Group 8

- **Comparison of the currency models (Procs & Cons)**

Models	**Pros**	**Cons**
Shared state concurrency	Share state among the threads. Different threads can use and share one or more objects between each other.	- Require locking mechanism like synchronization → difficult to manage correctly - Higher risk of concurrency bugs such as race condition and deadlock
Separate state	Do not share any objects or data → Avoids a lot of the concurrent access problems like race condition and deadlock	- Can require more complex design patterns such as message-passing

- **Differences between Concurrency vs Parallelism**

Aspect	**Concurrency**	**Parallelism**
Define	Execute more than one task - at the same time or at least seemingly at the same time	Executing multiple tasks simultaneously
Algorithms	Switching between the different tasks during execution	Split tasks up into smaller subtasks which can be processed in parallel
Hardware	Works on single-core or multi-core processors	Requires more than one CPU or CPU core
Benefit	Improve response time	Reduce execution time

- **Explain the usage of Blocking Concurrency Algorithms and Non-blocking Concurrency Algorithms**

Aspect	**Blocking Concurrency Algorithms**	**Non-blocking Concurrency Algorithms**
Define	Uses locks to ensure that only one thread can access a resource at a time, causing other threads to wait if the resource is already in use.	Uses non-blocking techniques to manage access to shared resources without requiring threads to wait for each other.
Algorithms	Uses locks, semaphores, or synchronized blocks to prevent conflicts.	Uses non-blocking data structures such as Atomic variables (AtomicBoolean, AtomicInteger, AtomicLong and AtomicReference)
Use cases	Suitable for simpler systems with low concurrency needs.	- Reduces the time threads spend waiting. - Ideal for highly concurrent systems requiring low latency.
Complexity	Generally simpler to implement and understand but can lead to deadlocks if not used carefully.	More complex to implement and understand but can be more efficient and avoids deadlocks