

**Saint Petersburg Electrotechnical University LETI**

Faculty of Computer Science and Technology, Department of Computer Science and Engineering

Master of Computer Science and Knowledge discovery

Course name: Parallel Computing

Student Name: Mohammed, Omar Taha

Professor : Natalya V. Razmochaeva

[**Task 1**](#_waux67vvb81y) **4**

[Answers:](#_5ycdu8cqohxv) 4

[**Task 2**](#_psxav8sawwfi) **4**

[Answers:](#_3uhzwmqgxvs0) 4

[**Task 3**](#_qg4uybkre3gc) **4**

[Answers:](#_7yylq2x6royi) 4

[**Task 4**](#_mnl34tcfhjfs) **4**

[Answers:](#_6x85nn92e7bo) 4

[**Task 5**](#_18zhu9a228ts) **5**

[Answers:](#_hd6kb8l5zna5) 5

[**Task 6**](#_t5aw67bywwcq) **5**

[Answers:](#_tiomd4thr0xs) 5

[**Task 7**](#_n70m33t92yrc) **5**

[Answers:](#_3513wrmjjglb) 5

[**Task 8**](#_lc5oatj2n827) **5**

[Answers:](#_fclht6nkmx8w) 6

[**Task 9**](#_p38ye3winlbz) **6**

[Answers:](#_uwwo2wnr2kev) 6

[**Task 10**](#_e3x1fc83h4a) **6**

[Answers:](#_cuktaovaoemx) 6

[**Task 11**](#_lrzsy1gf3854) **6**

[Answers:](#_8tvd8d3lelk3) 6

[**Task 12**](#_928qa7wxkmht) **7**

[Answers:](#_2aohjokv3n0e) 7

[**Task 13**](#_akerheqqhcir) **7**

[Answers:](#_9o98kh1llfsn) 7

[**Task 14**](#_n9brepjd18yd) **7**

[Answers:](#_bm8wlzusj6pj) 7

[**Task 15**](#_5emh8gk8f24i) **7**

[Answers:](#_msc2uzp5ooyw) 7

[**Task 16**](#_i8tr7epcudtd) **8**

[Answers:](#_b2h91u9ayerv) 8

[**Task 17**](#_veai6ex2fg44) **8**

[Answers:](#_d1mrlm6k05ms) 8

[**Task 18**](#_lpxyf9axtkox) **8**

[Answers:](#_65ajyuqwl93z) 8

[**Task 19**](#_n6m68ymgh7j8) **8**

[Answers:](#_n5fydk61ue1b) 9

[**Task 20**](#_dwrds1wrz6an) **9**

[Answers:](#_p9351yuxsse0) 9

[**Task 21**](#_elmuzsj0bxxe) **9**

[Answers:](#_tmnyhprqm5uu) 9

[**Task 22**](#_rc0vcvyem5mn) **9**

[Answers:](#_rdt11sshbv1x) 9

[**Task 23**](#_7glw7pnj3wy2) **10**

[Answers:](#_mshyz4ua5yj0) 10

# Task 1

The task wanted to find the package contains Concurrent hash map collection which is one of the .

## Answers:

* ConcurrentHashMap
* ConcurrentHashMap

# Task 2

The task wanted to make a change in the code to allow the program to terminate even if not all the threads are completed which is related to a property of the thread object which is Deamon property.

## Answers:

* setDaemon(true)

# Task 3

The task wanted to find a way to get the current thread (Currently running thread) to get the name of.

## Answers:

* currentThread()

# Task 4

The task wanted us to make the provided code threadSafe by synchronizing the access of class member methods.

## Answers:

* synchronized
* synchronized
* synchronized

# Task 5

The task wanted us to make some changes in the provided code to terminate after all threads do their own jobs by calling a method of the executor object.

## Answers:

* shutdown()

# Task 6

The task asked how do we wait for a thread to finish it’s job in java multithreading.

## Answers:

* join()

# Task 7

The task asked a method to start a thread after creation.

## Answers:

* start()

# Task 8

The task asked a multiple choice question to find the answer of how RWMutex is implemented in java cuncurrency.

## Answers:

* ReadWriteLock

# Task 9

The task wanted us to make some changes in the code to make it threadSafe by providing a missing package name the class type and constructor of the defined done variable.

## Answers:

* atomic.AtomicBoolean
* AtomicBoolean
* AtomicBoolean

# Task 10

The task wanted us to make some changes in the code to make it threadSafe by completing the missing try catch statement and make the explicit lock unlock after all and automatically.

## Answers:

* try
* finally
* unlock

# Task 11

The task wanted to find the missing package names for ReadWriteLock and ReentrantReadWriteLock to make the code threadSafe.

## Answers:

* ReadWriteLock
* ReentrantReadWriteLock
* ReadWriteLock
* ReentrantReadWriteLock
* w
* lock()
* w
* unlock()
* w
* lock()
* w
* unlock()
* w
* lock()
* w
* unlock()

# Task 12

The task wanted to select the right answer among the provided choices to get the number of processors in JAVA Virtual Machine.

## Answers:

* Runtime.getRuntime().availableProcessors()

# Task 13

The task wanted to have the counter variable always set to 50 when it is printed in the provided thread safe code with missing package names,class types, and calling appropriate member methods and an appropriate Barrier.

## Answers:

* CyclicBarrier
* CyclicBarrier
* CyclicBarrier
* await()

# Task 14

The task is like the previous one but with some differences but with CountDownLatch this time.

## Answers:

* CountDownLatch
* CountDownLatch
* CountDownLatch
* countDown()
* await()

# Task 15

The task wanted us to implement exception handling of thread interruption by interrupting it manually in the catch block.

## Answers:

* isInterrupted()
* interrupt()

# Task 16

The task wanted to make the provided code threadSafe by one of the low level memory synchronization mechanisms.

## Answers:

* volatile

# Task 17

The task wanted to find a legal way to terminate the thread execution.

## Answers:

* isInterrupted()
* interrupt()

# Task 18

The task wanted us to select an option from the given choices as an appropriate answer for java concurrency primitive - Pool.

## Answers:

* Java doesn't have one out of the box, but it can be modeled using one of the concurrent bounded blocking queues, like ArrayBlockingQueue, for example

# Task 19

The task wanted us to select an option from the given choices as an appropriate answer for java concurrency primitive - WaitGroup.

## Answers:

* CountDownLatch

# Task 20

The task wanted us to select an option from the given choices as an appropriate answer for java concurrency primitive - Once.

## Answers:

* ava doesn't have one out of the box, but it can be modeled using static initializer or single value enum

# Task 21

The task is What is the optimal number of threads for the IO intensive tasks?

## Answers:

* Number of available cores / (1 - Blocking coefficient)

# Task 22

The task wanted us to select an option from the given choices as an appropriate answer for java concurrency primitive - Cond.

## Answers:

* Condition
* Object wait/notify/notifyAll

# Task 23

The task is What is the optimal number of threads for the compute intensive tasks?

## Answers:

* Number of available cores
* Number of available cores + 1