Lec 14 OBJECT:

This lesson explains the core concept of objects in Java, showing how classes act as blueprints and objects are actual usable items in memory. It covers object characteristics (state, behavior, identity), memory management, and how Java handles object creation, usage, and cleanup through garbage collection.

What is a Class and an Object?

A class in Java is a blueprint or template that defines the structure and behavior for objects, but does not take up memory until an object is created. An object is a real instance of a class, occupying memory and having physical existence in a program.

Objects as the Basic Unit in Java

In object-oriented programming, the object is the most fundamental unit. You cannot perform object-oriented programming without creating and using objects.

Characteristics of Objects

Every object in Java has three main characteristics: state (data stored in variables), behavior (actions

performed through methods), and identity (a unique reference or address).

Instance Variables and Methods

Instance variables represent the state of an object (like weight, height, color in a Dog object). Methods represent the behavior (like barking or eating).

Object Identity and Reference Variables

When an object is created, Java assigns it a unique identity using a hash code, not a direct memory address, for security reasons. Reference variables store this identity to access the object.

Static Typing and Reference Variables

Java is statically typed, so the type of object a reference variable can hold must be declared. Reference variables point to objects and are used to access their data and methods.

Objects as First-Class Citizens

In Java, objects are first-class citizens, meaning they can be stored in variables, passed as parameters to methods, and returned from methods, just like basic data types.

Multiple Objects from One Class

Many objects can be created from the same class blueprint. Each object has its own memory location and state, so changes to one object do not affect others.

Java Memory Management Basics

Memory management in Java involves allocating memory to objects when they are created and deallocating it when they are no longer needed. The JVM handles this automatically.

Class Loading and Bytecode Verification

Before an object can be created, the class is loaded into the JVM by the class loader and its bytecode is verified for security and correctness.

Metadata and Method Area

Metadata about classes (like class name, method names, modifiers) is stored in a special area of memory called the method area. This happens before object creation.

Heap Memory Allocation for Objects

The JVM allocates memory for object instance variables in the heap memory. After allocation, default values are set, and constructors can customize these values.

Reference Variables and Stack Memory

Reference variables, which hold the identity of objects, are usually stored in the stack if they are local variables. They allow access to objects in the heap.

Object Usage and Method Access

Once an object is referenced, its fields and methods can be accessed using the reference variable. The constructor is called automatically when the object is created.

Garbage Collection and Unreachable Objects

If an object becomes unreachable (no reference points to it), Java's garbage collector automatically reclaims its memory, preventing memory leaks.

Full Object Lifecycle Recap

The full lifecycle includes class loading, object creation, reference assignment, object usage, and eventual cleanup by garbage collection when the object is no longer needed.