**OOPS Concepts**

**Class:**

* In java, Class is a blueprint or template that defines the properties (fields) and behaviours (methods) common to a group of objects.
* It serves as a model for creating objects with specific attributes and functionalities.
* Class is a blueprint for creating objects in Java.

**Object:**

* An object is an instance of a class that represents a real-world entity. It encapsulates data (fields) and behaviour (methods) defined in its class, allowing manipulation and interaction within a program.
* objects are instances of classes.

**Encapsulation:**

* Encapsulation is the concept of bundling data (fields) and methods (behaviours) within a class, where data is accessed and modified only through methods, ensuring data integrity and providing control over access levels (public, private, protected).
* Encapsulation is the process of bundling data and methods within a class to restrict access and ensure data integrity.

**This keyword:**

* **This** keyword refers to the current object in a method or constructor.

**Constructors:**

* A constructor in Java is a special method that is used to initialize objects.
* The constructor is called when an object of a class is created.
* It can be used to set initial values for object attributes

**Access Modifiers:**

* Controls the access level for classes, attributes, methods and constructors.
* Public – Accessed by other classes. (Class)
* Default – If we don’t provide anything, default is used. It provides access to class in the same package. (Class).

**For Attributes, methods and constructors**

* Public - The code is accessible for all classes
* Private- The code is only accessible within the declared class
* default - The code is only accessible in the same package. This is used when you don't specify a modifier. You will learn more about packages in the Packages chapter
* protected-The code is accessible in the same package and subclasses

**Non-Access Modifiers:**

**Classes**

* Final - The class cannot be inherited by other classes
* Abstract - The class cannot be used to create objects (To access an abstract class, it must be inherited from another class.

**For attributes and methods**

* Final- Attributes and methods cannot be overridden/modified
* Static- Attributes and methods belongs to the class, rather than an object
* Abstract- Can only be used in an abstract class and can only be used on methods. The method does not have a body.
* Transient- Attributes and methods are skipped when serializing the object containing them
* Synchronized- Methods can only be accessed by one thread at a time
* Volatile- The value of an attribute is not cached thread-locally, and is always read from the "main memory"

**Inheritance:**

* In Java, it is possible to inherit attributes and methods from one class to another.
* It is useful for code reusability: reuse attributes and methods of an existing class when you create a new class.
* The done using **extends** keyword.
* Superclass – parent class from which we can inherit
* Subclass/child class - the class that inherits from another class.

**Polymorphism:**

* Inheritance lets us inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.
* With method overloading, multiple methods can have the same name with different parameters.
* The **print ()** method is also an example of polymorphism. It is used to print values of different types like char, int, string, etc.

**Method Overriding:**

* If the same method is present in both the superclass and the subclass. Then, the method in the subclass overrides the same method in the superclass. This is called method overriding

**Method Overloading:**

* we can create methods with the same name if they differ in parameters.

**Java Super:**

* The super keyword in Java is used in subclasses to access superclass members (attributes, constructors and methods).

**Data Abstraction:**

* Data abstraction is the process of hiding certain details and showing only essential information to the user.
* Abstraction can be achieved with either abstract classes or interfaces.

**Abstract Method:**

* A method that doesn't have its body is known as an abstract method. We use the same abstract keyword to create abstract methods.