```
package //wpisz swoj

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Semaphore;

public class ThreadsAndFile {
   public static void main(String[] args) throws InterruptedException {
      String[] files = {"plik1.txt", "plik2.txt", "plik3.txt", "plik4.txt", "plik5.txt"};
      Semaphore[] semaphores = new Semaphore[files.length];
      for (int i = 0; i <semaphores.length; i++){
         semaphores[i] = new Semaphore(1);
      }
      List<Thread> threads = new ArrayList<>();
      int n = 10;
      for (int i = 0; i < n; i++) {
         threads.add(new PhilosopherFile(files, semaphores));
      }
      threads.forEach(Thread::start);
      Thread.sleep(100);
      threads.forEach(Thread::interrupt);
   }
}
```

```
package //wpisz swoj

import java.util.concurrent.Semaphore;

public class PhilosopherWithoutExtraLock extends Thread {
   private Semaphore[] locks;
   private int philosopherID;

   public PhilosopherWithoutExtraLock(Semaphore[] locks, int philosopherID) {
      this.locks = locks;
      this.philosopherID = philosopherID;
   }

   @Override
   public void run() {
      try{
         while (true){
            System.out.println("Mysle " + philosopherID);
            Thread.sleep((long) (7*Math.random()));
            int id1 = philosopherID;
            int id2 = (philosopherID+1) % locks.length;

            int id1sem = id1 > id2 ? id2 : id1;
            int id2sem = id1 > id2 ? id1 : id2;


            locks[id1sem].acquireUninterruptibly();
            locks[id2sem].acquireUninterruptibly();

            System.out.println("Zaczyna jesc " + philosopherID);
            Thread.sleep((long)(5*Math.random()));
```

```
            System.out.println("Konczy jesc " + philosopherID);


            locks[id1sem].release();
            locks[id2sem].release();
          }
        } catch (InterruptedException e) {


        }
    }
}
```