Contents

HASHMAP : HashMap<K, V> is a part of Java's collection since Java 1.2. This class is found in the java.util package. It provides the basic implementation of the Map interface of Java. It stores the data in (Key, Value) pairs.

Challenge 1 : Create a Person class with properties as id , age and name . Now create a main function in the Person class and define 2 person objects with same property values .

```
Person person1 = new Person(1, "ABC", 10);

Person person2 = new Person(1, "ABC", 10);
```

What will be the output of :

```
System.out.println(person1.equals(person2));
```

(HINT : because object.equals() checks for the object address for equality .)

Challenge 2 : How to see the hashcode of an object ? (HINT : person1.hashcode())

Challenge 3 : Can we define our own method in order to make `System.out.println(person1.equals(person2));` as true ? (HINT : Override equals method)

Challenge 4 : Define a Hashmap with key as Person and value as Boolean . Lets now put person 1 and person 2 into the hashmap .

```
 hashMap.put(person1, true);

 hashMap.put(person2, true);
```

What is the size of the hashmap now ?

Challenge 5 : Can we put person 1 and person 2 in the hashmap and still get the size of the hashmap as 1 ? (HINT : Firstly , make the hashcode of both person objects same by overriding the hashcode method , Secondly override the equals method .)

Challenge 6 : Suppose now we put another object into the hashmap .

```
Person person3 = new Person(2, "ABC", 15);

hashMap.put(person3, true );

hashMap.put(person2, true);

hashMap.put(person1, false);

boolean ans = hashMap.get(person3);

System.out.println(ans);
```

(i) What is the size of the hashmap now ?

(ii) What should I do to make it size 1 ?

(iii) What is ans which gets printed ?

References :

https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/

https://www.geeksforgeeks.org/internal-working-of-hashmap-java/ [IMP]

MULTITHREADING : Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such 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 start we see its 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 be executed first ?

HINT : Think if they are executing sequentially or parallely by two different threads . You can check for multithreading by putting debug point 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 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

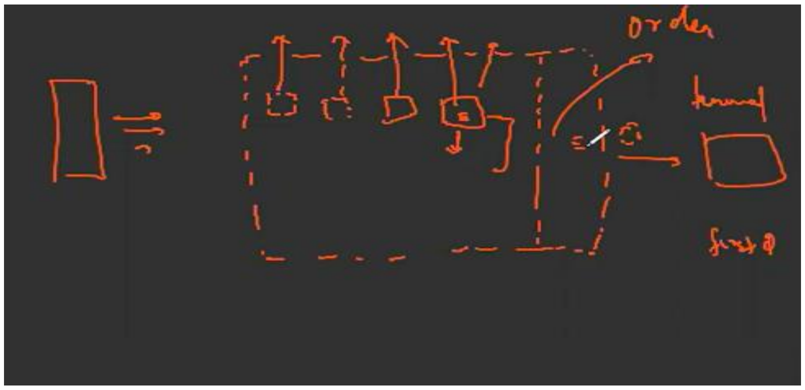
Challenge 14 : Consider the following code :

```
          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 makes our solution to be parallel or is it still multithreading and the thread.join() call is not blocking ?

HINT : 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() in this image : 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/

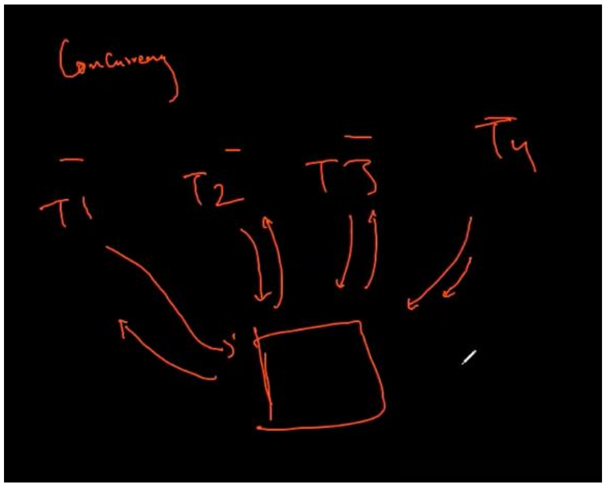
https://www.geeksforgeeks.org/print-even-and-odd-numbers-in-increasing-order-using-two-threads-in-java/?ref=rp

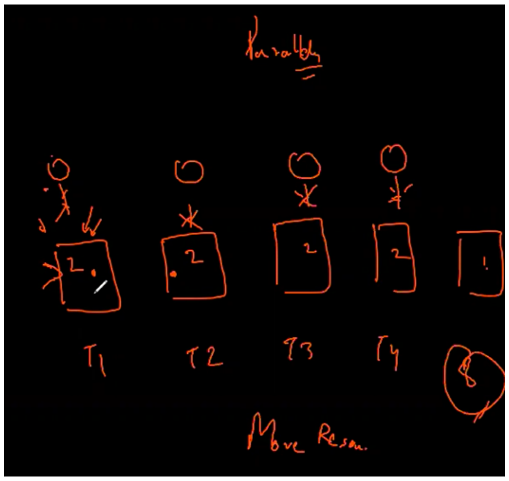
Challenge 21 : Is Concurrency same as Parallelism ?

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 :

https://www.geeksforgeeks.org/multithreading-in-java/

https://www.geeksforgeeks.org/lifecycle-and-states-of-a-thread-in-java/ IMP

https://www.geeksforgeeks.org/difference-between-thread-start-and-thread-run-in-java/ IMP

https://stackoverflow.com/questions/8052522/why-we-call-thread-start-method-which-in-turns-calls-run-method#:~:text=It's%20due%20to%20the%20design,not%20start%20a%20new%20Thread.

https://www.geeksforgeeks.org/joining-threads-in-java/ IMP

https://www.geeksforgeeks.org/java-lang-threadgroup-class-java/

https://www.geeksforgeeks.org/biginteger-class-in-java/

https://www.geeksforgeeks.org/biginteger-intvalue-method-in-java/

https://www.interviewbit.com/multithreading-interview-questions/#is-it-possible-that-each-thread-can-have-its-stack-in-multithreaded-programming

https://www.baeldung.com/java-concurrency-interview-questions

https://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers

https://pediaa.com/what-is-the-difference-between-serial-and-parallel-processing-in-computer-architecture/ IMP

https://www.baeldung.com/java-volatile

LIVE
Java
Backend
Development