Java

Fundamental Part-1:-

1.What is the use of programming languages?

- It is used for communication between systems and humans. communicating with systems means giving commands using softwares.
- Also it is used to create or develop applications.such as web applications,mobile applications,website,restful services,iot applications,games.

. 2.What is Java?

- Java is a high level Object Oriented Programming language.Which is developed by SunMicrosystem and created by James Gosling in 1991.
- That is used for developing or creating a variety of applications including Enterprises application,Web application,Mobile Application,big data applications,server side technologies,IOT applications ,Games.
- Java is known for being simple ,efficient and secure.

3.What is platform independence?

- Platform independence means the ability of systems or programs to run on multiple operating systems such as Windows,MAC,Linux.

4.Why is the Java platform independent?

- Java is platform independent because compiled code of java we can run on any operating system.
- Because compiled code of java does not contain any machine instruction.
- It contains a bytecode which does not belong to any operating system.
- Java achieves platform independence because of compiled code.

5.What are the Features of java ?

- Simple
- Platform independent
- Architectural Neutral
- Robust/Strong
- Portable
- Distributed
- MultiThreaded
- Secure

6.Why is Java simple?

- Java is Simple because Java doesn't support a complex concept like a pointer so that Java program execution time will be less as well as less development time.
- Java is Simple because Java added a concept call garbage collection to destroy the useless object which free memory.

7.What is architectural neutral?

- There is no implementation dependent features e.g. size of
  primitive types are fixed.
- In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture.
- But in java, it occupies 4 bytes of memory for both 32 and 64 bit architectures.
- Hence java is architectural neutral with respect to memory management.

8.What is the use of translators like a compiler/ interpreter?

- In java translator like a compiler or interpreter are used to convert human readable code into machine code.
- The Java compiler takes the Java code and converts it into an intermediate form called bytecode.
- This bytecode is not directly run by the computer but by the **Java Virtual Machine (JVM)**.
- The JVM uses an **interpreter** to translate the bytecode into machine-specific instructions that the computer can execute.

9.What is class?

- Class is a keyword which is used to define user defined data type.
- Class is a logical entity.
- Class is just an imagination.
- Class is a blueprint of real time object.
- Class is a set of objects which shares common characteristics /behavior and common properties /attributes

10.What is an Object?

- Object is the physical existence of a class.
- Object is a physical entity.
- If we want to allocate memory for class non static variables then we have to create an object of that class.
- Creating an object means allocating memory for its non-static variables.

11.Class VS Object

- Class
  1.Class is a keyword which is used to define user defined data types.
  2.class is a logical entity.
  3.class is just an imagination.
  4. Class is a blueprint of real time object.
  5.Class is a set of objects which shares common characteristics/behavior and common properties / attributes.
- Object ➖
  1. Object is a physical existance of class
  2.Object is a physical entity.
  3. If we want to allocate memory for class non static variables then we can create an object of that class.
  4.creating an object means allocating memory for its non-static variables.

12 .How many objects can we create for a class?

- We can create multiple objects of one class.

13.What is the difference between a static variable and non-static variable?

- Static Variables:-

  -the variable which declared at class level with A static keyword is called a static variable.

  - lifeTime of static variables is from class loading to class unloading.

- Non-Static Variables:-

  .The variable which is declared at class level without a static keyword is called a non-static variable.

  .The lifetime of non-static variables is from object creation to object destruction.

  .non static variables get memory on object creation.

14.What is the scope of the local variable?

- Scope of local variables is for that method only,not outside the method.
- Scope of local variables is limited to the block of code.

15.What is the lifetime of a local variable?

- Lifetime of a local variable is from method loading to method unloading.

16.What is the lifetime of a static variable?

- Lifetime of a static variable is from class loading to the class unloading.

17.What is the scope of non-static variables?

- Scope of non static variable is within the method or outside the method,i,e at class level
- The scope of non-static variables is specific to the instance of a class, similar to a local variable. This means that you can only access non-static variables through an instance of a class.

Fundamental part - 2 :

1.what is the use of a static block in java.

- Static block is a one time execution block ,if we want to execute some logic after class loading then we can write that logic in a static block.
- Static block lifetime is from class creation to class destruction.
- Static block executed before main method.
- We can declare a static block with a static keyword.

2.what is a non-static block?

- A non-static block in java also called instance initializer block.
- It is a block of code which executes after creating an object of that class.
- Non-static blocks run each time an object of class is instantiated.

3. What is a constructor?

- constructor is a special time method whose name is the same name as a class name and doesn't have any return type.
- Generally we use it to initialize an instance of a variable.
- Constructor is an object level one time execution block.
- Constructor is automatically called when we create an object with a new keyword.

4.How many constructors can we write inside the class?

- In Java, you can write any number of constructors inside a class, as long as each constructor has a unique parameter list. This concept is known as constructor overloading. It allows you to define multiple constructors with different signatures (i.e., different numbers or types of parameters) in the same class

5.What are the types of a constructor?

- Default constructor.
- Explicit constructors.
- Copy constructors.
- Canonical constructors.
- Superclass constructors.
- Anonymous constructors.
- Inner class constructors.
- Enum constructors.
- Generic constructors.

6.What is a copy constructor?

- In Java, a copy constructor is a special type of constructor used to create a new object as a copy of an existing object.
- Although Java does not have a built-in copy constructor like some other languages (e.g., C++), you can implement it by defining a constructor that takes an object of the same class as a parameter.
- The copy constructor duplicates the values of the fields from the passed object into the new object.

7.What is class loading?

- Class loading in Java is the process by which the Java Virtual Machine (JVM) dynamically loads classes into memory during runtime.
- It is a critical part of the Java execution environment, allowing the JVM to locate and load Java classes into the runtime environment so that they can be used to create objects or execute code.
- Java uses a lazy loading mechanism, meaning that classes are only loaded when they are first referenced (used in the code), rather than loading all classes at the start of the application.

8.What are the access specifiers ? Explain to me different access specifiers?

- Access specifiers in java define access permission for a member class or method or variable or inner class.
- There are 4 types of access specifiers in java.

1.public

- Public specifiers achieve the highest level of accessibility in java .
- Classes methods one field declared as a public can be accessed from any class in the java program whether these classes are in the same package or in another pacakage.

2.private

- Private specifiers in java achieve the lowest level of accessiblilty.
- Private methods and fields can only be accessed within the same class to which method and fields belong.
- Private methods and fields are not visible within the subclasses and not inherited by subclasses .

3.protected

- Methods and fields declared protected can be accessed by the subclasses in other packages or any class within the package or any class within the package of the protected member class .

    - The protected specifiers can not be applied to the class and interface.

4.default.

- When we don't set access specifiers for the element it will follow the default accessibility level.
- There is no default specifier keyword.
- Classes ,variables and methods can be default accessed.
- Using this we can access class methods or fields which belong to the same package but not from outside the package.

9.How can we access private variables outside the class?

- Using the getter setter method we can access private variables outside the class.

10.What is the difference between protected and default specifiers?

1.protected

- ■ Methods and fields declared protected can be accessed by the subclasses in other packages or any class within the package or any class within the package of the protected member class .
- ■ The protected specifiers can not be applied to the class and interface.

2.default.

- When we don't set access specifiers for the element it will follow the default accessibility level.
- There is no default specifier keyword.
- Classes ,variables and methods can be default accessed.
- Using this we can access class methods or fields which belong to the same package but not from outside the package.

11.How many keywords are there in java?

- The Java programming language has 68 reserved keywords.
- These keywords have predefined meanings and cannot be used for certain purposes, such as variable, method, or class names.

12.Tell me coding standards related to class , variable, method.

Class Naming Standards:

- Class names should be written in PascalCase (also known as UpperCamelCase), where each word starts with a capital letter.
- Class names should be nouns or noun phrases that describe what the class represents.
- Keep the class name concise but descriptive.

Variable Naming Standards:

- Variable names should be written in camelCase, where the first word is lowercase, and subsequent words start with a capital letter.
- Use meaningful names that clearly indicate the purpose of the variable.
- Avoid using single-character variable names (unless in loops like `i`, `j`).
- Constants should be written in UPPER_CASE with words separated by underscores

Method Naming Standards:

- **Method** names should be written in camelCase (similar to variables) and should typically start with a verb because methods represent actions.
- The name should clearly describe what the method does.
- Method names should be descriptive but not too long.

Oop concept part -1

1.What is data hiding?

- Hiding internal confidential data from outsiders and providing only when he provide proper authentication is called data hiding.

2.How do we achieve data hiding in java?

- In java we achieve data hiding using private keywords.

3.what is an abstraction?

- Hiding internal implementation from the end user and providing only services is called abstraction.
- This helps in achieving data hiding and protects the integrity of objects.

4.what is an encapsulation?

- Encapsulation in java is a process of binding method and variables in a single unit are called as an encapsulation.
- Class is the best example of encapsulation.

5.what is an inheritance?

- Reusing a parent class property into a child class is called an inheritance.
- When we want some members from another class then we can extend this class.
- If we extend one class into another, one class becomes a parent and another becomes a child.
- All the members of the parent class are available to the child class so we don't have to write it again.

6.what is a multiple inheritance?

- Extending more than one class into one class is called multiple inheritance in java.
- And java does not support multiple inheritance.

7.why java does not support multiple inheritance?

- Java does not support multiple inheritance because of ambiguity problems.
- I.e if child classes are trying to access a member which is present in both parent classes then JVM will get confused from where it should access that member**.**

8.what is a multilevel inheritance?

- A class is derived from another class which is also derived from another class is called multilevel inheritance.
- Multilevel inheritance forming a chain of inheritance.

9.what is the differnce between IS-A and HAS-A relationship.?

- IS-A: this relation means when we want some member from any other class then we can extend that class and get members is called IS-A relation.
- HAS-A: this relation is when we want some member from another class then we can create an object of that class and access the member of that class.

10. What is constructor chaining?

- Constructor chaining is a technique in java object oriented programming ,where one constructor calls another constructor in the same class or super class .
- This is typically done to avoid the code duplication and modification.
- And to ensure that object initialization is handled in a consistent manner.
- constructor chaining is achieved using two ways.
    - Within the same class:
        - A constructor called another constructor of the same class using this keyword.

- From a superclass:
  A constructor calls another constructor of the super class using super keyword.

11.What is Aggregation?

- Aggregation is also known as has-a relationship
- Where part exist without the whole.
- Aggregation in java is a concept related to oops classes where one class contains a reference to one or more

objects to another class; this is also known as relationship.

12. What is a composition?

- Composition is a special form of aggregation where part can not exist without whole.
- Composition is a strong association.
- Composition is a relationship represented like aggregation with one difference that the diamond shape is filled.

13.How can we stop inheritance?

- We can stop inheritance using declaring this class method as final.

Oop concept part - 2:

1. **What is an overloading?**
    - If a class has multiple methods having the same name but different in parameters it is known as overloading.
    - If we have to perform only one operation, having the same name of the method increases the readability of the program .

2.what is the use of overloading ?

- Suppose we have to perform addition of the given numbers but there can be any number of arguments ,if you write method such as a(int a,int b) for two parameters and b(int a,int b,int c) for three parameters then it may be difficult for you as well as programmers to understand the behavior of the method because its name differ. That is why we use method overloading.
- Method overloading increases the readability of a program.

3.Why do we call overloading compile time polymorphism?

- Overloading is also called compile time polymorphism because overloading takes place at compile time and depends on reference type.
- Overloading is also called static polymorphism or early binding polymorphism.

4.Why do we call overloading static polymorphism?

- It's called static polymorphism because the decision of which method to invoke is made by the compiler before the program runs.
- The compiler knows which method to call based on the method signature (the method name and its parameters).

5.What is overriding?

- Overriding mens redefining parent class implementation in child class is called overrding.
- When a child class is not satisfied with parent class implementation , then the child class redefines that method in its own class.

6.What is the purpose of overriding?

- Purpose of overriding is redefining parent class implementation in the child class.
- Overdding allows you to write more flexible and reusable code.
- Overriding allows subclasses to provide specific functionality.
- While a parent class can define a generic implementation, each subclass can tailor that behavior to its needs without altering the parent class.

7.What are the rules of overriding?

- Return type should be the same or covariant.
- Access specifiers should be the same or increasing order.
- Method signature should be the same.
- We cannot override private methods.
- We can stop overriding by declaring the method as final.
- We can not override the main method.

8.What is the difference between overloading and overriding?

- **Overloading:-**
  - It is called compile time polymorphism.
  - Method signature should be diifernt.
  - Return type can be anything.
  - Access specifiers can be anything.
  - It achieves code redability.
  - It can happen in the same class as well as in child relation.

- **Overriding:-**
  - It is called run time polymorphism.
  - Method signatures are the same.
  - Return type should be the same or covariant.
  - Access specifiers should be the same or in increasing order.
  - It achieves code redefinition.
  - It can happen only in parent-child relations.

9.How can we stop overriding?

- We can stop overriding by declaring that method as final.

10.can we override private methods?(why)

- No, we cannot override private methods in java.
- Because In Java, private methods are only accessible within the class where they are defined and are not visible to subclasses.
- Since they are not inherited by child classes, they cannot be overridden in the subclass.

11.Can we override the static method?

- No, we cannot override static methods.
- Because overriding static methods means method hiding.
- Overriding involves dynamic method dispatch, which works based on the runtime type of the object.
- Since static methods are resolved at compile time (based on the class), they do not fit the mechanism of overriding, which requires the runtime determination of the method.

12.What is the difference between method hiding and overloading?

- **Method Hiding:-**
  - Resolution Time: Method hiding is resolved at compile time.
  - Binding: Static methods are class-level methods, and the method called is based on the reference type, not the object type.
  - Inheritance: In method hiding, the static method of the subclass hides the static method of the superclass, but it doesn't override it.
  - Behavior: The method is hidden, and the version from the superclass or subclass is called depending on the reference type (not the runtime object).
  - Scope: Hides the method in the superclass without overriding it. The superclass method still exists and can be called using

the superclass reference.
  - Modifiers: The subclass static method can have the same or different access modifiers as the superclass static method.
  - Runtime Type: The method called depends on the compile-time type of the reference, not the runtime type of the object.
  - Use Case: Typically used when you want to define a different implementation at the class level, but not associated with object behavior.

- **Method Overriding:-**
  - Resolution Time: Method overriding is resolved at runtime.
  - Binding: Instance methods are object-level methods, and the method called is based on the actual type of the object (runtime type).
  - Inheritance: Overriding allows the subclass to provide a specific implementation of a method that is inherited from the superclass.
  - Behavior: The overridden method in the subclass replaces the method in the superclass when called on a subclass instance.
  - Scope: The superclass method is effectively overridden and cannot be directly called once overridden, unless explicitly using `super`.
  - Modifiers: The overriding method must have the same or broader access than the method in the superclass.
  - Runtime Type: The method called is determined based on the runtime type of the object, regardless of the reference type.
  - Use Case: Overriding is used to provide a specific behavior for a method in the subclass while maintaining the same method signature.

Oop concept part -3:-

1.What is an abstract method?

- While declaring a method inside a class ,if we don't know what logic has to write inside that method then we can simply declare this method without logic .
- If a method doesn't have logic then we have to declare that method as an abstract method.
- I.e method without logic is called an abstract method.

2.What is an abstract class?

- Abstract class is a collection of abstract as well as normal methods.
- If class contains at least one abstract method then we can declare that class as abstract.

- Whenever we know partial implementation in that case we declare an abstract class.

3.Who provides implementation for abstract methods?

- Child class provides implementation for abstract class.

4.Can I instantiate an abstract class?

- No, you cannot instantiate an abstract class in Java. An abstract class is a class that cannot be instantiated directly and is meant to be subclassed.
- Because an abstract class is an incomplete class (incomplete in the sense it contains abstract methods without body and output) we cannot create an instance or object.

5.Can I make the abstract class final? (why)

- No, we cannot make an abstract class as final because an abstract class means to be extended by other classes to provide implementation for abstract class and making class final is to not extend it .
- I.e its child classes provide logic to abstract class methods.
- If we declare abstract class is final then it is not possible.

6.Can I declare the abstract method final?(why)

- No, you cannot declare an abstract method as `final`.
- Abstract methods are meant to be overridden in subclasses. They define a contract that the subclasses must fulfill by providing their specific implementations.
- Final methods, on the other hand, cannot be overridden by subclasses. A final method indicates

that the method's behavior is complete and cannot be changed by further inheritance.

7.Can I declare the abstract method static? (why)

- No, you cannot declare an abstract method as `static`.
- Static method means we can't create objects for static methods we directly call using class name.
- Abstract method does not contain any logic ,if we call that method as class name.method() then there is no sense.

8.Can I declare the abstract method private?(why)

- No, we cannot declare abstract methods as private.
- because, For abstract methods its child classes provide logic but if we declare abstract methods as private then this method is not visible to the child classes.

9.If we can't create objects of abstract class, why do we need a constructor inside abstract class?

- We need a constructor in an abstract class because even though we can't directly create objects of the abstract class ,its constructor can be used by subclasses.
- When a subclass inherits from an abstract class helps initialize common properties or perform necessary setup or fields that the subclass will also use.
- This ensures that all the inherited properties from the abstract class are properly initialized when creating an object of subclass.

10.If all methods are normal, can I make the class abstract? (what is the need of it)

- Yes we can make the class as abstarct.
- This is useful when you want subclasses to extend the functionality or provide specific implementations but don't want to allow creating instances of the base class itself.

11.What is the interface?

- Interface is a pure abstract class in java.
- Every method in an interface is a by default abstract and every variable in interface public static final .
- Interface contact between client and service provider.

12.Who provides logic to interface methods?

- Its child classes prvoide logic to interface methods.

13.Can I implement multiple interfaces? Will I get ambiguity?

- Yes, we can implement multiple interfaces.
- No, we can't get ambiguity.
- Java allows a class to implement more than one interface, which is a key feature for achieving multiple inheritance of behavior.
- However, ambiguity can arise if multiple interfaces declare methods with the same signature but without providing an implementation.

14.What is the difference between abstract class and interface?

- **Abstract class:-**
  - Abstract classes can have abstract and non abstract methods.
  - abstract class doesn't support multiple inheritance.
  - abstract classes have final ,non-final,static ,non-static variables.
  - Can provide the implementation of the interface.
  - abstract keyword use to declare abstract class.
  - Can extend another java class and implement multiple java interfaces.

- **Interface:-**
  - Interfaces can have only abstract methods.
  - Since java 8 it can have default and static methods also.
  - Support multiple inheritance.
  - Has only static and final variables.
  - Cant provide implementation of abstract class.
  - Interface keyword is used to declare interface.
  - An interface can extend other java interfaces only.

15.What is the marker interface?

- An empty interface we can implement in your class object will get some extra ability,then that interface is called a marker interface.

16.Why can't we write a constructor inside the interface?

- We cannot write a constructor inside an interface because by default all the variables of interface are public static and final ,so we cannot define non-static variables inside the interface hence there is no need to write a constructor.

Lang package :

1.Which package is the default package in java?

- The Java.lang package is the default package in java.

2.Why is the Object class the parent class of all?

- Object class is the default parent of every java class because this class contains some methods which are required in every java class.
- That is why declared this class as default parent.

3.List of methods of Object class

- *toString();*
- *hashcode();*
- *equals();*
- *finalize();*
- *getClass();*
- *clone();*
- *wait();*
- *wait(int ,int);*
- *wait(int,int,int);*
- *notify();*
- *notifyAll();*
- *forName();*

4.explain to me the String method.

- This method returns a String representation of the object.
- When we print a reference variable internally JVM calls the toString() method.
- If we want to print data from that object then we have to override the toString() method.
- When we print any reference variable then it will print classname@hashcode in hexadecimal format.

5.Why do we override the toString method?

- If we want to print data from that object then we have to override the toString() method.

6.Explain to me the hashCode method?

- Using this method we can get the hashcode of an object.
- Hashcode is a unique value which we assign to an object.
- Hashcode represents the memory address of an object in simplified format.

7.What is the clone method?

- Clone method is used to create a clone of an object,i.e copy of the object.
- It does shallow cloning, not deep cloning.
- To use it effectively ,a class must implement the cloneable interface and override the clone method.
- We need to override the clone method to make it public and handle the specific cloning for your class.
- If we want deep cloning then we have to override the clone method.

8.Difference Between Shallow Cloning and Deep cloning.

- **Shallow Cloning:-**
  - In shallow cloning only outer object clones will be obtained.
  - Create a new object but does not recursively clone the object that is referenced by the original object.
  - Copies primitive data type and references to objects.
- **Deep Cloning:-**
  - In Deep cloning outer as well as inner object clones will be obtained.
  - Deep cloning involves creating a new object and also recursively cloning all objects referenced by the original objects.
  - Create a completely independent copy of the original object and all objects referenced by it.

9.What is the finalize method?

- This method is called by gc thread before destroying an unreferenced object to perform clean up operations.
- It closes all connections.

10. What is the difference between == and equals method?

- **== ➖**
  - It is an operator.
  - If we use the == operator to compare normal variables it compares data .
  - If we use compare references it compares references.
  - Safely compare null values
- **equals():-**
  - It is a method present in an object class.
  - Which is meant to compare references.
  - It can be overridden to compare object content
  - Can throw a null pointer exception if called and null references.

11.Why do we override equals method?

- If we want to compare data then we can override the equals method.
- Equals method returns int data type.

12. Why should we override hashcode and equals method together?

- Overriding both `hashCode()` and `equals()` methods together is important when you want to ensure the correct behavior of objects in collections like `HashSet`, `HashMap`, `LinkedHashSet`, etc
- Overriding both methods ensures that logically equal objects are treated the same in hash-based collections.
- It prevents potential issues with duplicate entries, incorrect lookups, or poor performance.

Exception handling :

1.What is an exception?

- Exception is an unwanted situation because of which our programs terminate abnormally.
- Exception is a predefined class which is a child of a throwable class.
- Exceptions occur due to mistakes in logic.

2.What is an error?

- Error is a predefined class which is a child of a throwable class.
- Error occurs due to lack of system resources.
- Types of error
  - Out of memory error:-
  - Stack overflow error:-

3.What is the difference between exception and error?

- **Exception:-**
  - Exception is an unwanted situation because of which our programs terminate abnormally.
  - Exception is a predefined class which is a child of a throwable class.
  - Exceptions occur due to mistakes in logic.

- **Error:-**
  - Error is a predefined class which is a child of a throwable class.
  - Error occurs due to lack of system resources.

4.What is the difference between checked and unchecked exceptions?

- **Checked Exception:-**
    - Exceptions which are checked at compile time whether you handled it or not ,or checked exceptions we have to compulsory handle it or else you can not call a compiled code called as checked exception.
    - example:-
        - Filenotfound exception
        - Classnotfound exception

- **Unchecked Exception:-**
    - Exceptions which are not checked at compile time whether it is handled or not are called as unchecked exceptions.
    - These exceptions are even though we don't handle it still the program will compile sucessfully.
    - example:-
        - Arithmetic exception
        - Null pointer exception

5.How can we handle exceptions?

- We can handle exceptions using try catch blocks.

6.What is the use of try catch block?

- Try catch block is used to handle exceptions.
- So whatever statement raises an exception we put in try block and catch block will catch the exception so that program does not stop abnormally.
- The statements which can raise an exception we will write into a try block .
- If an exception occurs try block then catch block will get executed ,otherwise catch block will not execute.

7.What is the use of multiple catch blocks for single try block

- If in a try block multiple exceptions are going to occur then we can write multiple catch blocks.
- But at a time only one catch block will get executed.
- B+ecause if an exception occurs at try block, the remaining statement from try block will not get executed.
- We can write a multiple catch block but parent block should be last catch block.that is it should be in child to parent order.

8.What is the use of finally block?

- Finally Block we use with the try catch blocks.
- It executes in both cases whether an exception has occurred or not,so whatever connections we are using in the try block ,we close in the finally block.

9.What if we add a return statement in try block, then finally block executes or not?

- Yes, the `finally` block will execute even if a `return` statement is placed inside the `try` block.
- In Java, the `finally` block is always executed, regardless of what happens in the `try` or `catch` blocks, whether there is a `return` statement, an exception is thrown, or the program exits.
- The only exceptions are when the JVM terminates (e.g., via `System.exit()` or a fatal error) before the `finally` block can execute.

10.What is the use of throws keyword.?

- The java throws keyword is used to declare an exception.
- It gives information to programmers that an exception may occur so it is better for the programmer to provide the exception handling code so that normal flow can be maintained.
- Throws keyword never handles exceptions, it will terminate program abnormally so never use throws keyword in main method.

11.*What is the use of the throw keyword?*

- Throw keyword is used to raise a custom exception.
- Throw keyword is used to generate exceptions manually.

12.how can we develop custom exceptions?

- We can develop custom exceptions by extending our class with one of the classes from the exception hierarchy.
- There is no predefined exception which can display user defined messages,so we can create custom exceptions.

13.What is a try with resources?

- Try with resources which came in java 1.7 version ,in this we don't have to close the connection.
- It closes automatically once you get out of try catch blocks ,so we don't need to close the connections when we are using try with resources.
- If we open the connection in the try block then we have to close the connection in the finally block.
- But it is a headache to close the connection in finally block.
- Connection open in try block that should be get close automatically ,tpo do this we can use try with resources as given below.

14.what is a multi-catch block?

- A multi-catch block in Java allows you to handle multiple exceptions within a single `catch` block.
- If in a try block multiple exceptions are going to occur then we can write multiple catch blocks.
- But at a time only one catch block will get executed.
- Because if an exception occurs at try block, the remaining statement from try block will not get executed.
- We can write a multiple catch block but parent block should be last catch block.that is it should be in child to parent order

Collection part -1

1.What is a collection?

- In java collection is a framework that provides an architecture to store and manipulate a group of objects.
- Collection can achieve all the operations that you perform on data such as searching ,sorting ,insertion and manipulation.

2.What is the difference between collection and arrays?

- **Arrays:-**
    - The size of an array is fixed at the time of creation. Once created, you cannot change its size. You need to know the number of elements you want to store in advance.
    - Arrays can store both primitives (like `int`, `char`, etc.) and objects (like `String`, `Integer`, etc.).
    - Arrays allocate memory for a fixed number of elements, whether or not you use all of them.
    - Arrays are simple structures with basic functionality such as accessing elements by index, looping through them, etc.
    - They don't have built-in methods for adding, removing, or searching for elements.
    - An array is a single data structure, which stores elements of the same type in a fixed-size sequence.
    - Arrays generally provide better performance for accessing elements because they are stored in contiguous memory locations, allowing direct indexing.
    - Arrays are type-safe. If you create an array of a certain type (e.g., `int[]`), it can only hold elements of that type.
    - Arrays can be iterated using a traditional `for` loop or an enhanced `for-each` loop.
    - Arrays have limited utility methods provided by the `java.util.Arrays` class, such as `sort()`, `binarySearch()`, `fill()`, etc.

- Arrays are part of the Java language itself and are not part of any hierarchy.

- **Collections:-**
  - The size of most collections (like `ArrayList`, `HashSet`, etc.) is dynamic. Collections can grow or shrink in size as elements are added or removed.
  - Collections can only store objects. To store primitive types, you need to use their corresponding wrapper classes (e.g., `Integer` for `int`, `Double` for `double`).
  - Collections adjust their memory usage dynamically as elements are added or removed, making them more memory-efficient.
  - Collections, being part of the Java Collections Framework, provide many useful methods for managing elements, like `add()`, `remove()`, `contains()`, `size()`, and more. Collections also support data structures like lists, sets, and maps.
  - Collections are an entire framework of various data structures such as `List`, `Set`, `Queue`, `Map`, etc., each with different characteristics and usage scenarios.
  - Collections have more overhead due to the dynamic nature and added features. Access time can vary depending on the type of collection (e.g., `ArrayList`, `LinkedList`, `HashSet`).
  - Before Java 5, collections used to hold objects of type `Object`, which required explicit casting. Starting from Java 5, generics were introduced, making collections type-safe (e.g., `ArrayList<Integer>`).
  - Collections can be iterated using a `for-each` loop, or by using an iterator or a stream (in Java 8 and above).
  - Collections have a vast set of utility methods provided by the `java.util.Collections` class, such as `sort()`, `reverse()`, `shuffle()`, `min()`, `max()`, etc.
  - Collections are part of the Java Collections Framework, which defines interfaces like `List`, `Set`, `Queue`, and implementations like `ArrayList`, `HashSet`, `LinkedList`, etc.

3.Can you explain me the hierarchy

- The Java Collections Framework (JCF) provides a well-defined hierarchy of interfaces and classes to represent various types of data structures like lists, sets, queues, maps, and more.

```
                    Iterable
                       |
                    Collection
                       |
  -- -----------------------------------------------------------------
     |              |                       |
    List           Set                      Map
     |              |                    HashMap
 ArrayList       HashSet                  LinkedHashmap
LinkedList      LinkedHashSet            TreeMap
Vector          TreeSet
Stack
```

List Interface

- List (java.util.List): Represents an ordered collection (also known as a sequence). Elements in a `List` can be accessed by their index, and duplicates are allowed.
  - **Implementations:**
    - `ArrayList`: Resizable array implementation of `List`.
    - `LinkedList`: Doubly-linked list implementation of `List`.
    - `Vector`: Legacy class, similar to `ArrayList` but synchronized.
    - `Stack`: A subclass of `Vector`, implementing a last-in, first-out (LIFO) stack.

Set Interface

- Set (java.util.Set): Represents an unordered collection of unique elements (no duplicates allowed).
    - **Implementations:**
        - `HashSet`: Stores elements in a hash table, allowing fast access.
        - `LinkedHashSet`: Preserves insertion order with elements stored in a linked list.
        - `TreeSet`: Stores elements in a sorted order (implements `NavigableSet` and `SortedSet`), backed by a Red-Black tree.

Map Interface

- Map (java.util.Map): Represents a mapping from keys to values. Duplicate keys are not allowed, but values can be duplicated.
    - **Implementations:**
        - `HashMap`: Most commonly used map, backed by a hash table, providing fast access to elements based on keys.
        - `LinkedHashMap`: Similar to `HashMap`, but maintains insertion order.
        - `TreeMap`: Implements a sorted map, using a Red-Black tree to order keys.

4.What is the difference between List and Set?

- **List:-**
  - This interface we use when we want to group multiple objects into a single unit and we want to allow duplicates and preserve insertion order.
  - List Interface is implemented by ArrayList class,LinkedList Class and Vector and Stack.

- **Set:-**
  - This interface we use when we don't want duplicates and we don't bother about insertion order.
  - Set interface is implemented by HashSet class,LinkedHashset class and TreeSet class.

5.Explain me ArrayList class

- Basically the ArrayList class is implemented by List interface.
- This class we use when we want to group multiple objects in a single unit and we want insertion order preserved and we want to allow duplicates and perform more read operations on stored data.
- Default initial capacity is 10 blocks.
- Incremental capacity 3/2*current capacity +1.
- And internal data structure is a growable array.

6.Explain me Vector class

- Vector class we use when we want to allow duplicates and we want to insertion order preserve and we want to perform more read operations on stored data.
- Initial capacity is 10 blocks and will get reserved.
- Incremental capacity 2*current Capacity.
- This is the best choice when we want to perform more read operations on stored data,because internally the vector class implements a random access interface.
- When we want synchronization then we also use vector class because most of the methods of vector class are synchronized.

7.Explain me LinkedList class

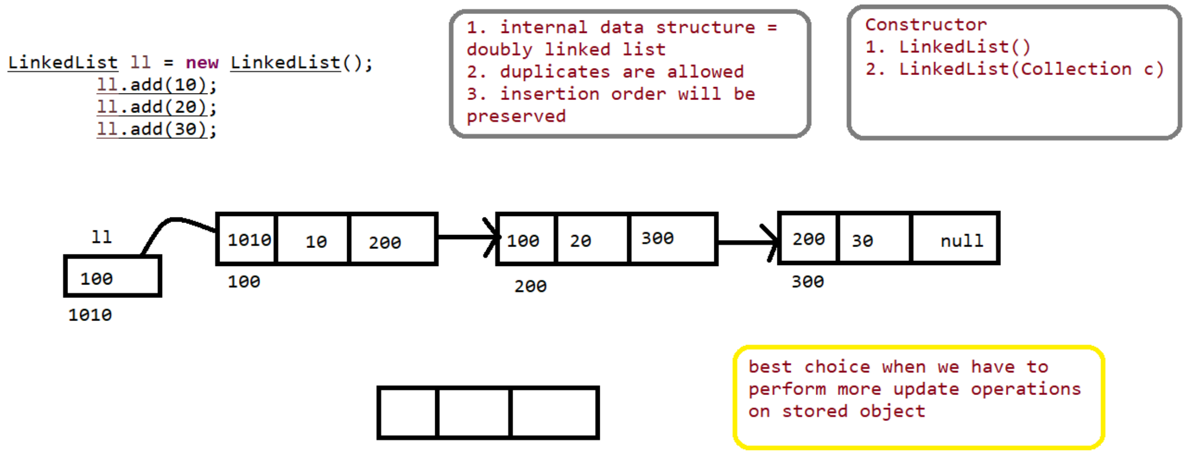
- Linkedlist class is implemented by list interface .
- We use linked list class when we want to allow duplicates and we want to preserve insertion order and we want to perform more update operations on stored data.
- This is best option when we want perform more update operations on stored data.

8.What is the difference between ArrayList and Vector

- **ArrayList:-**
  - It is a non legacy class.
  - Most of the methods of arraylist are non synchronized.
  - Less security.
  - High performance.
  - Incremental capacity =3/2*current capacity+1.

- **Vector:-**
  - This is a legacy class.
  - Most of the methods of vectors are synchronized.
  - High security.
  - Low performance
  - Incremental capacity=2*current capacity.

9.Can you explain to me how LinkedList works?

- Yes sure,
- A `LinkedList` is a doubly linked list where each element (node) contains a reference to the next and previous nodes.
- It allows efficient insertions and deletions at both ends or the middle (O(1) for these operations).
- However, accessing elements by index is slower (O(n)) because it requires traversing the list.
- It's ideal for frequent inserts/removals but not for random access, unlike an `ArrayList`, which provides faster index-based access (O(1)).
- 

```
LinkedList ll = new LinkedList();
        ll.add(10);
        ll.add(20);
        ll.add(30);
```

1. internal data structure = doubly linked list
2. duplicates are allowed
3. insertion order will be preserved

Constructor
1. LinkedList()
2. LinkedList(Collection c)

best choice when we have to perform more update operations on stored object

-

10.What is the initial and incremental capacity of ArrayList?

- Initial capacity is 10 blocks.
- And incremental capacity is 3/2 * current capacity +1.

11.What is the initial and incremental capacity of Vector?

- Initial capacity is 10 blocks.
- And incremental capacity is 2 * current capacity.

12.How can I make the ArrayList thread safe?

- **To make an `ArrayList` thread-safe, you can use:**
    - **Collections.synchronizedList()**
        - List<Integer> synchronizedList = Collections.synchronizedList(new ArrayList<>());
    - **Using `CopyOnWriteArrayList`:**
        - List<Integer> threadSafeList = new CopyOnWriteArrayList<>();

13.How can i remove duplicates from ArrayList?

- **Using a `Set`:**
    - Convert the `ArrayList` to a `Set`, which inherently does not allow duplicates, and then back to a list:
- **Using Java 8 Streams:**
    - You can use streams to filter duplicates:
- **Using `Iterator`:**
    - Manually remove duplicates while iterating:

Collection part -2

1.What is Set?

- Set is an interface which is used to store multiple objects into a single unit.
- We use this interface when we don’t want to allow duplicates and we don't bother about insertion order.
- Set interface is implemented by the HasSet class ,LinkedHashset class, treeSet class.

2.Explain to me the internal working of HashSet?

- A `HashSet` internally uses a `HashMap` to store its elements. Here's how it works:
  - Each element is stored as a key in the `HashMap`, with a constant dummy value (`Object`).
  - Hashing: The `hashCode()` of the element determines its position (bucket) in the underlying array.
  - If two elements have the same `hashCode()`, they are stored in the same bucket but linked via a chain (collision handling via chaining).
  - The `HashSet` does not allow duplicates, as keys in a `HashMap` must be unique.
  - So, when you add an element to a `HashSet`, it's essentially added as a key in the `HashMap.`

3. How HashSet identifies duplicate data?

- `HashSet` identifies duplicate data using two methods:

1. `hashCode()`: When an element is added, its hash code is computed. If two elements have the same hash code, they might be duplicates, but further checking is needed.
2. `equals()`: If two elements have the same hash code, `HashSet` then calls the `equals()` method to check for actual equality. If `equals()` returns `true`, the new element is considered a duplicate and is not added.

- So, both `hashCode()` and `equals()` must be implemented properly to ensure accurate duplicate detection.

4. When we add custom objects in Hashset, which two methods we have to override?

- For adding custom objects in hashSet then we have to override equals(), and hashCode() methods.

5. What is the default capacity of a hashtable in HashSet?

- The default capacity of a hashtable in hashSet is 16 blocks.

6. what is the fill ratio?

- The fill ratio, also called the load factor, is the threshold that decides when the hash table should increase its size.
- In a `HashSet`, the default load factor is 0.75, meaning when the set is 75% full, it resizes to prevent too many collisions.
- For example, if the capacity is 16, the set resizes after 12 elements are added.

7.How can I change the initial capacity and fill ratio in HashSet?

- You can change the initial capacity and fill ratio in a `HashSet` by using its constructor that accepts these parameters:
    - HashSet<E> hashSet = new HashSet<>(initialCapacity, loadFactor);
- initialCapacity: Specifies the initial size of the hash table.
- loadFactor: Specifies the fill ratio (when to resize).
    - HashSet<String> hashSet = new HashSet<>(32, 0.5f);

8.What is the difference between HashSet and LinkedHashSet?

- HashSet:-
    - We use this class when we want to store unique elements and don't want to allow insertion order.
    - Internal dataStructure is a hash table.
    - Initial capacity 16.
    - Default fill ratio =0.75

- LinkedHashSet:-
    - We use this class when we want to store unique elements and insertion order will be preserved.
    - Slightly slower than `HashSet` due to maintaining the insertion order using a doubly linked list.
    - LinkedHashSet: Uses both a hash table and a doubly linked list to preserve insertion order.

9.What is the use of TreeSet?

- When we don’t want duplicates and we want to insert elements according to sorting order of elements then use treeset class.
- Here the object will insert according to the natural sorting order.
- In a tree set we can not add heterogeneous data because we need to compare data.
- It's useful when you want to sort elements and access them efficiently, especially for larger sets.

10.How to sort data in TreeSet in descending order?

- To sort data in a `TreeSet` in descending order, you can use the `Comparator` interface when creating the `TreeSet`.
- Use `Comparator.reverseOrder()` to create a TreeSet that sorts elements in descending order.
- You can also create a custom comparator if you need a more complex sorting logic.

Collection part -3 :

1.What is the use of Map?

- We use map interface when we want to group multiple objects in the form of key value pairs.
- This data will be stored as key value.
- Key and value both will be stored as objects.
- Values can be duplicated.
- Keys can not be duplicated.
- Keys and values can be heterogeneous.
- We can add a number of null values in this.
- We can add only one null key.

2.Explain the internal mechanism of HashMap

- In hash table insertion order will not preserve because elements will insert according to the hashcode of keys.
- Hashmap stores key value pair in map entry static nested class implementation.
- A `HashMap` in Java stores key-value pairs in buckets, using the key's hash code to find the right bucket. If two keys end up in the same bucket (a collision), they are stored in a linked list.
- The `HashMap` has a default load factor of 0.75, meaning it will resize (usually double its size) when it gets 75% full to keep things efficient.
- On average, adding, getting, or removing items takes constant time, but if there are many collisions, it can take longer.

3.What is the initial capacity of HashMap?

- The initial capacity of HashMap is 16 buckets.

4.What is the fill ratio?

- The fill ratio, also called the load factor, is the threshold that decides when the hash table should increase its size.
- In a `HashSet`, the default load factor is 0.75, meaning when the set is 75% full, it resizes to prevent too many collisions.
- For example, if the capacity is 16, the set resizes after 12 elements are added.

5.How can I change the initial capacity and fill ratio in HashMap?

- You can change the initial capacity and load factor (fill ratio) of a `HashMap` by using its constructor that accepts these parameters.
- HashMap<K, V> hashMap = new HashMap<>(initialCapacity, loadFactor);

6.What is the use of TreeMap?

- This class stores data according to the sorting order of keys.
- Insert data according to natural sorting order.
- If we use custom order then use comparator.
- Its primary uses include maintaining a sorted map, which is helpful when you need to retrieve elements in order.

7.What is the difference between Comparable and Comparator?

- Comparable:-
    - Used to store a collection or array of objects.
    - Provides compareto() method.
    - Provide a natural sorting order.
    - Part of java.lang package.
    - We can use it to provide sorting based on single logic.
    - Modifies the class that implements it.

- Comparator:-
    - Used to sort collection of array objects.
    - Provide compare()methods.
    - Used to provide different algorithms for sorting.
    - Import java.util package.
    - Used to sort given collection objects.

8.What is the difference between HashMap and HashTable?

- Hashmap:-
    - Hashmap implements map interface and looks similar.
    - Hashmap allows null keys and values.
    - Hashmap is not synchronized.
    - Hashmap is better for a single threaded environment.

- HashTable:-
    - HashTable implements map interface and looks similar.
    - hashTable doesn't allow null key and values.
    - Hashtable is synchronized.
    - Hashtable is suitable for multithreaded environments.

9.What can we make HashMap synchronized?

- To make a `HashMap` synchronized (thread-safe), you can use
    - Using `Collections.synchronizedMap()`
    - `Using ConcurrentHashMap`

10.What is the difference between Synchronized hashmap and concurrent hashmap?

- Synchronized hashmap:-
  - Uses a single lock for the entire map, blocking access for all threads.
  - Slower in concurrent scenarios due to locking the whole map.
  - Iterators are not thread-safe and must be manually synchronized.
  - Allows one null key and multiple null values.
  - Suitable for small maps or low contention scenarios.

- concurrent hashmap:-
  - Employs finer-grained locking, allowing multiple threads to access different segments concurrently.
  - Faster, as it enables concurrent reads and writes without blocking the entire map.
  - Iterators are weakly consistent, allowing for safe concurrent modifications.
  - Does not permit null keys or null values.
  - Ideal for high-concurrency scenarios requiring better performance.

11.What is copyOnwriteArrayList?

- copyOnwriteArrayList is a thread safe variant of arraylist .
- In which all mutative operations like add sets are implemented by creating a fresh copy of the underlying array.
- It comes with the Java 1.5 version.
- copyOnwriteArrayList is fail-safe and it will never throw concurrent modification exceptions during iteration.

12.What happens when we add a duplicate key in Hashmap?

- When you add a duplicate key to a `HashMap`, the existing value associated with that key is overwritten with the new value.
- The `HashMap` does not allow duplicate keys; each key can map to only one value.

13.Collection vs Collections:-

- Collection:-
  - An interface that represents a group of objects. It is the root interface in the Java Collections Framework, defining basic operations for collections of objects (e.g., adding, removing, and iterating).
  - Part of the Java Collections Framework and is implemented by various classes such as `List`, `Set`, and `Queue`.
  - Used to define the behavior of collections and to create new collection types.

- Collections:-
  - A utility class that provides static methods for operating on collections, such as sorting, searching, and creating synchronized collections.
  - Not part of the hierarchy; it is a utility class (java.util.Collections) with static methods.
  - Used for utility functions and operations on collection objects, such as sorting a `List` or creating a synchronized version of a `Collection`.

Java 8 features:

1.Can you list out important features added in Java 8?

- Lambda expressions
- Functional interfaces
- Stream API
- Default methods in interfaces
- Method references
- Optional class
- New Date and Time API (java.time package)
- Type annotations
- Repeating annotations
- Nashorn JavaScript engine

2.What is a lambda expression?

- A lambda expression is a concise way to represent an anonymous function (i.e., a block of code) that can be passed as an argument to methods or used to create instances of functional interfaces. The syntax is `(parameters) -> expression` or `(parameters) -> { statements; }`.

3.What is the advantage of a lambda expression?

- The main advantages of lambda expressions include:
    - Conciseness: They reduce boilerplate code.
    - Improved readability: Code becomes easier to understand.
    - Support for functional programming: They facilitate the use of functional programming paradigms.
    - Ability to use in functional interfaces, enabling cleaner code in APIs.

4.What is a functional interface?

- A functional interface is an interface that contains exactly one abstract method. They can have multiple default or static methods. Examples include `Runnable`, `Callable`, and various `java.util.function` interfaces.

5.What is the Predicate interface?

- The `Predicate` interface is a functional interface that represents a boolean-valued function of one argument. It has a method `test(T t)` that returns `true` or `false` based on the evaluation of the condition.

6.What is the Consumer interface?

- The `Consumer` interface is a functional interface that represents an operation that takes a single input argument and returns no result. It has a method `accept(T t)` for performing the operation.

7.What is the Supplier interface?

- The `Supplier` interface is a functional interface that represents a supplier of results. It has a method `get()` that returns a result with no input parameters.

8.What is the difference between an anonymous inner class and a lambda expression?

- **Anonymous Inner Class:**
    - **R**equires more boilerplate code.
    - Can have multiple methods.
- **Lambda Expression:**
    - More concise and readable.
    - Can only implement functional interfaces (single abstract method).

9.What is Stream API?

- The Stream API is a feature in Java 8 that allows for processing sequences of elements (like collections) in a functional style. It supports operations like filtering, mapping, and reducing.

10.What is the use of `filter` in Stream API?

- The `filter` method is used to select elements from a stream based on a given predicate. It returns a new stream that contains only the elements that match the condition specified in the predicate.

11.What is the use of `map` in Stream API?

- The `map` method is used to transform each element in the stream by applying a given function. It returns a new stream consisting of the results of applying the function to the elements of the original stream.

12.Explain something about Joda-Time API.

- Joda-Time is a date and time handling library for Java that provides a more comprehensive and user-friendly alternative to the Java Date and Calendar classes. It supports better time zone handling, date manipulation, and formatting.

13.What features have been added related to collections?

- Java 8 introduced new methods in the `Collection` interface such as `forEach`, `stream`, `parallelStream`, `removeIf`, `spliterator`, and `stream.collect()` for better manipulation and processing of collections.

14.What changes have been added in interfaces?

- Java 8 allows interfaces to have default methods (with implementation) and static methods, enabling interfaces to evolve without breaking existing implementations.

15.What is the purpose of a default method in an interface?

- Default methods allow you to add new functionality to interfaces without breaking existing implementations. They provide a default behavior that implementing classes can override if needed.

JDBC:

1. **What is JDBC?**
    - JDBC (Java Database +Connectivity) is a Java API that enables Java applications to interact with databases. It provides methods for querying and updating data in a database.
2. **What are the steps to connect with a Database?**
    - Load the JDBC driver
    - Establish a connection to the database using `DriverManager.getConnection()`
    - Create a statement using `Connection.createStatement()`
    - Execute SQL queries using the statement
    - Process the results
    - Close the connection
3. **How do we do driver registration?**
    - Driver registration is typically done using `Class.forName("com.mysql.cj.jdbc.Driver")` for MySQL. This loads the driver class dynamically.
4. **What is the difference between Statement and PreparedStatement?**
    - Statement: Used for executing simple SQL queries without parameters; vulnerable to SQL injection.
    - PreparedStatement: Used for executing precompiled SQL queries with parameters; more secure and efficient for repeated execution.
5. **What is the difference between PreparedStatement and CallableStatement?**
    - PreparedStatement: Used for executing parameterized SQL queries.
    - CallableStatement: Used to execute stored procedures in the database.

6. **What is the difference between executeUpdate and executeQuery?**
    - executeUpdate: Used for executing SQL statements that modify data (INSERT, UPDATE, DELETE) and returns the number of affected rows.
    - executeQuery: Used for executing SQL SELECT statements and returns a `ResultSet` object containing the result.
7. **How and where do we close the connection?**
    - The connection should be closed in a `finally` block or try-with-resources statement to ensure it closes even if an exception occurs:
      `connection.close();`

8. **What is the use of ResultSet?**
    - The `ResultSet` interface provides methods for retrieving and manipulating the data returned from a SQL query. It represents a table of data generated by executing a statement.
9. **What is transaction management?**
    - Transaction management ensures that a series of database operations are executed as a single unit of work. If any operation fails, the entire transaction can be rolled back to maintain data integrity.
10. **What is a JDBC driver?**
    - A JDBC driver is a software component that enables Java applications to interact with a database. It converts Java calls into database-specific calls. There are four types of JDBC drivers:
        - Type 1: JDBC-ODBC Bridge Driver
        - Type 2: Native-API Driver
        - Type 3: Network Protocol Driver
        - Type 4: Thin Driver

Servlet:

1. **What is the use of servlets?**
    - Servlets are Java programs that run on a web server and handle client requests, process them, and return responses. They are used for building dynamic web applications.
2. **What is a web page?**
    - A web page is a document on the World Wide Web that is displayed in a web browser. It is typically written in HTML and may contain text, images, and links to other web pages.
3. **What is the difference between static and dynamic web pages?**
    - Static Web Pages: Content is fixed and does not change unless manually updated (e.g., HTML files).
    - Dynamic Web Pages: Content is generated dynamically based on user requests or data from a database (e.g., JSP, Servlets).
4. **What is web technology?**
    - Web technology refers to the tools, programming languages, frameworks, and protocols used to develop and deliver web applications. Examples include HTML, CSS, JavaScript, and server-side technologies like Java and PHP.
5. **What is the difference between client-side and server-side technology?**
    - Client-side Technology: Refers to code that runs in the user's browser (e.g., HTML, CSS, JavaScript).
    - Server-side Technology: Refers to code that runs on the server, processing requests and generating responses (e.g., Java Servlets, PHP, Node.js).

6. What is the use of a web server (Tomcat)?

- Tomcat is a web server and servlet container that provides a platform for running Java Servlets and JSPs. It handles HTTP requests and responses and serves web applications.

7. What is the use of the service method?

- The `service()` method in a Servlet is responsible for processing client requests. It takes two parameters: `HttpServletRequest` and `HttpServletResponse`, and dispatches the request to the appropriate method (doGet, doPost, etc.) based on the HTTP method used.

8. Explain the servlet lifecycle.

- The servlet lifecycle consists of the following phases:
  1. Loading: The servlet class is loaded into memory.
  2. Instantiation: The servlet instance is created.
  3. Initialization: The `init()` method is called to initialize the servlet.
  4. Request Handling: The `service()` method is called to handle requests.
  5. Destruction: The `destroy()` method is called before the servlet is removed from memory.

9. How can we read data from a form?

- Data from a form can be read using `HttpServletRequest` methods like `getParameter()`. For example:

```
String name = request.getParameter("name");
```

10. What is the difference between GET and POST?

- GET: Sends data in the URL; limited data size; used for retrieving data.
- POST: Sends data in the request body; no size limit; used for submitting data (e.g., form submissions).

11. **Tell me the different ways to create a Servlet.**
   - Using Annotations: With `@WebServlet` annotation.
   - Using Deployment Descriptor: Defining the servlet in the `web.xml` file.
   - Extending HttpServlet: Creating a class that extends `HttpServlet` and overrides methods like `doGet()` and `doPost()`.

Spring (Part 1)

1. **What is Spring?**
    - Spring is a comprehensive framework for building enterprise Java applications. It provides support for dependency injection, aspect-oriented programming, and various other features for building robust and scalable applications.
2. **What is the framework in general?**
    - A framework is a reusable software platform that provides a foundation for developing applications. It includes libraries, tools, and conventions that simplify the development process.
3. **What are the features of Spring?**
    - Dependency injection
    - Aspect-oriented programming
    - Transaction management
    - MVC framework
    - Data access (JDBC, JPA, Hibernate)
    - Integration with various technologies (e.g., REST, messaging)
    - Support for testing
4. **What is IoC?**
    - Inversion of Control (IoC) is a design principle in which the control of object creation and management is transferred from the application code to a container or framework (e.g., Spring).
5. **What is loose coupling?**
    - Loose coupling is a design principle that allows components to be independent of each other. Changes in one component do not directly affect others, making the system easier to manage and modify.
6. **What is dependency injection?**
    - Dependency Injection (DI) is a design pattern used to implement IoC, where an object's dependencies are provided (injected) externally rather than being created within the object itself.

7. **What are the different types of dependency injection?**
    - Constructor Injection: Dependencies are provided through the class constructor.
    - Setter Injection: Dependencies are provided through setter methods after object creation.
8. **Setter injection vs. constructor injection:**
    - **Setter Injection:**
        - Dependencies can be changed after object creation.
        - Suitable for optional dependencies.
    - Constructor Injection:
        - Ensures that all required dependencies are provided at instantiation.
        - Makes the object immutable once created.
9. **What are the different types of IoC containers?**
    - BeanFactory: The simplest container, providing basic support for DI.
    - ApplicationContext: A more advanced container that provides additional features such as event propagation, internationalization, and more.
10. **ApplicationContext vs. BeanFactory container:**
    - **ApplicationContext:**
        - More feature-rich (supports events, internationalization).
        - Loads beans eagerly.
    - BeanFactory:
        - Lightweight and used for simple applications.
        - Loads beans lazily.
11. **What is a Spring bean?**
    - A Spring bean is an object that is instantiated, managed, and configured by the Spring IoC container. Beans are created based on the configuration defined in XML files or annotations.

Spring Part -2

1. What is the default scope of a Spring bean?

The default scope of a Spring bean is singleton, meaning that only one instance of the bean will be created and shared across the entire Spring context. Each request for the bean will return the same instance.

2. How can I change the scope of Spring beans?

You can change the scope of a Spring bean by using the `@Scope` annotation in your configuration. The common scopes are:

- Singleton: One instance per Spring IoC container (default).
- Prototype: A new instance for each request.
- Request: One instance per HTTP request (valid in a web application).
- Session: One instance per HTTP session (valid in a web application).
- Global Session: One instance per global HTTP session (used in portlet contexts).

Example:

```
@Scope("prototype")
public class MyBean {
    // ...
}
```

3. What are the different ways to configure Spring beans?

There are several ways to configure Spring beans:

- XML Configuration: Define beans in an XML file (e.g., `applicationContext.xml`).
- Java Configuration: Use `@Configuration` classes with `@Bean` methods.
- Component Scanning: Annotate classes with `@Component`, `@Service`, `@Repository`, etc., and use `@ComponentScan` to scan packages.
- Properties Files: Externalize configuration using properties files.

4. What is an inner bean or nested bean?

An inner bean (or nested bean) is a bean defined within the scope of another bean in the Spring configuration. It is often used to encapsulate the configuration of related beans. Inner beans are defined directly within another bean's configuration block.

Example in XML:

```
<bean id="outerBean"
class="com.example.OuterClass">
    <property name="innerBean">
        <bean class="com.example.InnerClass"/>
    </property>
</bean>
```

5. What is the `p` namespace?

The `p` namespace is a shorthand for property injection in Spring XML configuration. It allows you to set properties of beans using a simpler syntax. By using the `p` namespace, you can avoid writing separate `<property>` tags.

Example:

```
<bean id="myBean" class="com.example.MyClass"
p:propertyName="value"/>
```

6. What is autowire?

Autowire is a Spring feature that allows you to automatically inject dependencies into your beans. Instead of explicitly defining the dependencies, Spring resolves and injects them automatically based on the configuration.

7. What is autowire byName and byType?

- Autowire byName: Spring searches for a bean with the same name as the property to be injected. If found, it is injected into the target bean.
- Autowire byType: Spring searches for a bean of the same type as the property. If exactly one matching bean is found, it is injected.

8. What is an autowire candidate?

An autowire candidate is a bean that can be considered for autowiring by Spring. It must be a Spring-managed bean (i.e., defined in the Spring context). If multiple candidates are found, Spring needs to resolve which one to use, often using qualifiers or specific autowiring strategies.

9. What is `DispatcherServlet`?

DispatcherServlet is a core component of the Spring MVC framework. It acts as the front controller that handles incoming HTTP requests, dispatches them to the appropriate controllers, and manages the entire request-response lifecycle.

10. What is the use of `spring-servlet.xml` file?

The `spring-servlet.xml` file is a configuration file typically used in Spring MVC applications. It contains bean definitions, view resolver configurations, and other MVC-specific settings, including mapping URLs to specific controllers.

11. What is the use of `@Controller` annotation?

The `@Controller` annotation is used to mark a class as a Spring MVC controller. It indicates that the class will handle web requests and define request-handling methods annotated with `@RequestMapping` or similar annotations.

12. What is the use of `@RequestMapping` annotation?

The `@RequestMapping` annotation is used to map HTTP requests to handler methods in a controller. It can specify the request path, HTTP method (GET, POST, etc.), request parameters, headers, and more.

Example:

```
@RequestMapping(value = "/home", method =
RequestMethod.GET)
public String home() {
    return "home";
}
```

Spring Part-3

1. What is MVC architecture?

MVC (Model-View-Controller) is a design pattern used for developing user interfaces. It separates an application into three interconnected components:

- Model: Represents the data and the business logic. It directly manages the data, logic, and rules of the application.
- View: Represents the presentation layer. It displays the data from the model to the user and sends user commands to the controller.
- Controller: Acts as an intermediary between the Model and View. It processes user input, interacts with the model, and updates the view.

2. What is the difference between `@Controller` and `@RestController` annotation?

- `@Controller`: Used in MVC applications to define a controller that returns views (HTML pages). It typically works with `ModelAndView`.
- `@RestController`: A specialized version of `@Controller`, used in RESTful web services. It combines `@Controller` and `@ResponseBody`, meaning it automatically serializes the response to JSON or XML instead of returning a view.

3. What is form backup support?

Form backup support refers to a mechanism in web applications to preserve the state of user input in forms. This allows users to navigate away and return without losing their input, often achieved through session management or browser storage.

4. What is JSR validation?

JSR (Java Specification Request) validation is a standard for defining validation rules in Java applications. JSR 303 (Bean Validation) allows you to use annotations (like `@NotNull`, `@Size`, etc.) to validate object fields, ensuring that the input data meets specified constraints before processing.

5. What is the use of `JdbcTemplate`?

`JdbcTemplate` is a Spring framework class that simplifies the use of JDBC (Java Database Connectivity). It provides a set of methods for querying and updating the database, handling exceptions, and managing resources, which reduces boilerplate code and enhances productivity.

6. What is the use of `RowMapper`?

`RowMapper` is an interface in Spring JDBC that helps convert rows of a ResultSet into objects. It defines a single method, `mapRow()`, which takes a ResultSet and row number as arguments, allowing you to specify how to map each row to an object.

7. What is the difference between `template.query` and `template.queryForObject` method?

- `template.query`: Used to execute a query that returns multiple rows. It maps each row to an object using a `RowMapper`.
- `template.queryForObject`: Used to execute a query that is expected to return a single object. If no rows or more than one row is returned, it throws an exception.

8. What is the use of `ViewResolver`?

`ViewResolver` is a Spring component that resolves logical view names returned by controllers into actual view implementations. It helps to determine which view to render (like JSP or HTML) based on the logical name provided.

9. What is the use of `@Autowired` annotation?

`@Autowired` is a Spring annotation that allows for automatic dependency injection. It tells Spring to automatically wire a bean into another bean without explicitly defining it in the configuration.

10. What is the use of `@Qualifier` annotation?

`@Qualifier` is used in conjunction with `@Autowired` to resolve ambiguity when multiple beans of the same type exist in the Spring context. It allows you to specify which bean to inject by providing the bean's name.

11.Explain to me the difference between MVC and Rest architecture.

MVC (Model-View-Controller)

1. Definition:
    - MVC is a design pattern used primarily for developing user interfaces by separating an application into three interconnected components: Model, View, and Controller.
2. Components:
    - Model: Manages the data and business logic of the application. It communicates with the database and processes data.
    - View: Represents the presentation layer. It displays the data to the user and allows for user interaction.
    - Controller: Acts as an intermediary between the Model and View. It processes user input, manipulates the Model, and updates the View.
3. Data Handling:
    - Typically returns HTML views that are rendered on the client side, often using templating engines.
4. Use Case:
    - Ideal for traditional web applications where user interfaces are heavily involved, and there is a need for interaction between the user and the server.
5. Statefulness:
    - Generally stateful, maintaining session information across requests (e.g., user sessions).

REST (Representational State Transfer)

1. Definition:
   - REST is an architectural style for designing networked applications. It is based on stateless, client-server communication, often used for web services.
2. Components:
   - There are no defined components like MVC; instead, it revolves around resources and their representations (often JSON or XML).
3. Data Handling:
   - Operates over HTTP using standard methods (GET, POST, PUT, DELETE). Returns data in formats like JSON or XML instead of rendering HTML views.
4. Use Case:
   - Suitable for APIs and microservices where the focus is on resource manipulation and data exchange between clients and servers.
5. Statelessness:
   - Stateless by design, meaning each request from the client to the server must contain all the information needed to understand and process the request. No session information is stored on the server.

Spring Boot 1

1. **What is Spring Boot?**
   - Spring Boot is an extension of the Spring framework that simplifies the setup and development of new Spring applications. It provides a range of out-of-the-box features and conventions that allow developers to create standalone, production-ready Spring applications with minimal configuration.
2. **What is the difference between Spring and Spring Boot?**
   - Spring: A comprehensive framework for building Java applications, requiring manual configuration of components.
   - Spring Boot: A framework that simplifies Spring application development by providing default configurations, embedded servers, and easier dependency management through starters.
3. **List of some important features of Spring Boot:**
   - Auto-configuration
   - Standalone applications (embedded servers like Tomcat, Jetty)
   - Spring Boot Starter dependencies
   - Production-ready features (Actuator)
   - Command-line interface (CLI)
   - Integrated with Spring Security
   - Simplified application properties management
   - Easy configuration with YAML or properties files

4. **What is `@GetMapping`?**
   - `@GetMapping` is a shortcut for `@RequestMapping(method = RequestMethod.GET)`. It is used to map HTTP GET requests to specific handler methods in a controller.

5. **What is the difference between GET and POST?**
   - GET: Retrieves data from the server; parameters are sent in the URL. It is idempotent and can be cached.
   - POST: Sends data to the server to create or update a resource; parameters are sent in the request body. It is not idempotent and cannot be cached.

6. **What is the difference between POST and PUT?**
   - POST: Used to create a new resource. The server assigns the new resource's ID.
   - PUT: Used to update an existing resource, replacing the current representation with the request payload. The client specifies the resource's ID.
7. **What is the difference between PUT and PATCH?**
   - PUT: Replaces the entire resource with the provided data.
   - PATCH: Applies partial modifications to a resource, updating only the fields provided in the request.
8. **What is `@RestController` annotation?**
   - `@RestController` is a specialized version of `@Controller` that combines `@Controller` and `@ResponseBody`. It indicates that the methods return data (typically in JSON or XML format) instead of views.
9. **What is the difference between `@RestController` and `@Controller`?**
   - `@RestController`: Automatically serializes the response body into JSON or XML.
   - `@Controller`: Typically returns view names for rendering HTML pages.

10. What is the difference between `@Controller`, `@Service`, `@Repository`, and `@Component` annotations?

- `@Controller`: Used in the presentation layer to handle user requests.
- `@Service`: Used in the service layer to encapsulate business logic.
- `@Repository`: Used in the persistence layer for data access logic (interacting with the database).
- `@Component`: A generic stereotype for any Spring-managed component, typically used for beans that do not fit into the other categories.

11. Which annotations have you used in JPA?

- Common JPA annotations include:
  - `@Entity`: Marks a class as a JPA entity.
  - `@Table`: Specifies the table name.
  - `@Id`: Marks the primary key field.
  - `@GeneratedValue`: Specifies how the primary key should be generated.
  - `@Column`: Specifies the column properties.
  - `@ManyToOne`, `@OneToMany`, `@ManyToMany`: Defines relationships between entities.

12. What is the use of the Service layer?

- The Service layer contains business logic and serves as an intermediary between the controller and repository layers. It handles the application's core functionality and processes requests from the controller.

13. What is the use of the DAO layer?

- The DAO (Data Access Object) layer provides an abstraction for data access. It contains methods for interacting with the database (CRUD operations) and is usually implemented using the Repository pattern.

Spring Boot Part-2

1.Which new features have been added in Spring Boot?

- Some new features in recent versions of Spring Boot include:
  - Improved support for reactive programming
  - Enhanced configuration properties binding
  - New metrics and observability features
  - Support for Spring WebFlux
  - Updated Actuator endpoints

2. What is an actuator?

- The Spring Boot Actuator provides production-ready features such as health checks, metrics, and application monitoring. It exposes endpoints that give insights into the application's internal state and behavior.

3.What is autoconfiguration?

- Autoconfiguration is a feature in Spring Boot that automatically configures your application based on the dependencies present in the classpath. It aims to reduce the amount of manual configuration required.

4.What is the use of Postman?

- Postman is an API development and testing tool that allows developers to send requests to web services, inspect responses, and automate API tests. It simplifies the process of working with APIs.

5.What is the use of Content-Type?

- The `Content-Type` header specifies the media type of the resource being sent to the server or returned in the response. It tells the server how to interpret the incoming request body or how the client should interpret the response.

6.If I want to return data from my REST service in the form of XML, then what should I do?

- To return XML from a REST service, ensure you have a dependency for XML support (e.g., Jackson XML). Set the `Content-Type` header to `application/xml` in the request, and use appropriate annotations in your controller to produce XML.

7.What is CORS policy error?

- CORS (Cross-Origin Resource Sharing) policy error occurs when a web application running at one origin (domain) attempts to make requests to a resource on a different origin. Browsers restrict such requests for security reasons unless the server explicitly allows it.

8.What is the use of properties file?

- Properties files in Spring Boot are used for externalizing configuration. They allow you to define application settings, such as database connection details, application port, etc., in a structured way.

9.How can we read data from properties file?

- You can read data from a properties file using the `@Value` annotation or by using `@ConfigurationProperties`. For example:

```
@Value("${app.name}")
private String appName;
```

10. What is the use of `@PropertySource` annotation?

- The `@PropertySource` annotation is used to specify the location of properties files to be loaded into the Spring Environment. It allows you to include additional property files in your application context.

11. What is the use of `@ComponentScan` annotation?

- The `@ComponentScan` annotation is used to specify the packages to be scanned for Spring components (beans). It is typically used in configuration classes to automatically discover and register beans.

12. What is JSR validation?

- JSR (Java Specification Request) validation refers to a standard way of validating Java beans using annotations defined in the Bean Validation API (JSR 303). It provides a set of annotations for validating fields in Java classes.

13.Explain the coding standards for writing an API.

- Some coding standards for writing APIs include:
  - Use meaningful and descriptive endpoint names.
  - Follow RESTful conventions (use appropriate HTTP methods).
  - Implement proper error handling and return appropriate status codes.
  - Use consistent data formats (JSON or XML).
  - Document the API using OpenAPI (Swagger) or similar tools.
  - Version your API for backward compatibility.

14.What is the use of RequestDto and ResponseDto?

- RequestDto: Used to encapsulate and validate the data sent from the client to the server in API requests.
- ResponseDto: Used to structure the data returned from the server to the client, often providing a standardized format for responses.

15.What is the use of the H2 database?

- H2 is an in-memory database that is lightweight, fast, and easy to use. It is commonly used for development, testing, and prototyping in Spring Boot applications due to its simplicity and ability to run without installation.