The program is deadlock free thanks to creating a Queue class and instantiating one queue object which is then passed to every philosopher instance. Queue method canPhilosopherEat(Philosopher instance) returns boolean value which represents if philosopher can pick up forks and start eating. Each philosopher before picking up forks has to ask queue instance if it is his turn. Philosopher picks up his forks by obtaining forks' monitor objects using synchronized block of code. Philosophers eat in turns 1 and 3, 2 and 4, ... 5 and 2 then repeat the cycle. Philosopher can eat for a random amount of time. However philosopher's forks can not be picked up before both philosophers in a turn finish eating. To sum up, deadlock free solution was achieved by creating eating in turns "pattern" which is enforced by a Queue instance and ensures the program will be constantly making progress without deadlock appearing.