Lec 12 Stack Overflow:

The session explains type casting in Java, focusing on widening (implicit) and narrowing (explicit) conversions, and the risks of data loss when converting larger types to smaller ones. It also introduces the stack data structure, its role in method execution, and explains stack overflow errors in recursion, including how stack size can be managed in Java.

Type Casting: Widening and Narrowing

Type casting is converting one data type into another. Widening (implicit casting) happens automatically when a smaller data type is stored in a larger one (e.g., byte to int), while narrowing (explicit casting) requires extra code and occurs when a larger data type is converted to a smaller one (e.g., int to byte).

Unicode and Character Limits

The char data type in Java can store Unicode values up to 65535. If a value exceeds this, wrapper classes like Character are needed to handle more complex or larger Unicode representations, such as emojis.

Widening Limitations

Not all type conversions are allowed. For example, a char cannot be directly cast to byte or short through implicit casting; explicit casting is required to avoid errors.

Narrowing and Data Loss

Narrowing involves converting a large data type to a smaller one, which can cause data loss. For example, converting a large int to a byte may result in negative or incorrect values due to overflow.

Order of Data Types and Narrowing Examples

Data types can be ordered by size: double > float > long > int > short > byte. Converting from a larger to a smaller type (narrowing) can be done, but may result in information loss or unexpected results.

Narrowing with Characters and Unicode

Assigning an integer to a character variable gives the Unicode character for that integer value. This is a practical way to explore Unicode mappings in Java.

Practical Importance of Type Casting

Type casting is essential in real-world applications, such as web forms, where input data (like age or password) is received as strings and must be converted to appropriate types for processing.

Local and Instance Variables in Recursion

Variables can be local (used within a method) or instance (used within a class). Their scope and access control (like private) affect how they behave in recursive methods.

Stack Overflow in Recursion

A stack overflow error occurs when recursive calls exceed the stack's memory limit. This happens because each method call adds a new "activation record" to the stack, and if the base case is missing or recursion is too deep, the stack fills up and crashes the program.

Stack Data Structure Explained

A stack is a linear data structure that follows Last-In-First-Out (LIFO) order. Inserting is called "push" and removing is called "pop." Stacks are used for managing method calls and undo operations in text editors.

Stack in Memory and Real-World Application

The stack is stored in RAM and is used for temporary data like method calls. Real-world uses include undo features in text editors, where each action is pushed onto the stack and undone in reverse order.

Stack and Method Activation Records

Each running method has an activation record stored on the stack. When a method finishes, its record is removed. In recursion, each call adds another record, leading to stack overflow if not managed.

Stack Size and JVM Configuration

The default stack size in Java depends on the operating system and JVM (e.g., 1MB on Windows, 320KB on Linux). The stack size can be increased using the -Xss flag in Java, which allows deeper recursion before overflow occurs.

Demonstrating Stack Size Increase

Increasing the stack size allows a program to handle more recursive calls before overflowing. This is

demonstrated by running Java with the -Xss option and observing higher recursion limits.

Introduction to Method Overloading and OOP Preview

Method overloading (same method name, different parameters) enables compile-time polymorphism in Java. Understanding type casting is important before learning overloading, as errors can occur if types are not managed properly. The next topic will be Object-Oriented Programming (OOP).