Lec 11 Recursion:

This session covers key Java programming concepts: recursion (methods calling themselves), how classes are organized using packages, the difference between local and instance variables, and the basics of type casting and character encoding (ASCII vs Unicode). These concepts are essential for writing organized, efficient, and globally compatible Java programs.

Recap of Methods and Introduction to Recursion

Methods can be created with or without parameters and return types. Recursion is introduced as a technique where a method calls itself, often used to solve complex problems like backtracking and generating sequences (e.g., Fibonacci).

What is Recursion?

Recursion is a programming technique where a method repeatedly calls itself to solve smaller instances of a problem. It's important for logical thinking and is foundational in many coding tasks.

Software Organization and Packages

Large programs are divided into multiple classes, often grouped by functionality (like payment, product, search) into packages. Packages help categorize and manage code, similar to organizing items in different rooms at home.

Naming Packages and Unique Identification

Packages are often named using domain names in reverse (e.g., com.start) for uniqueness. This convention prevents naming conflicts and helps identify where code originates, especially in larger projects.

Writing and Calling Methods in Java Classes

To call a non-static method, an object of the class must be created. If a method is only called once, the object can be created inline; otherwise, a reference variable is used to call multiple methods.

Recursion and Stack Overflow

If a recursive method lacks a stopping condition (base case), it calls itself indefinitely, causing a stack overflow error. Adding a condition limits recursion and prevents errors.

Types of Variables in Java

Java variables are classified as local (declared inside methods), instance (declared inside the class but outside methods), and static (not covered in detail here). Local variables exist only within their block; instance variables belong to objects and can have access modifiers.

Access Modifiers and Variable Scope

Access modifiers (public, private, default) control where instance variables and methods can be accessed. Public allows access everywhere, private restricts to within the class, and default allows access within the package.

Initialization and Default Values

Local variables must be explicitly initialized before use and have no default value. Instance variables get default values based on their data type (e.g., int defaults to 0, boolean to false).

Type Casting in Java

Type casting is converting one data type to another. Implicit (widening) casting occurs automatically when moving from a smaller to a larger data type (e.g., byte to int). Explicit (narrowing) casting requires manual intervention and is needed when converting from larger to smaller types.

Character Encoding: ASCII vs Unicode

Character encoding converts characters to numeric values for digital storage and transmission. ASCII supports 128 English characters and symbols, while Unicode supports thousands of characters from many languages and symbol sets. Java uses Unicode, making it suitable for internationalization.

Why Characters Don’t Always Fit in Smaller Types

Unicode characters can have large numeric values that don’t fit into smaller types like byte or short, which is why characters are typically stored in int or larger types in Java.

Conclusion and Next Steps

The session wraps up by emphasizing the importance of understanding implicit and explicit type casting, and

hints at covering explicit casting in future lessons.