Contents

Multithreading

MULTITHREADING

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilisation of CPU. Each part of such a program is called a thread. So, threads are light-weight processes within a process.

Challenge 1 : What is the Main thread ?
HINT: default thread initialised by JVM on running main function . This is the thread in which all of your code runs .

Challenge 2 : Why multithreading : Parallelism ? or Sequential processing ?

Challenge 3 : How to print the name of the current running thread ?
HINT : Thread.currentThread().getName()

Challenge 4 : If we call a function , do we launch a new thread ? Lets now create our own custom thread named MyThread and spawn it .
HINT : Threads can be only be created by using two mechanisms :
(i) Extending the Thread class
(ii) Implementing the Runnable Interface

Challenge 5 : What is a native function ?
HINT : currentThread() inside Thread class gets its definition from Hardware OS library.

Challenge 6 : How to get the number of processors available in your machine ? How many max threads can run simultaneously in your system ?
HINT :
System.out.println(Runtime.getRuntime().availableProcessors());
System.out.println("Total memory available to JVM "+
Runtime.getRuntime().totalMemory()
System.out.println("Total bytes of memory used by JVM :
"+Runtime.getRuntime().totalMemory() -
Runtime.getRuntime().freeMemory());

Challenge 7: In Thread class definition , why is start0 a private function and start a public function?

Challenge 8 : Although we are overriding the run function in our custom thread class. If you put a print statement to check the name of a thread in your run function , on calling we see it's a new thread. Then why do we call the start method and not the run method ?

Challenge 9 : Create 2 custom threads and using debugger lets check which spawns faster .

Challenge 10 : https://ide.geeksforgeeks.org/NY98CbN0vk
Which line out 15 and line 28 will be executed first ?
HINT : Think if they are executing sequentially or parallely by two different threads . You can check for multithreading by putting debug points at line 15 and 28.

Challenge 11 : Can the same thread object call start() twice ? HINT : IllegalThreadStateException . We can check this by putting debug points and observing
the threadstatus property .If thread status is not 0 , this means thread has already
started so it throws the exception simply .

Challenge 12 : Which out of the two ways (extending Thread vs implementing
Runnable) is a better way to create Threads ?

Challenge 13 : You are given an array of integers and you need to calculate the factorial of all these .
int[] numbers = {15000, 52000, 80000, 60000, 70000, 6000, 80000, 24000, 40000, 300, 400, 5000, 6000};

Way 1 : Sequential
Way 2 : Using parallel streams (check the order this time while printing and while collecting in list)
Way 3 : Multithreading

Challenge 14 : Consider the following code :

```
int[] num = {1,2,3,6,60,72};
int number = Arrays.stream(num).parallel().filter(n
->n%6==0).findFirst().orElse(-1);
System.out.println(number);
```

The parallel streams ignore order while processing , then do we get the actual first element which is a multiple of 6 or not . Give reason to support your answer.
HINT : Before terminal operation (findFirst here) , the numbers in the stream get ordered according to their position in the array which was streamed .

Challenge 15 : What is the issue with the below Multithreaded solution for the above challenge ?
Incorrect Solution: https://ide.geeksforgeeks.org/yIQPkKOXnD
Corrected Solution: https://ide.geeksforgeeks.org/5YeOHNzLj4

Challenge 16 : The correct solution has used thread.join() . Does this make our solution to be parallel or is it still multithreading and the thread.join() call is not blocking ?

HINT 1 : When we are waiting for a thread to die , are we stopping the processing for the other threads? Execution cant go ahead until the thread on which join is called dies, but Are we stopping other threads processing? (think about it) Thread.join is waiting for a thread to die while other threads are also processing . It's just other threads can't proceed ahead in execution of code until that thread is alive . This is 50% faster than sequential .
HINT 2 : Please have a look at the debug Thread trace with debug point at thread.join()
https://drive.google.com/file/d/19bWesFSJha8aEyOeS93IrfLvDpomr8uK/view?usp=sharing

Challenge 17 : Is parallelStreams always faster than sequential streams ?
HINT : Find the first even number in an array of one million items.

Challenge 18 : Thread vs Process
HINT:
https://www.geeksforgeeks.org/difference-between-process-and-thread/

Challenge 19 : User Threads vs Daemon Threads
HINT :
https://www.geeksforgeeks.org/difference-between-daemon-threads-and-user-threads-in-java/?ref=rp

Challenge 20 : (Homework) Print even and odd numbers in increasing order using two threads in Java
HINT : First learn about wait() and notify() concept
https://www.geeksforgeeks.org/difference-between-wait-and-notify-in-java/
https://www.geeksforgeeks.org/print-even-and-odd-numbers-in-increasing-order-using-two-threads-in-java/?ref=rp

Challenge 21 : Is Concurrency the same as Parallelism ?

HINT : Concurrency is the task of running and managing the multiple computations at the same time .While parallelism is the task of running multiple computations simultaneously. Then what is sequential ?
https://www.geeksforgeeks.org/difference-between-concurrency-and-parallelism/

Challenge 22 : What is the volatile keyword in Java ?
HINT : Volatile keyword is used to modify the value of a variable by different threads. It is also used to make classes thread safe. It means that multiple threads can use a method and instance of the classes at the same time without any problem.
https://www.geeksforgeeks.org/volatile-keyword-in-java/

References:

1. https://www.geeksforgeeks.org/multithreading-in-java/
2. https://www.geeksforgeeks.org/lifecycle-and-states-of-a-thread-in-java/ **[IMP]**
3. https://www.geeksforgeeks.org/difference-between-thread-start-and-thread-run-in-java/ **[IMP]**
4. https://stackoverflow.com/q/8052522
5. https://www.geeksforgeeks.org/joining-threads-in-java/ **[IMP]**
6. https://www.geeksforgeeks.org/java-lang-threadgroup-class-java/
7. https://www.geeksforgeeks.org/biginteger-class-in-java/
8. https://www.geeksforgeeks.org/biginteger-intvalue-method-in-java/
9. https://www.interviewbit.com/multithreading-interview-questions/#is-it-possible-that-each-thread-can-have-its-stack-in-multithreaded-programming
10. https://www.baeldung.com/java-concurrency-interview-questions
11. https://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers
12. https://pediaa.com/what-is-the-difference-between-serial-and-parallel-processing-in-computer-architecture/ **[IMP]**
13. https://www.baeldung.com/java-volatile