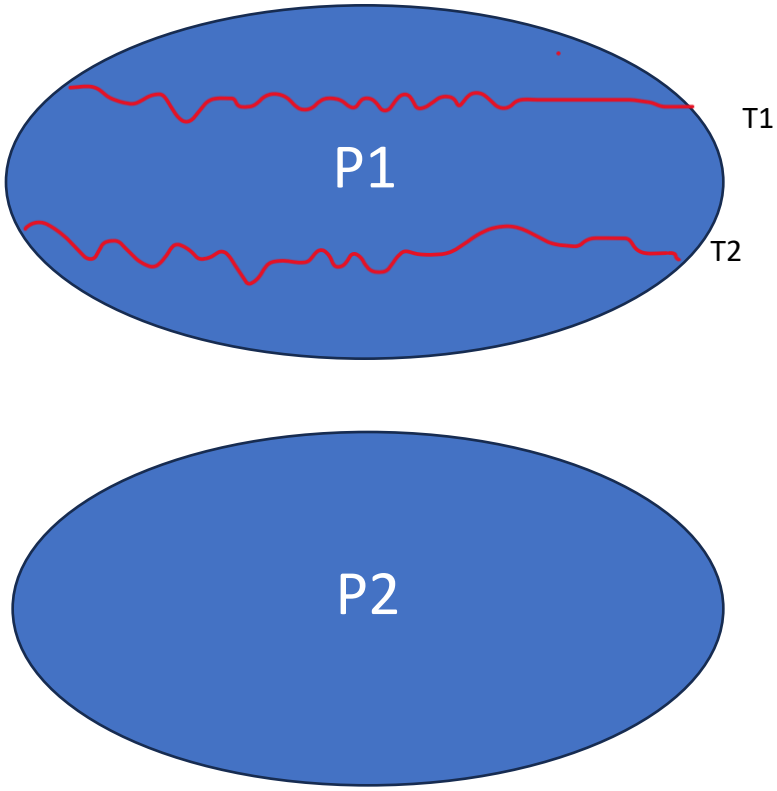

- If each thread takes the memory space of 512 KB – 1 MB,
- If we create 1000 threads, it will take the memory space of 0.5GB – 1GB
- If we create 100000 threads, it will take the memory space of 50-100GB

- That is where the concepts like Thread Pool, Executor etc. were introduced.
- Thread Pool – We're going to keep a few threads in the Thread Pool, which can be fixed or can be dynamic.
- If it is fixed , always there will be so many number of threads will be there in the Thread Pool even when None of the threads will be used. The threads are going to stay idle in the pool.

- If it is dynamic, depending upon the demand, threads will be created in the pool. So the number of threads can grow or shrink depending upon the number of tasks that needs to be performed. If there are no tasks is being performed, then all the threads can be removed from the pool after waiting for a period of time.
- So here we can perform 100000 of tasks using couple of threads in the thread pool without creating 100000 of threads that we would've done in traditional way.
- There are couple of interfaces and classes are available in Java to support thread (Introducing Java 5.0)
    - Executor Interface
    - ExecutorService Interface
    - ScheduledExecutorService Interface
    - Executors Utility Class

Exercise:
Using the concept of thread pools and executors discussed above, how would you design a system to handle thousands of bank transactions efficiently when only one teller is available?