The aim of the program is to solve the dining philosophers' problem using Java threads. The philosophers start walking and take a random amount of time between 1 and 10 seconds. Each philosopher is denoted by a separate thread which sleeps for the time the philosopher is walking towards the table. A barrier implementation ensures that no thread proceeds until all threads (philosophers) reach the table. Each thread is checked and blocked if other threads have not arrived. After reaching the table, each philosopher is in one of three sates: hungry, eating, thinking at a time. When thinking, the philosophers try to think for a random amount of time i.e. the threads sleep for anytime between 0 and 10 seconds. When a philosopher is hungry they tries to take the forks on their left and right. The forks being a shared resource are protected by mutex. If the forks to the left and right are not being used by another thread i.e. left and right philosophers are not eating, then the hungry philosopher is granted access to the shared resource which are the forks. Else, the thread is blocked by the semaphore if forks are not acquired. After eating, philosopher enters the critical region through mutex, puts back forks and changes state to thinking. It also checks if other philosophers to its left and right can now eat and then leaves the critical region. Each philosopher goes through the recurring cycle of thinking, hungry, and eating denoted in the program by the think, takeFork, eat, and putFork functions.