PPCP

Exercise 1.1

1. The output value is different each time, because the value depends on the interleavings of the threads, therefore a race condition occurs, and the scheduler choses threads non-deterministically. It can be 20 million or in most cases less than 20 million.

2. The reason why the output value is more likely to be correct, is that the number of interleavings is less than before, increasing the chance of the correct interleavings. But the scheduler still choses threads in a non-deterministic way.

3. It does not change the output value, because none of the assignments are atomic.

4. Using the lock() and unlock() methods we ensure that two threads cannot be executing their critical sections at the same time, in this way we guarantee Mutual exclusion on the shared resource (the counter). The thread that get the lock on the shared resource can enter its critical section and modify the counter (*count = count +1*) then he has to release the lock and so he leaves the critical section; since every thread releases the lock after modifying the counter and does not asks for locks on other resources while he is in the critical section the Absence of deadlocks is guaranteed. A thread has a definite number of iterations and we have a finite number of threads, therefore eventually every thread enters its critical section and this guarantees Absence of starvation. The three properties of the mutual execution problem are ensured by our solution, therefore its ideal and there can't be race conditions.
   .
5. The defined critical section contains the least number of code. Even if we split up the count = count +1 line, to two lines of code, we couldn't leave any of them out of the critical section, because different interleavings lead to different output values. Also in the critical section there should be only the instructions that modify the shared resource, so only the instructions that can lead to a race condition.

Exercise 1.2

2. if the interleaving is like the following:
   t1(1), t2(1), t1(2), t2(2), t1(3), t2(3) (see comments in the code)
   the output will be equal to - - | |
3. We use Java ReentrantLock in order to ensure the correct interleaving of the execution of our code.
   By embedding the code between the two prints (including them) we prevent one thread from printing anything until the other has printed both characters. If we assume that there are no missing pieces, the above output is not possible because there is no thread interleaving that allows us to print -|-|-|| after a correct thread alternation (a - is missing).

Exercise 1.3

2. We used the ReentrantLock to ensure the correct behavior. We used the lock() function before checking the counter, ensuring that the value of the counter will not change after the check. If the value of the counter reaches the value of the MAX_PEOPLE_COVID, we end the thread and release the lock to prevent a deadlock. The use of unlock is also necessary after the modification of the counter.

Exercise 1.4

1. Goetz's definitions of resource optimization, fairness, and convenience see concurrency with the goal of improving system performance . On the other hand, the other three motivations (inherent, exploitation, and hidden) focus seems to be on the inherent, hardware exploitation, and resource sharing aspects of concurrency.

   In particular resource utilization is about efficiently utilizing available system resources to improve overall system performance while inherent concurrency suggests that concurrency is a natural aspect of such systems due to the need to handle multiple input/output operations at the same time.

   We could not think of examples of systems which are included in the categories of Goetz, but not in those in the concurrency note, and vice versa because those categories tend to overlap in real-life examples, their mean difference is in the purpose of the concurrency, but in real examples the purpose depends on how your interpretation of the problem.

2. Some examples are:

   Inherent:

   1. Microsoft Windows, it uses a GUIs, those inherently involve concurrency as they need to handle multiple user interactions simultaneously
   2. Unity, which is a video game engine, which inherently requires high-performance parallelism to manage graphics rendering, multiple outputs, and user input.
   3. Apache HTTP Server which is a web server and so it deals with concurrent requests from multiple clients.

   Exploitation:

   1. Amazon DynamoDB, a Distributed Database which exploits hardware capabilities to enable sharing data among independent computers.
   2. Amazon CloudFront, a CDN which exploit distributed hardware capabilities to optimize content delivery by distributing copies of content across multiple servers globally

3. BitTorrent a p2p network for file sharing, where individual devices communicate and share resources directly with each other without the need for a central server

Hidden:

1. Oracle VM VirtualBox: allow multiple virtual machines (VMs) to share physical resources. Each one operates as if it has sole ownership of these resources, hiding the underlying concurrency.
2. Oracle Database, a DBMS able concurrent access to data by multiple users or applications. They use concurrency control mechanisms to ensure data consistency, hiding it to the user.
3. Microsoft SQL Server, another dbms.