* What is Java?
* Java is a high-level programming language and is platform independent.
* What is the difference between JDK, JRE, JVM?
* JVM (Java Virtual Machine) A virtual computer that is implemented in software allowing Java to run on many different operating system
* JRE (Java Runtime Environment) is a runtime environment which implements JVM and **provides all class libraries and other files that JVM uses at runtime**.
* JDK(Java Development Kit) is the tool necessary to compile, document and package Java programs. The JDK completely includes JRE.
* Explain to me about Object Oriented Programming Concepts
* Modelling the problem and the solution as a set of objects.
* Everything is an object
* A program is a collection of these objects interacting to solve a problem by sending messages to one another
* An object can be made up of other objects
* Every object has at least one type
* Objects of the same type can receive the same messages
* State, Behaviour, and Identity.
* Classes are like the blueprint of a house.
* Encapsulation 🡪 In OOP, we encapsulate by binding the data and functions that operate on that data into a single unit known as the class. This hides private details of a class from the outside world and only exposes functionality important for interfacing with it. When a class does not allow calling code access to its private data directly, we say that it is well encapsulated.
* Abstraction 🡪 This process is called “abstraction” in OOP because we are abstracting away the implementation details of a class and only presenting a clean, easy-to-use interface via the class’s member functions. Carefully used, abstraction helps isolate the impact of changes made to the code so that if something goes wrong, the change will only affect the implementation details of a class and not the outside code.
* Inheritance 🡪 Classes can be organized into hierarchies where a class might have one or more parent or child classes. For example, a parent class could be a class Animal 🡪 where it has walk() as a method. Its child class could be a class Dog 🡪 where we can create a Dog object where it inherits its parent method and attributes.
* Polymorphism 🡪 In OOP, polymorphism allows for the uniform treatment of classes in a hierarchy. Therefore, calling code only needs to be written to handle objects from the root of the hierarchy, and any object instantiated by any child class in the hierarchy will be handled in the same way. For example,  we have a class called, “Animal” and two child classes, “Cat,” and “Dog.” If the Animal class has a method to make a noise, called, “makeNoise,” then, we can override the "makeNoise" function that is inherited by the sub-classes, "Cat" and "Dog," to be “meow” and “bark,” respectively. Another function can, then, be written that accepts any Animal object as a parameter and invokes its "makeNoise" member function. The noise will be different: either a “meow” or a “bark” depending on the type of animal object that was actually passed to the function.