Lec 33 Multi-Threading Part 4 :

- The synchronized keyword helps in putting lock on resources by a particular thread.
- StringBuffer is synchronized (means all the methods of StringBuffer has synchronized keyword in them, if we check the StringBuffer class in java.lang package).
- Run method is not synchronized.

This lesson explains the concept of race conditions in Java multithreading and how synchronization prevents unexpected behavior when multiple threads access shared resources. It highlights the practical use of synchronized methods and thread-safe classes like StringBuffer to ensure data consistency, especially in applications like banking and gaming.

Introduction to Thread Join and Priority

The join method in Java is used to pause the execution of the current thread until another specified thread finishes its work. This is essential when you want one thread to complete before another continues, ensuring correct sequencing in multithreaded programs.

Understanding Race Conditions

A race condition occurs when two or more threads access and modify shared data simultaneously, leading to unpredictable and incorrect results. This typically happens when threads "race" to update the same resource without proper coordination.

Real-World Example: Bank Account and Debit Cards

If a bank account is accessed by two debit cards (threads) at the same time to withdraw money, both may try to update the balance simultaneously, causing unexpected behavior like incorrect balances. This demonstrates how real-world scenarios can suffer from race conditions when shared data is not protected.

Mutation and Thread Safety

Mutation refers to changing data. In a multithreaded environment, if multiple threads mutate the same data without control, unexpected issues arise. Therefore, mutations should not occur in a non-synchronized (unprotected) environment.

Applications Affected by Race Conditions

Banking, gaming, and even email systems can face problems if multiple threads or users change shared data at the same time. Sometimes, systems allow

certain mutations (like email deletion syncing across devices), but in other cases, such as gaming accounts, simultaneous mutations are restricted to avoid errors.

Coding Example: Simulating Race Condition

By creating two threads representing debit cards and repeatedly deducting money from a shared account, the code demonstrates how running threads in parallel without synchronization can lead to incorrect final balances due to overlapping operations.

Observing Unexpected Behavior

When both threads deduct from the balance many times, the expected result is not always achieved. Sometimes the balance is correct, but often it's not, depending on how the threads overlap, clearly showing the race condition in action.

Solution: Synchronization

Synchronization ensures that only one thread can access a critical section of code (like modifying a balance) at a time. Using the synchronized keyword on methods or code blocks prevents data corruption by locking the resource during use.

Synchronized vs. Non-Synchronized Environments

Ways to Achieve Synchronization in Java

Java provides synchronized methods and synchronized blocks to protect code sections. Synchronized methods lock the entire method, while synchronized blocks allow more precise control, locking only specific parts of code.

String Mutation and Thread Safety

Modifying strings in a multithreaded environment can lead to problems because standard String objects are immutable and not thread-safe. Using classes like StringBuffer (which is synchronized) ensures that multiple threads can safely mutate the same string.

Choosing Between Synchronized and Non-Synchronized Classes

StringBuffer is synchronized and safe for use by multiple threads, while StringBuilder is not synchronized and is faster but unsafe in multithreaded scenarios. The choice depends on whether thread safety or speed is more important for the application.

Performance Trade-Offs of Synchronization

Synchronization can slow down performance because only one thread can access the critical section at a time, causing others to wait. However, it is necessary for correctness in many cases, such as banking or shared printers, where data integrity is more important than speed.

Synchronized Methods vs. Synchronized Blocks

Synchronized methods lock the whole method, while synchronized blocks can be used to lock only the required part of the code, offering more flexibility and potentially better performance by reducing the locked section.--- If you need further clarification on any section or wish to explore more advanced synchronization techniques, consider how different synchronization strategies might affect complex, real-world systems.