java Application.

What is Dependency Injection?
Dependency injection is a programming technique that makes a class independent of its dependencies. It makes Java classes independent of each other and the container frees them from object creation and maintenance. This is called as Inversion of Control (IoC) where the control of objects or portions of a program is transferred to a container or framework.

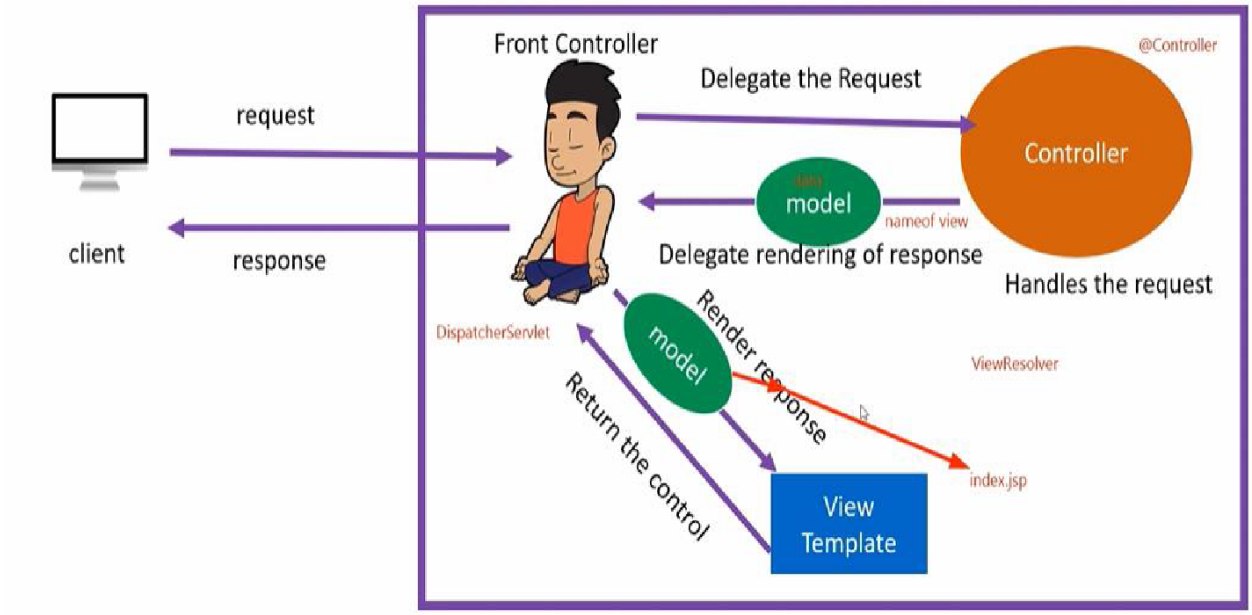


Working of Spring MVC?

Spring MVC framework is request-driven and is designed around a central Servlet that handles all the HTTP requests and responses. The DispatcherServlet, however, does a lot more than just that. It seamlessly integrates with the IoC container and allows you to use each feature of Spring.

On receiving an HTTP request, the DispatcherServlet consults HandlerMapping (these are the configuration files) to call the appropriate Controller.
Then, the controller calls appropriate service methods to set the Model data. It also returns the view name to DispatcherServlet.

DispatcherServlet, with the help of ViewResolver, picks up the defined view for the request. Once the view is finalized, the DispatcherServlet passes the Model data to View - where it is finally rendered on the browser



https://www.javatpoint.com/ioc-container

https://www.javatpoint.com/dependency-injection-in-spring

https://www.javatpoint.com/spring-tutorial-dependency-injection-by-constructor

https://www.tutorialspoint.com/spring/spring_bean_definition.htm

https://www.tutorialspoint.com/spring/spring_bean_scopes.htm

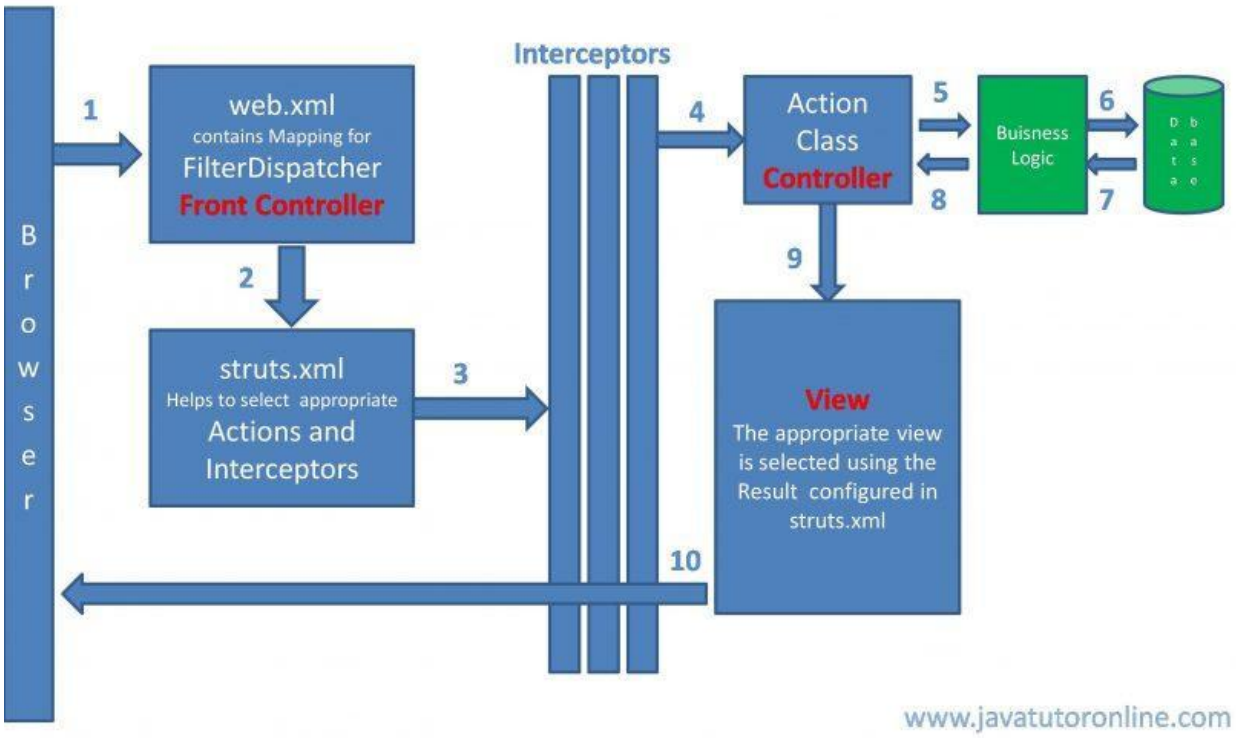
https://www.tutorialspoint.com/spring/spring_bean_life_cycle.htm

https://docs.spring.io/spring-framework/docs/1.2.x/reference/beans.html

https://dzone.com/articles/difference-between-beanfactory-and-applicationcont

STRUTS

1. The request is first sent from the browser to the server. Then the server loads the web.xml and if the request pattern matches then it is forwarded to the FilterDispatcher. In struts 2 the FilterDispatcher is the Front Controller.
2. Based on the request url and it's mapping in the struts.xml the appropriate action class to be executed is decided.
3. Before the Action class is executed the request is passed through the interceptors.
4. The action method of the action class (controller) is executed.
5. The action class calls the business logic function.
6. The Business Logic class works with the database.
7. Bushiness logic class gets the data from the database.
8. Processed data from bushiness logic is sent back to the Action class or the controller.
9. Depending on the result the Controller identifies the view to be rendered.
10. Before the response is generated the interceptors are executed again.



https://www.javatutoronline.com/struts/how-struts2-works/

Struts Interceptors:

https://www.tutorialspoint.com/struts_2/struts_interceptors.htm

https://struts.apache.org/getting-started/introducing-interceptors.html

https://www.javatpoint.com/struts-2-interceptors-tutorial

Indexes in MySQL:

https://www.tutorialspoint.com/mysql/mysql-indexes.htm

https://www.peachpit.com/articles/article.aspx?p=30885&seqNum=9

https://vanseodesign.com/web-design/the-types-of-indexes-you-can-add-to-mysql-tables/

https://severalnines.com/database-blog/guide-mysql-indexes

Interview Questions:

1. https://howtodoinjava.com/java-interview-questions/
2. https://www.bestinterviewquestion.com/mysql-interview-questions
3.