Lec 13 OOPS:

Object-Oriented Programming (OOP) is a way of writing computer programs by organizing code into "classes" that group related data and actions, making programs easier to understand, manage, and expand. A class is just a blueprint or plan, and only when you create an "object" from that class does it take up memory and become something you can use in your program.

Introduction and Java Basics

Early programming concepts are reviewed, including the difference between machine-dependent and portable languages, and how Java was designed to be platform-independent. Java's features and architecture (JDK, JRE, JVM) are also introduced.

Object-Oriented Programming (OOP) Overview

OOP is explained as a programming paradigm (not a language or system) that uses real-world objects as the basis for code. OOP includes concepts like class, object, encapsulation, polymorphism, abstraction, and inheritance.

Real-World Analogy for OOP Concepts

OOP concepts are best understood by relating them to real-world entities. For example, classes are like categories (e.g., types of train tickets or classrooms), and objects are like actual things (e.g., a specific ticket or classroom). Pakistan bhuka pyasa, hindustan hara bhara → sinx, cosx and tanx analogy.

The Role of Classes in OOP

In both real life and programming, a "class" is used for grouping or categorizing things. In Java, classes help organize code by grouping related data and actions. This makes programs more organized and easier to manage.

Object-Oriented vs. Object-Based Languages

Not all languages that use objects are fully object-oriented. A pure OOP language treats everything as an object, while object-based languages may lack features like inheritance or polymorphism. Java is not purely object-oriented because it supports primitive data types, but it can convert them to objects using wrapper classes.

The Importance of Classes: Practical Example

Using a calculator application as an example, the difference between procedural programming (like C) and OOP (like Java) is shown. OOP organizes related functions into classes, improving readability and maintainability, especially as programs grow larger.

Naming and Categorizing Classes and Packages

Classes and packages should have meaningful names that reflect their purpose. Packages are used to group related classes, much like rooms in a house or sections in a library, which helps keep large projects organized.

Memory Allocation: Classes vs. Objects

Creating a class does not use up memory in RAM; it only exists as a blueprint on disk. Only when you create an object from a class does memory get allocated, making the object a real, usable thing in your program.

Blueprint Analogy for Classes and Objects

A class is like a blueprint for a house—it doesn't exist physically until you build it. Similarly, an object is the actual house built from the blueprint, existing in memory and able to perform actions.

Object Creation and Real-Time Existence

An object is an instance of a class, meaning it is the real, concrete version of the blueprint (class) that exists in memory. Only after creating an object can you use its data and methods.

Recap and Next Steps

Understanding classes as blueprints and objects as real instances is fundamental to OOP. The session ends with a preview of learning about object creation using the "new" operator and deeper OOP concepts in future lessons.