```
// Vahe Karamian - CS 431 - Project 2 - Dinning Philosopher Problem
// Filename: DinningPhilosopher.cpp

#include <stdio.h>
#include <assert.h>

#include "CMcl.h"

enum { NUMBER_PHILOSOPHERS = 5 };

int  GLOBAL_COUNTER = 1;

CMclWaitableCollection collection;

// Table class
class Table
{
  public:
    Table() { m_bContinue = TRUE; };

    // Check to see if you can grab the forks
    void TakeForks(int nId)
    {
      m_cmForks[nId].WaitForTwo( m_cmForks[(nId + 1) % NUMBER_PHILOSOPHERS], TRUE, INFINITE);
    };

    // Drop the forks
    void DropForks(int nId)
    {
      m_cmForks[nId].Release();
      m_cmForks[(nId + 1) % NUMBER_PHILOSOPHERS].Release();
    }

    BOOL Continue(void) { return m_bContinue; };

    void Stop(void) { m_bContinue = FALSE; };

  private:
    CMclMutex m_cmForks[NUMBER_PHILOSOPHERS];
    BOOL m_bContinue;
};

// Philosopher class
class Philosopher : public CMclThreadHandler
{
  public:
    Philosopher( int nId, Table *pt) : m_nId(nId), m_pTable(pt) { return; };

    unsigned ThreadHandlerProc(void)
    {
      Philosophize();
      return 0;
    }

    // Make him get fork eat, drop fork and think
    void Philosophize(void)
    {
      GLOBAL_COUNTER++;
      if( GLOBAL_COUNTER >= 500 )
      {
        m_pTable->Stop( );
        collection.Wait( TRUE, INFINITE);
      }
      else
      {
        while (m_pTable->Continue())
        {
          m_pTable->TakeForks(m_nId);
          Eat();
          m_pTable->DropForks(m_nId);
          Think();
        }
        printf( "Philosopher %d is done.\n", m_nId );
      }
    };

    // Thinking function
    void Think(void)
    {
```

```
      GLOBAL_COUNTER++;
      if( GLOBAL_COUNTER >= 500 )
      {
        m_pTable->Stop( );
        collection.Wait( TRUE, INFINITE);
      }
      else
      {
        printf( "Philosopher %d Thinking.\n", m_nId );
        Sleep( rand() % 100 );
      }
    };

    // Eating function
    void Eat(void)
    {
      GLOBAL_COUNTER++;
      if( GLOBAL_COUNTER >= 500 )
      {
        m_pTable->Stop( );
        collection.Wait( TRUE, INFINITE);
      }
      else
      {
        printf( "Philosopher %d Eating.\n", m_nId );
        Sleep( rand() % 100 );
      }
    };

  private:
    int m_nId;
    Table *m_pTable;
};

// Philosopher thread class
class PhilosopherThread : public CMclThread
{
  public:
    ~PhilosopherThread() { delete m_pPhilosopher; };

    static PhilosopherThread *CreatePhilosopher( int nId, Table *pt)
    {
        Philosopher *pPhilosopher = new Philosopher( nId,pt);
        return new PhilosopherThread(pPhilosopher);
    }

  private:
    Philosopher *m_pPhilosopher;

    PhilosopherThread( Philosopher *pPhilosopher) : CMclThread(pPhilosopher)
    {
        m_pPhilosopher = pPhilosopher;
    };
};

// Starting point
int main( int argc, char *argv[])
{
  Table table;
  CMclThreadAutoPtr Philosophers[NUMBER_PHILOSOPHERS];
  // CMclWaitableCollection collection;

  for (int i = 0; i < NUMBER_PHILOSOPHERS; i++)
  {
    Philosophers[i] = PhilosopherThread::CreatePhilosopher(i, &table);
    collection.AddObject(*Philosophers[i]);
  }

  // check every second to see if # events is > 500 stop if so
  while( GLOBAL_COUNTER < 500 )
    Sleep( 1000 );

  printf( "All done. \n" );

  return 0;
}
```