Keywords – Asynchronous, Concurrency and Multthreading

concurrency is built around the concept of concurrent tasks. A task is an asynchronous operation. Tasks offer a higher level of abstraction for concurrent code than traditional thread-centric approaches. Tasks can be chained together, meaning that tasks pass their asynchronous result from one to another, where the result of one task is used as if it were a parameter or an intermediate value of another ongoing task. Tasks allow applications to utilize available hardware resources better and scale much more than using raw threads, since tasks can be suspended, awaiting another task to produce a result, without blocking underlying OS-threads. Tasks bring much more productivity to developers by allowing them to focus more on business-logic and less on low-level concepts like thread management and inter-thread synchronization.

While tasks specify *what* actions have to be executed, *executors* are worker-objects that specify *where and how* to execute tasks. Executors spare applications the tedious management of thread pools and task queues. Executors also decouple those concepts away from application code, by providing a unified API for creating and scheduling tasks.

concurrency brings the power of concurrent tasks to the Java world, allowing developers to write highly concurrent applications easily and safely by using tasks, executors and coroutines. By using concurrency applications can break down big procedures that need to be processed asynchronously into smaller tasks that run concurrently and work in a co-operative manner to achieve the wanted result. It also allows applications to write parallel algorithms easily by using parallel coroutines.

main advantages are:

* Writing highly concurrent and parallel applications that scale automatically to use all hardware resources, as needed.
* Mutual Exclusion
* Achieving non-blocking, synchronous-like code easily
* Reducing the possibility of race conditions, data races and deadlocks by using high-level objects with built-in synchronization.

Study the below project.

https://www.geeksforgeeks.org/creating-an-asynchronous-multithreaded-chat-application-in-java/

Revise it with ExceutorService