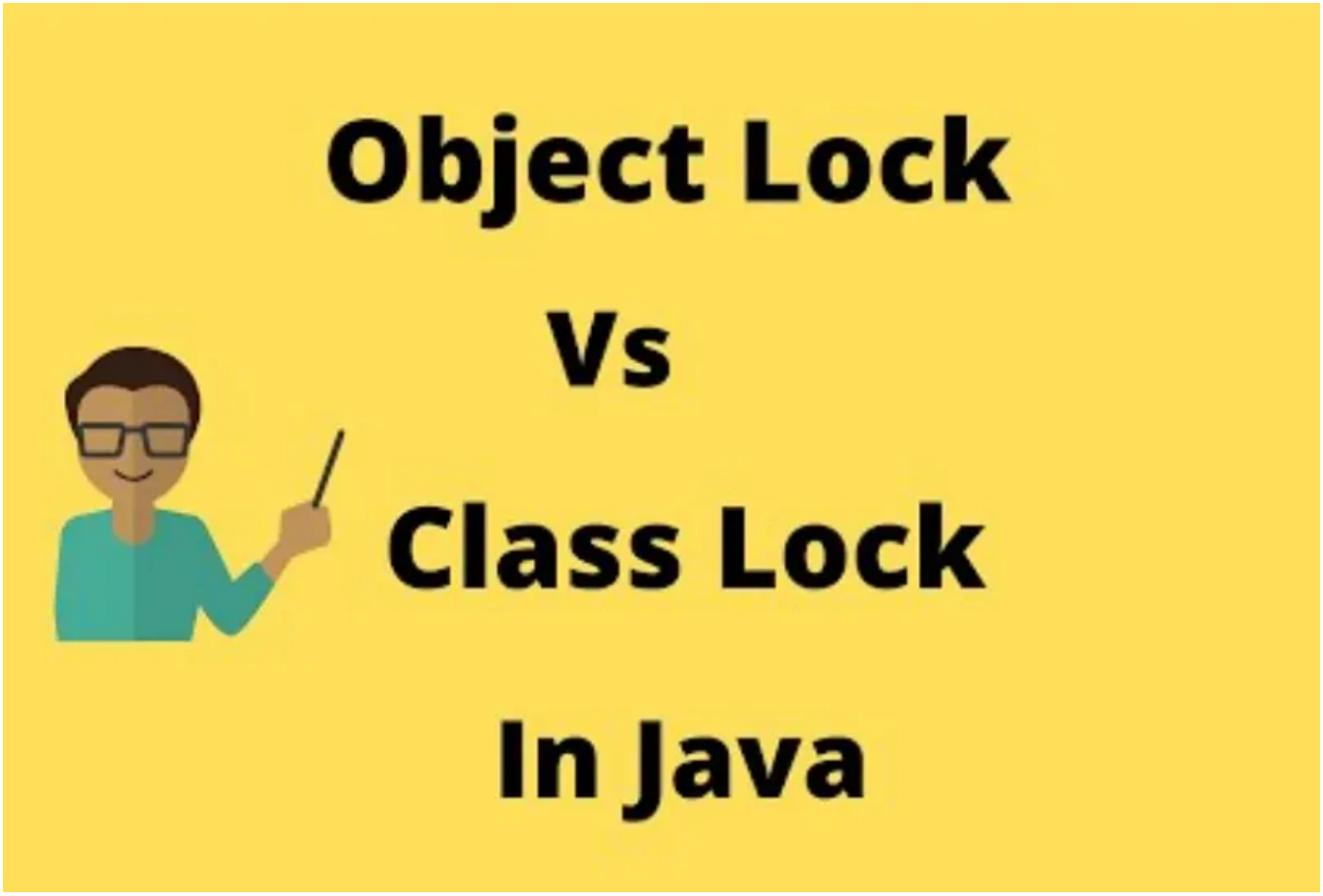
✨ Member-only story

Object-Level and Class-Level Locking in Java

FullStackTips · Follow

4 min read · Aug 15, 2023



Object-level vs Class level locking in Java

Object-Level Locking

In the parallel universe of multithreaded programming, chaos reigns supreme without proper synchronization. Object-level locking in Java orchestrates the cacophony, ensuring that threads march to the same rhythm.

1. Types of Object-Level Locks

Object-level locks in Java can be manifested in two primary forms:

block of code.

2. Synchronized Methods

A synchronized method allows only one thread to execute the method at a time on the same object.

Example: A Ticket Booking System

Imagine a movie theater where only one ticket can be booked at a time for a particular seat.

```
public class TicketBooking {
    private int seatsAvailable = 10;

    public synchronized void bookTicket() {
        if (seatsAvailable > 0) {
            seatsAvailable--;
            System.out.println("Ticket booked successfully!");
        } else {
            System.out.println("No seats available!");
        }
    }
}
```

- **How It Works:** When a thread calls `bookTicket`, it acquires an intrinsic lock on the object, preventing other threads from accessing the synchronized method on the same object simultaneously.

3. Synchronized Blocks

A synchronized block provides finer control, allowing synchronization of specific parts of a method.

Example: A Shared Library System

Consider a library where multiple students can borrow books, but only one student can update the library catalog at a time.

```
public class Library {
    private int booksAvailable = 100;

    public void borrowBook() {
        synchronized (this) {
            if (booksAvailable > 0) {
                booksAvailable--;
                System.out.println("Book borrowed successfully!");
            } else {
                System.out.println("No books available!");
            }
        }
        // Other non-synchronized code here
    }
}
```

- **How It Works:** The synchronized block (`synchronized (this)`) allows only one thread at a time to execute the block of code related to borrowing a book. Other non-synchronized parts of the method can be accessed by multiple threads simultaneously.

4. Comparing Synchronized Methods vs. Blocks

- Methods: Suitable when the entire method needs to be synchronized.
- Blocks: Offers more granular control, synchronizing only the necessary parts, potentially improving performance.

Class-Level Locking

In the world of multithreading, class-level locking is like a master key that controls access to shared class resources. Only one thread at a time can

access the synchronized portions of the class, creating an ordered and reliable environment.

1. Types of Class-Level Locks

Class-level locks are generally manifested in two forms:

- Synchronized Static Methods: By using the `synchronized` keyword with a static method.
- Synchronized Blocks on Class Literal: Using the `synchronized` keyword on the class literal itself (`ClassName.class`).

2. Synchronized Static Methods

When a method is static and synchronized, only one thread can execute that method across all instances of the class.

Example: A Global Conference Room Booking System

Imagine a global company with a conference room that can be booked by employees from any branch.

```
public class ConferenceRoom {
    private static int roomsAvailable = 5;

    public static synchronized void bookRoom() {
        if (roomsAvailable > 0) {
            roomsAvailable--;
            System.out.println("Room booked successfully!");
        } else {
            System.out.println("No rooms available!");
        }
    }
}
```

- **How It Works:** Any thread attempting to call `bookRoom` must acquire the lock on the `ConferenceRoom` class itself, not on an instance.

3. Synchronized Blocks on Class Literal

You can also synchronize a specific block of code using the class literal, providing more granular control.

Example: A Global Printing Service

Consider a company with a centralized printing service shared across multiple departments.

```
public class PrintingService {
    private static int printCredits = 1000;

    public void printDocument() {
        synchronized (PrintingService.class) {
            if (printCredits > 0) {
                printCredits--;
                System.out.println("Document printed successfully!");
            } else {
                System.out.println("No print credits available!");
            }
        }
        // Other non-synchronized code here
    }
}
```

- **How It Works**: The synchronized block (`synchronized (PrintingService.class)`) locks on the class, not the instance, ensuring that only one thread at a time can execute the block related to printing.

4. Comparing Synchronized Static Methods vs. Blocks

- Methods: Suitable when the entire static method must be synchronized across all instances.
- Blocks: Offers more granular control, synchronizing only the necessary parts of a method, allowing other parts to run concurrently.

There are other types of locking mechanisms such as `RetrantLocks` in Java. Refer to the story for more information.

Summary:

In the world of Java programming, object-level and class-level locking are key tools for managing multiple tasks at once. Object-level locking helps control access to individual items, while class-level locking oversees shared resources. Together, they make sure that everything runs smoothly, without clashes or confusion. Whether you're a beginner or an expert, understanding these concepts can help you write better, more reliable code. It's like having traffic lights at busy intersections, guiding the flow and keeping everything in order.

Thanks for reading!!

Java Concurrency Multithreading Synchronization Programming

Written by FullStackTips

Follow

747 Followers

I am full stack developer with over 15 years of experience in various programming languages. https://medium.com/@fullstacktips/membership

More from FullStackTips

FullStackTips

Java 'Arrays' Class: Most Used Methods with Code Examples

Arrays utility class and its methods.

· 3 min read · Feb 19, 2023

54

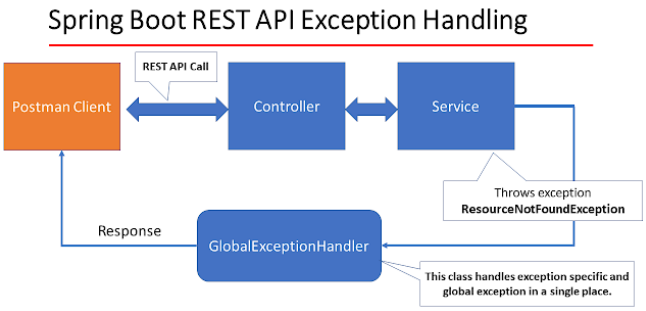
FullStackTips

AWS Redis Elasticache: How to Connect from Your Local...

I hope you already know what is AWS Redis Elasticache, if not, please go the story about...

· 4 min read · Jan 14, 2023

7



FullStackTips

Exception Handling in Spring Boot: Using @ControllerAdvice and...

Spring Boot is a popular framework for developing Java applications. One important...

✦ · 4 min read · Jan 23, 2023

36

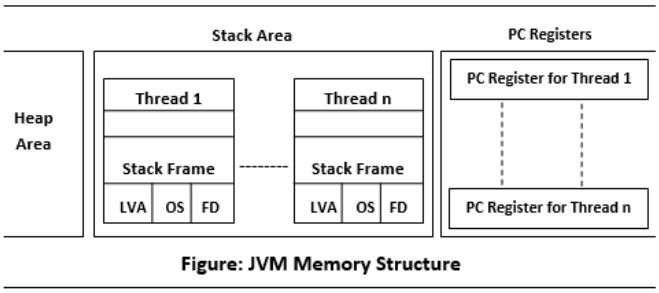


Figure: JVM Memory Structure

FullStackTips

Java Memory Management: Understanding the JVM, Heap,...

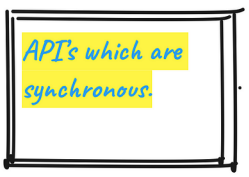
JVM heap memory structure(image source:javatpoint.com)

✦ · 4 min read · Jan 25, 2023

59

See all from FullStackTips

Recommended from Medium



Aesha Shingala in Simform Engineering

Spring Boot caching with Redis

Implementing Caching in Spring Boot Application Using Redis

8 min read · Aug 4, 2023

583

Vikas Taank

The Completable Future Primer.

Dear All please find the collection of all the stories that I wrote on the subject of...

2 min read · Oct 7, 2023

13

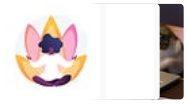
Lists

General Coding Knowledge

20 stories · 798 saves

Coding & Development

11 stories · 384 saves

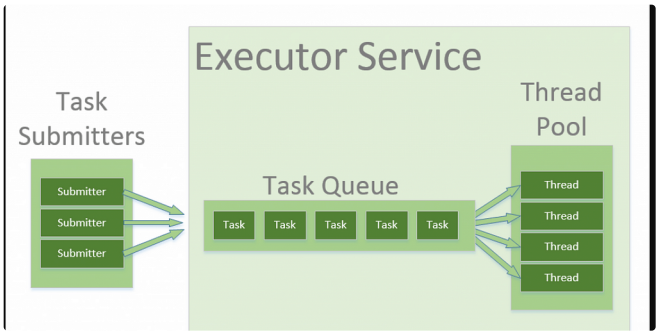


Stories to Help You Grow as a Software Developer

19 stories · 714 saves

ChatGPT

23 stories · 397 saves



Naveen Metta

Unveiling the Executor Framework in Java: A Comprehensive...

Introduction:

5 min read · 6 days ago

Reetesh Kumar

Java Multithreading Introduction

Introduction

9 min read · Jan 8

Shubham Kumbhar

Java Collections Interview Questions—Part 1

1. What are the main interfaces in the Java Collections Framework?

2 min read · Aug 10, 2023

9

Ramit Raj

Sorting in Java using Streams.

We will learn how to sort a list of objects based on a field name in ascending or...

1 min read · Dec 19, 2023

22

2

2

See more recommendations