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

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

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

OOP BASICS

What is an object?
An object is an instance of a class. It has its own state, behavior, and identity.

What is a class?
It is a blueprint that defines what an object should look like. Where you list all the fields and methods what an object should have.

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

What is 'this' operator?
'this' pointer refers to the current object of a class.

What is 'final' keyword?
Protects methods or variables to be over ridden or modified.

What is 'static' keyword?
A variable or a method can be accessed without requiring an instantiation of the class to which it belongs. However it can be changed but will changed for every object and that class.

What are constructors?
Constructors are special type of methods that are used for creating & initializing an object of a class.
1. Default Constructor (Automatically created if no constructor created.
2. Parameterized Constructor

ENCAPSULATION

What is Encapsulation?

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

Why we use getter and setter?

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

What are access modifiers?

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

What are different access modifiers?

1. Private: It can only be accessed within your class. Subclasses or any other classes within the same or a different package can't access this attribute or method.
2. No modifier: It can be accessed within your same class and from all classes within the same package. That's why it's often called package-private.
3. Protected: It can be accessed within your class, by all classes within the same package, and by all subclasses within the same or other packages.
4. Public: It can be accessed within your current class and by all other classes.

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

ABSTRACTION

What is Abstraction?

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

Example: TV, Laptop, Careem Application etc.

How abstraction can be achieved?

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

ABSTRACT CLASS

What is an abstract class?

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

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

class Samsung extends TV{

    public void changeVolume(){
        System.out.println("Volume in Samsung");
    }

    public void changeChannel(){
        System.out.println("Channel in Samsung");
    }
}
```

Example: Car, TV etc.

What is a concrete class?

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

What is an abstract method?

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

INTERFACE

What is an Interface?

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

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

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

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

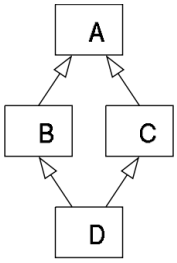
INHERITANCE

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

What are different types of inheritance?

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

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



POLYMORPHISM

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

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

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

What are different types of Polymorphism?

1. Method Overloading (Compile time Polymorphism, Static Binding, Early Binding)
2. Method Overriding (Run time Polymorphism, Dynamic Binding, Late Binding, Dynamic Dispatch)

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

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

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

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

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

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

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

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