Homework

1- **Concurrent Programming** is a technique in which two or more processes start, run in an interleaved fashion through context switching and complete in an overlapping time period by managing access to shared resources.
Paralel Programming is the process of decomposing a problem into smaller tasks that can be executed at the same time using multiple compute resources.

S.NO	Concurrency	Parallelism
1.	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.
2.	Concurrency is achieved through the interleaving operation of processes on the central processing unit(CPU) or in other words by the context switching.	While it is achieved by through multiple central processing units(CPUs).
3.	Concurrency can be done by using a single processing unit.	While this can't be done by using a single processing unit. it needs multiple processing units.
4.	Concurrency increases the amount of work finished at a time.	While it improves the throughput and computational speed of the system.
5.	Concurrency deals lot of things simultaneously.	While it do lot of things simultaneously.
6.	Concurrency is the non-deterministic control flow approach.	While it is deterministic control flow approach.
7.	In concurrency debugging is very hard.	While in this debugging is also hard but simple than concurrency.

2- **Mutex** is a property of concurrency control, which is instituted for the purpose of preventing race conditions.

Semaphore is a variable or abstract data type used to control access to a common resource by multiple threads and avoid critical section problems in a concurrent system such as a multitasking operating system.

3-

BASIS FOR COMPARISON	ERROR	EXCEPTION
Basic	An error is caused due to lack of system resources.	An exception is caused because of the code.
Recovery	An error is irrecoverable.	An exception is recoverable.
Keywords	There is no means to handle an error by the program code.	Exceptions are handled using three keywords "try", "catch", and "throw".
Consequences	As the error is detected the program will terminated abnormally.	As an exception is detected, it is thrown and caught by the "throw" and "catch" keywords correspondingly.
Types	Errors are classified as unchecked type.	Exceptions are classified as checked or unchecked type.
Package	In Java, errors are defined "java.lang.Error" package.	In Java, an exceptions are defined in"java.lang.Exception".
Example	OutOfMemory, StackOverFlow.	Checked Exceptions : NoSuchMethod, ClassNotFound. Unchecked Exceptions : NullPointer, IndexOutOfBounds.

The Throwable class is the superclass of all errors and exceptions in the Java language. Only objects that are instances of this class (or one of its subclasses) are thrown by the Java Virtual Machine or can be thrown by the Java throw statement.

4- The simple rules that we need to follow to annotate a method with @Scheduled are:

- the method should typically have a void return type (if not, the returned value will be ignored)
- the method should not expect any parameters

@EnableScheduling

@Scheduled(fixedDelay = 1000)

@Scheduled(fixedRate = 1000)

5- @Async has two limitations:
- It must be applied to public methods only.
- Self-invocation — calling the async method from within the same class — won't work.
- The reasons are simple: The method needs to be public so that it can be proxied. And self-invocation doesn't work because it bypasses the proxy and calls the underlying method directly.

6- **High availability** (HA) is a component of a technology system that eliminates single points of failure to ensure continuous operations or uptime for an extended period. High Availability ensures your systems, databases, and applications operate when and as needed.

7- **Entity** is a type that has an identity. The id could be anything but it has to be unique to the system and in fact depending on the subdomain you are in, the identity/entity might change.

Value Objects do not have an identity. They are defined by their attributes instead of an identifier. We can think of Value Objects as a complex attribute of an Entity.

8- **Ubiquitous Language** is a language structured around the domain model and used by all team members within a bounded context to connect all the activities of the team with the software.

9- **A core domain** is what makes an organization special and different from other organizations. An organization cannot succeed (or even exist) without being exceptionally good in their core domain.

A supporting subdomain is a subdomain that is necessary for the organization to succeed, but it does not fall into the core domain category. We may be able to start with an existing solution and tweak it or extend it to your specific needs.

A generic subdomain is a subdomain that does not contain anything special to the organization but is still needed for the overall solution to work. You can save a lot of time and work by trying to use off-the-shelf software for your generic subdomains.

10- **An Anaemic Domain Model** is a model with no logic in it. Domain classes look more like a bunch of public setters and getters without domain logic where the client of the class has control over how to instantiate and modify the class. In these models, the client has to interpret the class purpose and use. Usually, the logic has been pushed to other classes called something like services, helper or manager plus the name of the domain class. With the logic sitting in another class, there is nothing that helps the client to navigate or use the model class.

A good way to address the main disadvantages of Anaemic Domain Models is implementing **a Rich Domain Model**. The main difference with an Anaemic Domain Model is that our domain logic is part of our domain entities, data and behaviour sit together. That logic guides and controls how the entity is instantiated, validated and operated, preventing the client from having entities with an inconsistent state.