Lecture Summary: Stack vs Heap in Java

Generated by LectureMate

February 22, 2025

1 Introduction

This lecture focused on the differences between stack and heap memory in Java, including how objects and primitive types are stored, the implications of passing by reference, and the concepts of shallow and deep copies.

2 Announcements

2.1 Tuples Conference

Catherine from Type Sig announced the upcoming Tuples conference, which will feature 12 speakers discussing topics in theoretical computer science and programming languages. The conference is scheduled for February 23rd and will include two tracks: general theory and programming languages.

2.2 Flexible Learning Week

Colton, the TA for the course, reminded students that there will be no lectures, tutorials, or labs during the upcoming Flexible Learning Week.

3 Memory Management in Java

3.1 Stack Memory

- Stack memory is used for local variables and is transient (temporary).
- Each method call creates a new stack frame, which contains the method's local variables.
- Primitive types (e.g., int, double) are stored on the stack.
- Stack memory is finite; excessive use can lead to stack overflow, especially with recursion.

3.2 Heap Memory

- Heap memory is used for storing objects.
- When the `new` keyword is used, memory is allocated on the heap.
- Variables that reference objects are stored on the stack, while the actual objects reside in the heap.

4 Objects and References

4.1 Passing Objects

- Objects are passed by reference, meaning that when an object is passed to a method, the method receives a reference to the original object, not a copy.
- Modifying an object through one reference affects all references to that object.

4.2 Example with Circle Class

- A `Circle` class was introduced with methods to calculate area, enlarge the circle, and check equality.
- Demonstrated how objects can be compared using `==` (reference equality) and `equals()` (value equality).

5 Memory Management Concepts

5.1 Garbage Collection

- Java automatically manages memory through garbage collection, which cleans up unused objects in the heap.
- Unlike C, where memory must be manually managed, Java's garbage collector helps prevent memory leaks.

5.2 Shallow vs Deep Copy

- A shallow copy creates a new reference to the same object, while a deep copy creates a new object with the same values.
- Demonstrated how to implement both shallow and deep copies using arrays of objects.

6 Conclusion

The lecture concluded with a summary of key concepts: stack vs heap, object references, side effects, garbage collection, and the importance of understanding shallow and deep copies in Java programming.