**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.

**THREADS**

**Synchronization:**

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

Thread.currentThread().getName() – To get thread information

**Synchronized Method:**

* Locks the instance (object) on which the method is called, allowing only one thread to execute the synchronized method on that instance at a time

**Synchronized Block:**

* Allows for more granular synchronization by locking a specific block of code within a method, typically using synchronized(this) or a shared object to control access to critical sections.

**Static Synchronized:**

* Synchronizes a static method or block at the class level, preventing concurrent execution by multiple threads across all instances of the class by locking the class's monitor.

**wait ():**

* Causes the current thread to wait until another thread calls notify () or notifyAll() on the same object.

**notify ():**

* Wakes up one waiting thread that is waiting on the same object, allowing it to continue execution.

**notifyAll():**

* Wakes up all waiting threads that are waiting on the same object, allowing them to compete for the object's lock and resume execution.

**FILES**

* **createNewFile()** – To create new file
* **Scanner** – To read the file contents
* **getName()** – To get name of the file
* **Write()** – To write inti the file
* **FileWriter** – used along with write to write into the file.
* **Delete()** – to delete the file.

**BINARY FILES**

Binary format refers to representing data using the binary number system, which uses only two digits: 0 and 1. In computer systems, data is typically stored, processed, and transmitted in binary format because it directly corresponds to the on-off states of electronic switches in digital circuits, making it efficient for electronic devices to work with.

* **FileOutputStream:** Used for writing bytes to a file. It's part of Java's IO package (java.io).
* **FileInputStream:** Used for reading bytes from a file. Also part of Java's IO package (java.io).
* **ObjectOutputStream:** Used for writing serialized Java objects to an output stream (e.g., a file). It's used with FileOutputStream to save objects to a file.
* **ObjectInputStream:** Used for reading serialized Java objects from an input stream (e.g., a file). It's used with FileInputStream to read objects from a file.
* **FileOutputStream** writes bytes to a file.
* **FileInputStream** reads bytes from a file.
* **ObjectOutputStream** writes serialized Java objects to a stream. It "serializes" the objects, which means it converts them into a format that can be saved to a file.
* **ObjectInputStream** reads serialized Java objects from a stream

**Text File vs Binary File**

* A text file is a type of file that contains human-readable characters, such as letters, numbers, and symbols.
* Text files store data using character encoding, such as ASCII or Unicode, where each character corresponds to a specific binary value.
* binary file is a type of file that contains data in binary format, represented as sequences of 0s and 1s.
* Binary files store data in its raw binary form, without encoding characters directly. They are used for storing non-textual data, such as images, audio, video, or serialized Java objects.