Uriel Lezama
Front-end-Pacific Week 5

1. Four pillars of Object-Oriented Programming (OOP):

a. Encapsulation: Encapsulation is one of the fundamental concepts of OOP that refers to the bundling of data (attributes) and methods (functions) that operate on that data within a single unit called a class. This principle helps in hiding the internal implementation details of a class and allows the class to provide a well-defined interface to interact with its objects. Encapsulation promotes data abstraction and helps in creating modular and maintainable code.

b. Inheritance: Inheritance allows a class (subclass or derived class) to inherit properties and behaviors from another class (superclass or base class). By doing so, it promotes code reuse and hierarchical organization of classes. Subclasses can extend the functionality of the superclass by adding new methods or overriding existing ones. Inheritance fosters a "is-a" relationship between classes, where a subclass is a specialized version of its superclass.

c. Polymorphism: Polymorphism allows objects of different classes to be treated as objects of a common superclass. It enables the same method or function to behave differently based on the type of object it is called upon. Polymorphism can be achieved through method overriding (changing the behavior of a method in a subclass) and method overloading (defining multiple methods with the same name but different parameter lists).

d. Abstraction: Abstraction focuses on representing essential features of an object while hiding irrelevant details. It provides a simplified view of the object's behavior and characteristics, making it easier to work with complex systems. Abstraction allows developers to build models and interfaces that describe the functionalities without specifying their implementation.

Source: GeeksforGeeks -
https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/?ref=gcse

2. Relationship between a Class and an Object:

In object-oriented programming, a class and an object have a close relationship. A class is a blueprint or a template for creating objects, defining the properties (attributes) and behaviors (methods) that the objects of that class will have. It acts as a blueprint from which multiple instances of objects can be created. On the other hand, an object is a specific instance of a class that has its own unique data and can perform actions based on the defined methods.

The relationship can be compared to a cookie cutter and the cookies it produces. The class is like the cookie cutter, defining the shape and features that all cookies will have. Each individual cookie made using that cookie cutter is an object, possessing its own unique taste (data) and can be eaten (perform actions).

Source: TutorialsPoint -
https://www.tutorialspoint.com/java/java_object_classes.htm