1. What is oops in java?

Ans -

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

2. What is Object in java ?

Ans -

An object is an instance of a class. A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

Object Definitions:

- An object is *a real-world entity*.
- An object is *a runtime entity*.
- The object is *an entity which has state and behavior*.
- The object is *an instance of a class*.

For Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.

3. What is a class in java ?

Ans -

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

- Fields
- Methods
- Constructors
- Blocks
- Nested class and interface

4. Methods in java ?

Ans -

In Java, a method is like a function which is used to expose the behavior of an object.

Advantage of Method

- Code Reusability
- Code Optimization

5. **What are the principle of OOPS in java?**

Ans -

- Encapsulation
- Inheritance
- Abstraction
- Polymorphism

Encapsaulation :-

Encapsulation in Java is a *process of wrapping code and data together into a single unit*

eg. - The **Java Bean** *class is the example of a fully encapsulated class.*

- **Hiding data:** Users will have no idea how classes are being implemented or stored. All that users will know is that values are being passed and initialized.
- **More flexibility:** Enables you to set variables as red or write-only. Examples include: setName(), setAge() or to set variables as write-only then you only need to omit the get methods like getName(), getAge() etc.
- **Easy to reuse:** With encapsulation, it's easy to change and adapt to new requirements

Inheritance -

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system). The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class.

Why use inheritance in java

- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

Method Overriding in Java

If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in Java**.

- Method overriding is used for runtime polymorphism

Rules for Java Method Overriding

1. The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class.
3. There must be an IS-A relationship (inheritance).

```
class Bank{
int getRateOfInterest(){return 0;}
}
//Creating child classes.
class SBI extends Bank{
int getRateOfInterest(){return 8;}
}

class ICICI extends Bank{
int getRateOfInterest(){return 7;}
}
class AXIS extends Bank{
int getRateOfInterest(){return 9;}
}
//Test class to create objects and call the methods
class Test2{
public static void main(String args[]){
SBI s=new SBI();
ICICI i=new ICICI();
AXIS a=new AXIS();
System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());
System.out.println("ICICI Rate of Interest: "+i.getRateOfInterest());
System.out.println("AXIS Rate of Interest: "+a.getRateOfInterest());
```

Can we override static method?

No, a static method cannot be overridden.

Why can we not override static method?

It is because the static method is bound with class whereas instance method is bound with an object. Static belongs to the class area, and an instance belongs to the heap area.

Method Overloading in Java

If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**.

Advantage of method overloading

Method overloading **increases the readability of the program**. This provides flexibility to programmers so that they can call the same method for different types of data. This makes the code look clean. This reduces the execution time because the binding is done in compilation time itself.

Different ways to overload the method

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

1) Method Overloading: changing no. of arguments

```
class Adder{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}
class TestOverloading1{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));
}}
```

2) Method Overloading: changing data type of arguments

```
class Adder{
static int add(int a, int b){return a+b;}
static double add(double a, double b){return a+b;}
}
class TestOverloading2{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(12.3,12.6));
}}
```

Abstraction

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

A class which is declared with the abstract keyword is known as an abstract class in Java.

Abstract class: is a restricted class that cannot be used to create objects.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Ways to achieve Abstraction

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

Points to Remember

- An abstract class must be declared with an abstract keyword.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

Example of abstract class

```
abstract class A{}
```

Abstract Method in Java

A method which is declared as abstract and does not have implementation is known as an abstract method.

Example of abstract method

```
abstract void printStatus();//no method body and abstract
```

Example of Abstract class that has an abstract method

In this example, Bike is an abstract class that contains only one abstract method run. Its implementation is provided by the Honda class.

```
abstract class Bike{
  abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
```

an abstract class can have other methods.

Polymorphism

Polymorphism is an important concept of object-oriented programming. It simply means more than one form.

That is, the same entity (method or operator or object) can perform different operations in different scenarios.

We can achieve polymorphism in Java using the following ways:

1. Method Overriding
2. Method Overloading
3. Operator Overloading

6. Suppose there is a class which doesn't contain any abstract method, Can it also act as a abstract class.

Ans -

Yes it can

A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods.

- An abstract class must be declared with an abstract keyword.

7. What is interface ?

Ans -

An **interface in Java** is a blueprint of a class. It has static constants and abstract methods. It is *a mechanism to achieve abstraction*.

Interfaces can have abstract methods and variables. It cannot have a method body.

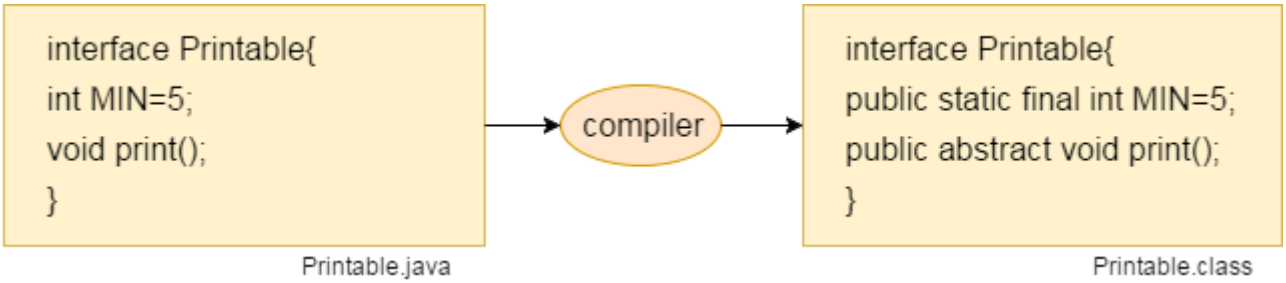
Java Interface also **represents the IS-A relationship**. It cannot be instantiated just like the abstract class.

Why use Java interface?

- It is used to achieve total abstraction.
- By interface, we can support the functionality of multiple inheritance.

It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are **public, static and final by default.** A class that implements an interface must implement all the methods declared in the interface.

In other words, Interface fields are public, static and final by default, and the methods are public and abstract.



8. **Can we implement more than one interface in java**

Ans :-: yes. Java does not support "multiple inheritance" (a class can only inherit from one superclass). However, it can be achieved with interfaces, because **the class can implement multiple interfaces**.

Note: To implement multiple interfaces, separate them with a comma.

9. **What is Static Keyword ?** (https://www.javatpoint.com/static-keyword-in-java)

Ans -

1) Java static variable

- The static variable gets memory only once in the class area at the time of class loading.
- It makes your program **memory efficient** (i.e., it saves memory).
- Static variable will get the memory only once.
- If any object changes the value of the static variable, it will retain its value.

2) Java static method

If you apply static keyword with any method, it is known as static method.

- A static method belongs to the class rather than the object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- A static method can access static data member and can change the value of it.

Auto (TypeScript)

```
//Java Program to demonstrate the use of a static method.
class Student{
     int rollno;
     String name;
     static String college = "ITS";
     //static method to change the value of static variable
     static void change(){
```

```
        college = "BBDIT";
        }
        //constructor to initialize the variable
        Student(int r, String n){
        rollno = r;
        name = n;
        }
        //method to display values
        void display(){System.out.println(rollno+" "+name+" "+college);}
    }
    //Test class to create and display the values of object
    public class TestStaticMethod{
        public static void main(String args[]){
        Student.change();//calling change method
        //creating objects
        Student s1 = new Student(111,"Karan");
        Student s2 = new Student(222,"Aryan");
        Student s3 = new Student(333,"Sonoo");
        //calling display method
        s1.display();
        s2.display();
        s3.display();
        }
    }
```

There are two main restrictions for the static method. They are:

1. The static method can not use non static data member or call non-static method directly.
2. this and super cannot be used in static context.

10. What is superclasses of all classes in java ?

Ans - Object is the superclass of all the classes.

11. Difference between Comparable and Comparator ?

Ans -

Difference between Comparable and Comparator - javatpoint

Comparable	Comparator
1) Comparable provides a **single sorting sequence**. In other words, we can sort the collection on the basis of a single element such as id, name, and price.	The Comparator provides **multiple sorting sequences**. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc.
2) Comparable **affects the original class**,	Comparator **doesn't affect the original**

i.e., the actual class is modified.	**class**, i.e., the actual class is not modified.
3) Comparable provides **compareTo() method** to sort elements.	Comparator provides **compare() method** to sort elements.
4) Comparable is present in **java.lang** package.	A Comparator is present in the **java.util** package.
5) We can sort the list elements of Comparable type by **Collections.sort(List)** method.	We can sort the list elements of Comparator type by **Collections.sort(List, Comparator)** method.

12. What are access specifiers in java?(Access modifiers in java - Javatpoint)

Ans -

Access specifiers define how the members (attributes and methods) of a class can be accessed.

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default. When no access modifier is specified for a class, method, or data member default is used.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

13. What is Final class, Final Variable and Final method, Final Static Variable?

Ans -:

- Final class

The main purpose of Final class is to prevent the class from being subclassed. If a class is marked as final then no class can inherit any feature from the final class.

You cannot extend a final class. If you try it gives you a compile time error.

Example

Auto (Scala)

```
final class Super {
   private int data = 30;
}
public class Sub extends Super{
   public static void main(String args[]){
   }
}
```

- Final Variable

  A variable cannot be modified after it is declared as final. In other words, a final variable is constant. So, a final variable must be initialized and an error occurs if there is any attempt to change the value.

Auto (C++)

```
public class Demo {
   public static void main(String[] args) {
      final double PI = 3.141592653589793;
      System.out.println("The value of pi is: " + PI);
   }
}
```

- Final Method

  We can declare a method as final, once you declare a method final it cannot be overridden. So, you cannot modify a final method from a sub class.

  The main intention of making a method final would be that the content of the method should not be changed by any outsider.

  ### Code

Auto (C#)

```
public class FinalMethodExample {
   public final void display(){
```

```
      System.out.println("Hello welcome to Tutorialspoint");
   }
   public static void main(String args[]){
      new FinalMethodExample().display();
   }
   class Sample extends FinalMethodExample{
      public void display(){
         System.out.println("hi");
      }
   }
}
```

Output

Auto (C#)

```
FinalMethodExample.java:12: error: display() in FinalMethodExample.Sample
cannot override display() in FinalMethodExample
public void display(){
            ^
overridden method is final
1 error
```

- Final Static Variable

Declaring them as Final static will help you to create a **CONSTANT**. Only one copy of variable exists which can't be reinitialize.

14. **Exception Handling in Java ?(**Exception Handling in Java | Java Exceptions - javatpoint**)**

Ans :- Exception handling in java is a mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

Advantage of Exception Handling

The core advantage of exception handling is **to maintain the normal flow of the application**. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions. Let's consider a scenario:

Auto (Bash)

```
statement 1;
statement 2;
statement 3;
statement 4;
statement 5;//exception occurs
```

```
statement 5;//exception occurs
statement 6;
statement 7;
statement 8;
statement 9;
statement 10;
```

Suppose there are 10 statements in a Java program and an exception occurs at statement 5; the rest of the code will not be executed, i.e., statements 6 to 10 will not be executed. However, when we perform exception handling, the rest of the statements will be executed. That is why we use exception handling in Java.

Types of Java Exceptions

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

1. Checked Exception
2. Unchecked Exception
3. Error

Difference between Checked and Unchecked Exceptions

1) Checked Exception

The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.

2) Unchecked Exception

The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

3) Error

Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.

Java provides five keywords that are used to handle the exception. The following table describes each.

Keyword	Description
try	The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by eith catch or finally.
catch	The "catch" block is used to handle the exception. It must be preceded by try bloc which means we can't use catch block alone. It can be followed by finally block late
finally	The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It specifies that there may oc an exception in the method. It doesn't throw an exception. It is always used with method signature.

Java Exception Handling Example

JavaExceptionExample.java

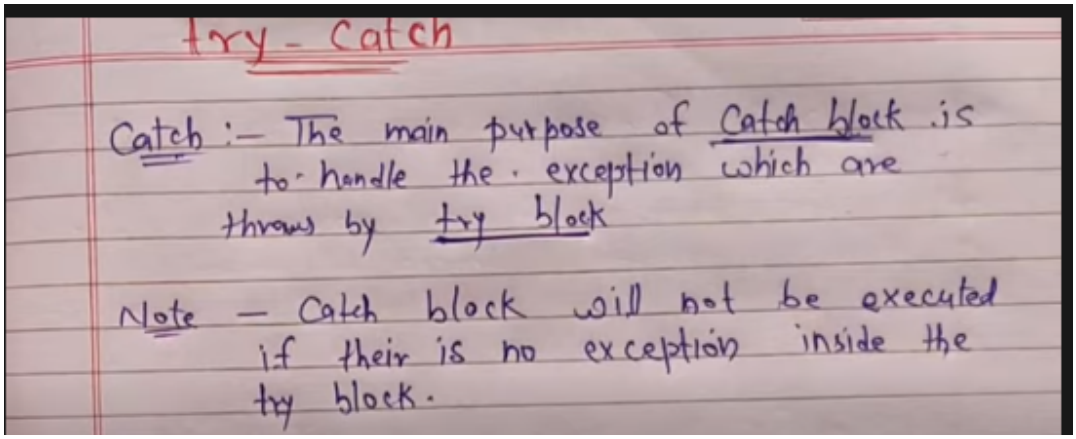
```
public class JavaExceptionExample{
 public static void main(String args[]){
  try{
    //code that may raise exception
    int data=100/0;
  }catch(ArithmeticException e){System.out.println(e);}
  //rest code of the program
  System.out.println("rest of the code...");
 }
}
```

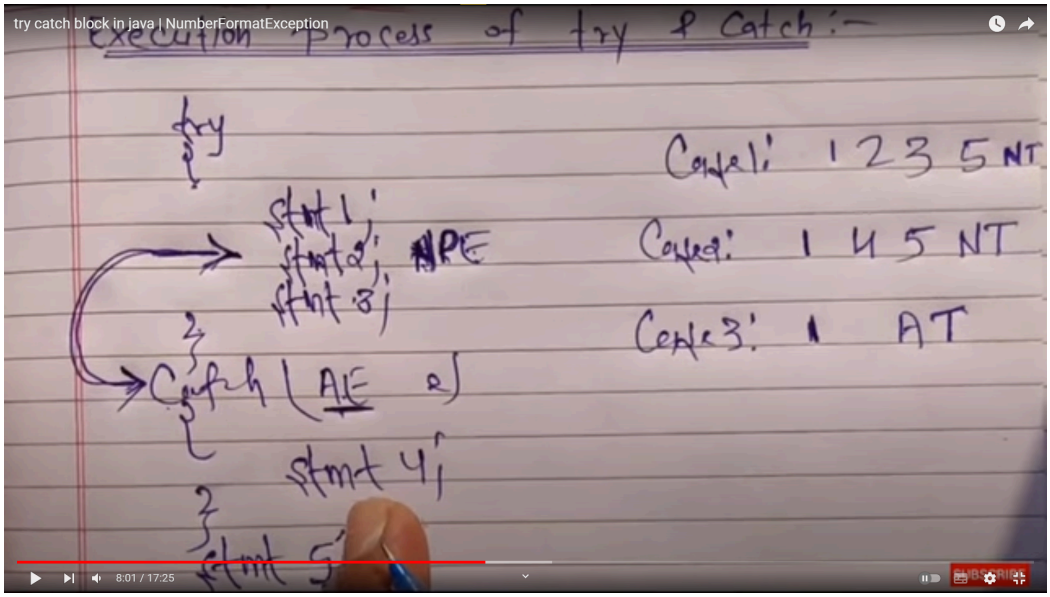
Java Exceptions Index

- Java Try-Catch Block
- Java Multiple Catch Block
- Java Nested Try
- Java Finally Block
- Java Throw Keyword
- Java Exception Propagation
- Java Throws Keyword
- Java Throw vs Throws
- Java Final vs Finally vs Finalize
- Java Exception Handling with Method Overriding
- Java Custom Exceptions

15. **Java try - catch block?When no access modifier is specified for a class, method, or data member**

Ans - https://youtu.be/aAkatKJkjYs





16. **Suppose there is a try block, Can we write finally block without catch block ?**

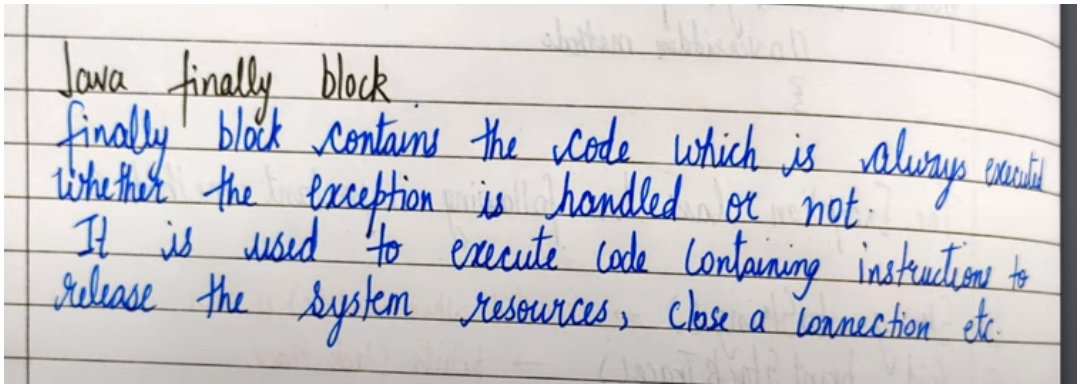
Ans -

Yes, it is not mandatory to use catch block with finally.

> Que - Is catch () mandatory in try catch finally block?
>
> Nope, not at all. **Its not mandatory to put catch after try block, unless and until the try block is followed by a finally block**. Just remember one thing, after try, a catch or a finally or both can work.

17. **Java try-finally block?** Java Finally block - javatpoint

Ans -



18. **Java Garbage collection ?**

Ans -

In java, garbage means unreferenced objects. it is a way to destroy the unused objects.

Advantage of Garbage Collection

- It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.
- It is **automatically done** by the garbage collector(a part of JVM) so we don't need to make extra efforts.

How can an object be unreferenced?

There are many ways :-

> 1) By nulling a reference:
>
> ```
> Employee e=new Employee();
> e=null;
> ```
>
> 2) By assigning a reference to another:
>
> ```
> Employee e1=new Employee();
> ```

```
Employee e2=new Employee();
e1=e2;//now the first object referred by e1 is available for garbage collection
```

3) By anonymous object:

```
new Employee();
```

finalize() method

The finalize() method is invoked each time before the object is garbage collected. This method can be used to perform cleanup processing. This method is defined in Object class as:

```
protected void finalize(){}
```

19. **What is java HashMap ?**

Ans - :

HashMap in Java - javatpoint

Java **HashMap** class implements the Map interface which allows us *to store key and value pair*, where keys should be unique. If you try to insert the duplicate key, it will replace the element of the corresponding key. It is easy to perform operations using the key index like updation, deletion, etc. HashMap class is found in the java.util package.

Spring

20. **Do u know in spring boot which front controller is used?**

Ans -

DispatcherServlet is the front controller in Spring Web MVC. Incoming requests for the HTML file are forwarded to the DispatcherServlet

21. **Do u know how request flow in Spring boot ?**

Ans -

Explanation:

- The Client makes an **HTTP** request(GET, PUT, POST, etc.)
- The HTTP request is forwarded to the **Controller**. The controller maps the request. It processes the handles and calls the server logic.
- The business logic is performed in the **Service layer**. The spring boot performs all the logic over the data of the database which is mapped to the spring boot model class through Java Persistence Library(**JPA**).
- The JSP page is returned as Response from the controller.

Spring boot annotations

Top 12 annotations of spring boot3

Top 12 annotations of spring boots

1. **SpringBootApplication -**

Ans -

SpringBootApplication annotation is a combination of three annotations as follow:-

- ***@EnableAutoConfiguration -***

  The @EnableAutoConfiguration annotation enables Spring Boot to auto-configure the application context. Therefore, it automatically creates and registers beans based on both the included jar files in the class path and the beans defined by us.

- ***@ComponentScan -***

  enables Spring to scan for things like configurations, controllers, services, and other components we define.

- **@Configuration -**

  Spring Configuration annotation **indicates that the class has @Bean definition methods**. So Spring container can process the class and generate Spring Beans to be used in the application

2. **@Autowired -** Spring @Autowired Annotation - GeeksforGeeks

Ans -

The @Autowired annotation marks a Constructor, Setter method, Properties and Config() method as to be autowired that is 'injecting beans'(Objects) at runtime by Spring Dependency Injection mechanism.

The **@SpringBootApplication** which is a combination of @Configuration, @EnableAutoConfiguration and @ComponentScan scans all the components or services and other configuration files included in the base and child packages. This will register them in Spring Context and inject the beans at runtime using @Autowired.

3. **@Component -**

Ans -

@Component is an annotation that allows Spring to automatically detect our custom beans.

In other words, without having to write any explicit code, Spring will:

Scan our application for classes annotated with @Component Instantiate them and inject any specified dependencies into them Inject them wherever needed

21. **In spring where do we keep or all configurations?**

Ans -

application.properties file me rkhte hai config.

22. **What is the difference between @controller and @RestController ?**

22. **What is the difference between @Controller and @RestController ?**

Ans -

https://youtu.be/RrE9qpCxdD4

```
@Controller
@RequestMapping("/api")
public class EmployeeController {

    @RequestMapping(value = "/create", method = RequestMethod.POST)
    public Employee createEmployee() {

    }

    @RequestMapping(value = "/get", method = RequestMethod.GET)
    @ResponseBody
    public Employee getEmployee() {

    }
```

I

With the help of Restcontroller we do not need to explicitly mention @Responsebody with @RequestMapping

we only have to write @RestController and that's it.

@Controller, we need to use @ResponseBody on ery handler method.	In @RestController, we don't need to use @ResponseBody on every handler methoc

23. **What is Spring JPA properties Hibernate dialect ?**

Ans -

Dialect is **a class and a bridge between Java JDBC types and SQL types, which contains mapping between java language data type and database datatype**. Dialect allows Hibernate to generate SQL optimized for a particular relational database

24. **Advantage and Disadvantage of Microoservice**

Ans -

Advantages of Microservices

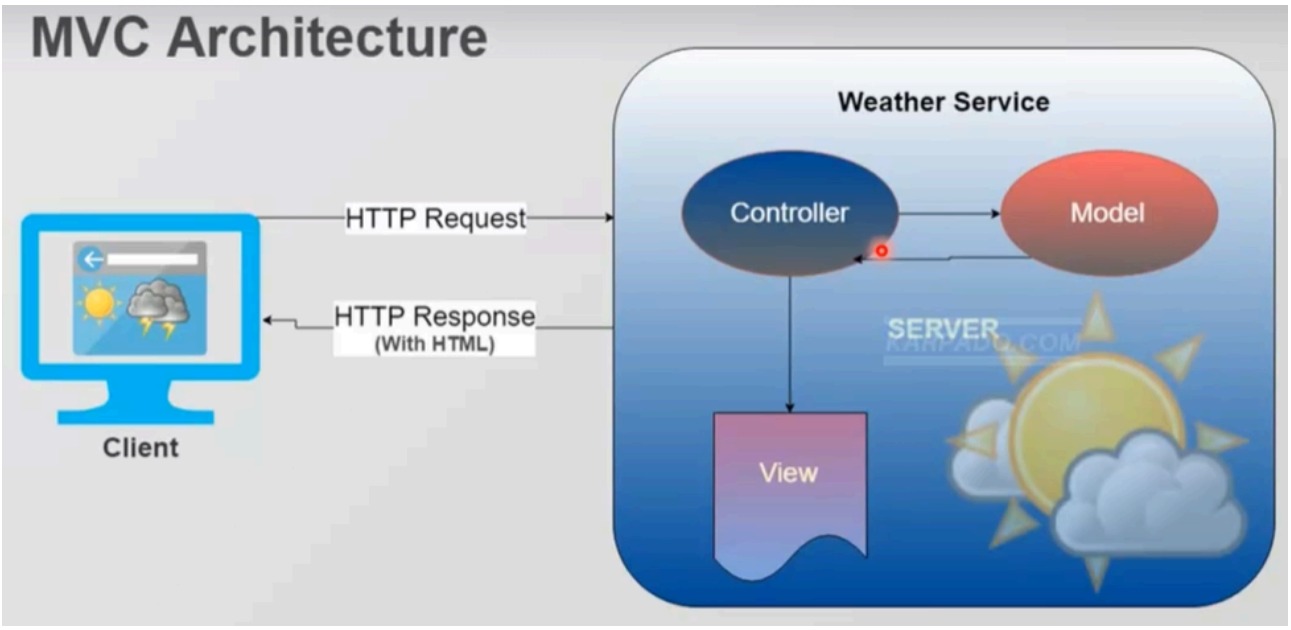
- Microservices are self-contained, independent deployment module.
- The cost of scaling is comparatively less than the monolithic architecture.
- Microservices are independently manageable services. It can enable more and more services as the need arises. It minimizes the impact on existing service.

- It is possible to change or upgrade each service individually rather than upgrading in the entire application.
- Microservices allows us to develop an application which is organic (an application which latterly upgrades by adding more functions or modules) in nature.
- Microservices follows the single responsibility principle.
- The demanding service can be deployed on multiple servers to enhance performance.
- Less dependency and easy to test.
- Dynamic scaling.
- Faster release cycle.

Disadvantages of Microservices

- Microservices has all the associated complexities of the distributed system.
- There is a higher chance of failure during communication between different services.
- Difficult to manage a large number of services.
- The developer needs to solve the problem, such as network latency and load balancing.
- Complex testing over a distributed environment.

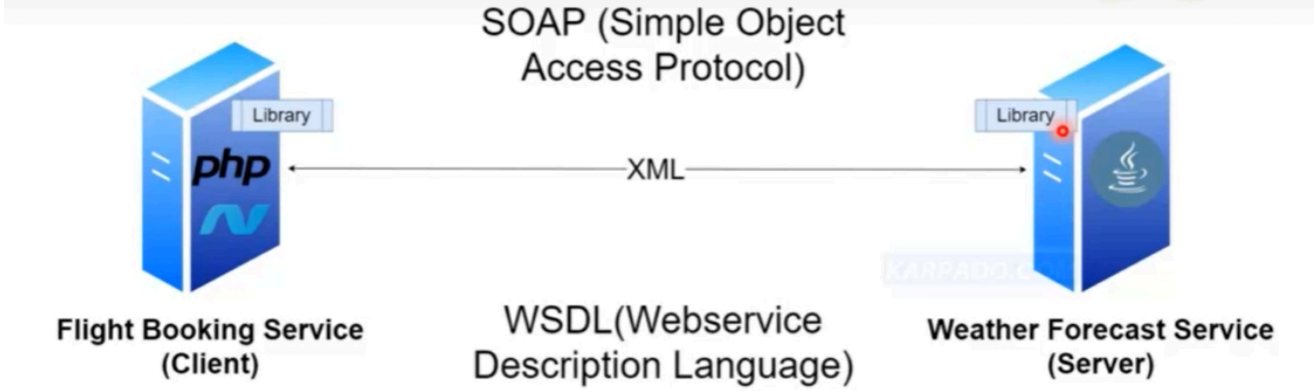
24.



25. What SOAP API means?

Ans - **Simple Object Access Protocol** (SOAP) is a message specification for exchanging information between systems and applications

https://stackoverflow.com/questions/2131965/main-differences-between-soap-and-restful-web-services-in-java



We have to use wisdom or lip service description language in order to define all the stuff.

26. **RESTful Web Service.**

Ans -

What are RESTful APIs Used For?

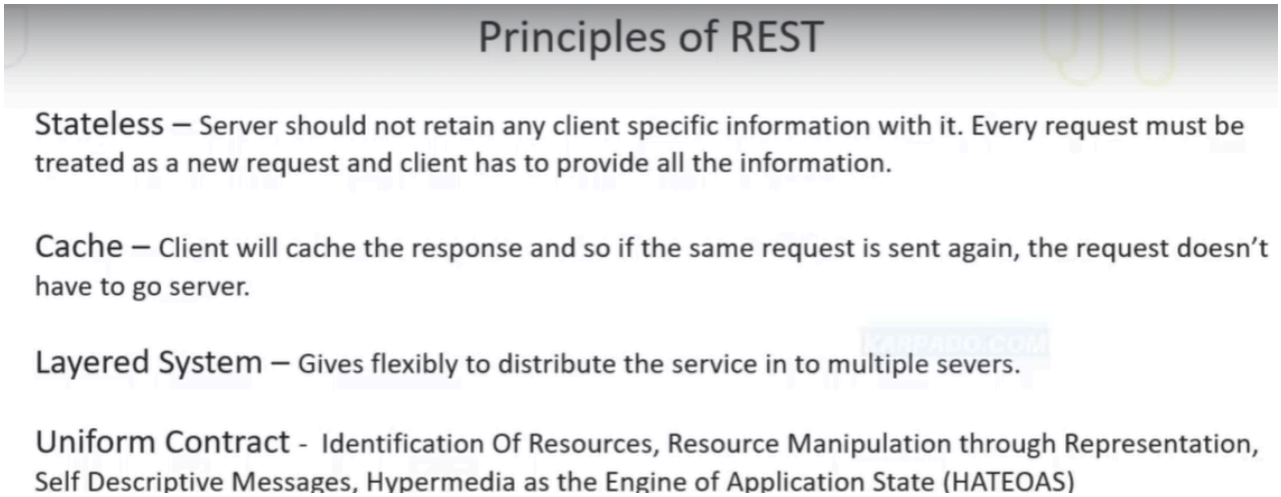
https://www.astera.com/type/blog/rest-api-definition/#What-is-REST-API?

A service which is built on REST architecture is called RESTful web-service.

RESTful webservices relies on HTTP methods to manipulate a resource on the server

- GET - To retrieve a resource from the server
- POST - To create a resource on the server
- PUT - To change the state or to update a resource
- DELETE - To remove or delete a resource from the server

REST API breaks a transaction down to generate a sequence of small components. Every component addresses a specific fundamental aspect of a transaction. This modularity makes it a flexible development approach.



Client-Server – Server will not care of presentation. Client won't care of data storage.

RESTful vs SOAP Webservices

- REST is an Architectural style. SOAP is a protocol
- SOAP is based on XML. REST supports JSON, XML, Plain text
- REST exposes API with URL Mapping. While SOAP use annotations.
- REST works well with JS technologies compared SOAP
- REST requires fewer resources and bandwidth as XML s
- REST response can be cached

Angular

What is Angular?

- Angular is a framework for building modern single-page applications

How is Single-Page application different than Traditional application?

Ans -

Traditional Application

- Each user action results in a full HTML page load

Single-Page Application

- A web application that is composed of a single page
- Based on user actions, the application page is updated
  - Normally performs a partial update … instead of full page load

TypeScript

- **FAQ: Why do most Angular developers use TypeScript?**
- Strongly-typed language with compile time checking and IDE suppo
- Increased developer productivity and efficiency
- The Angular framework is internally developed using TypeScript
- Docs, online blogs and tutorials use TypeScript for coding examples

- Web browsers do not understand TypeScript natively
- Have to convert TypeScript code to JavaScript code
- This is known as **"transpiling"**

Transpiling: Translating / Compiling