CONCURRENCY ISSUES AND SYNCHRONIZATION - I

SYNCHRONIZATION

IN JAVA THREADING SUPPORT, THREADS MOSTLY COMMUNICATE WITH EACH OTHER VIA SHARED OBJECTS OR SHARED MEMBER VARIABLES WITHIN THE SAME OBJECT

THREE TYPES OF COMPLICATIONS CAN ARISE FROM THIS..

THREAD INTERFERENCE

DIFFERENT THREADS ACCESS THE SAME DATA

MEMORY CONSISTENCY ERRORS

A THREAD SEES A STALE (INCONSISTENT) VALUE OF A VARIABLE

THREAD CONTENTION

THREADS GET IN EACH OTHER'S WAY, AND SLOW DOWN – OR SOMETIMES EVEN HAVE TO BE KILLED BY JAVA

THREAD INTERFERENCE, AND MEMORY CONSISTENCY ERRORS

IF TWO THREADS ACCESS THE SAME VARIABLE, ITS POSSIBLE FOR THEM TO GET IN EACH OTHER'S WAY

THAT'S BECAUSE JAVA MIGHT SWITCH EXECUTION FROM ONE THREAD TO ANOTHER EVEN MIDWAY THROUGH A SIMPLE, SEEMINGLY ATOMIC INSTRUCTION

FOR INSTANCE TWO THREADS INCREMENTING THE SAME VARIABLE COULD SIMPLY 'LOSE' ONE OF THE TWO INCREMENTS

"SYNCHRONIZED" KEYWORD

SOLUTION? MAKE SURE A SECTION OF CODE IS ONLY ACCESSED BY ONE THREAD A TIME

RESTRICTING ACCESS TO AN OBJECT OR A VARIABLE - AKIN TO LOCKING THE VARIABLE SO ONLY THREAD CAN ACCESS AT A TIME - IS A POWERFUL CONCEPT USED WIDELY IN COMPUTER SCIENCE, ESPECIALLY IN DATABASES

LOCKING VARIABLES CORRECTLY CAN ELIMINATE THREAD INTERFERENCE AND MEMORY CONSISTENCY ERRORS

..BUT IT SLOWS DOWN PERFORMANCE, AND CAN LEAD TO THREAD CONTENTION ISSUES (STARVATION, LIVELOCK, DEADLOCK..)

THE "SYNCHRONIZED" KEYWORD

EVERY OBJECT IN JAVA HAS A LOCK ASSOCIATED WITH IT

THIS LOCK IS CALLED THE INTRINSIC LOCK OR MONITOR

THIS LOCK IS USUALLY ALWAYS OPEN, I.E. ANY NUMBER OF THREADS CAN ACCESS THE OBJECT SIMULTANEOUSLY

IF SOME OTHER THREAD CURRENTLY HOLDS THAT LOCK, THE CURRENT THREAD MUST WAIT ITS TURN

BUT ITS POSSIBLE TO SPECIFY THAT A THREAD CAN ONLY EXECUTE A SECTION OF CODE ONCE IT HAS ACQUIRED THE LOCK ON SOME OBJECT

THIS IS ACHIEVED USING THE "SYNCHRONIZED" KEYWORD