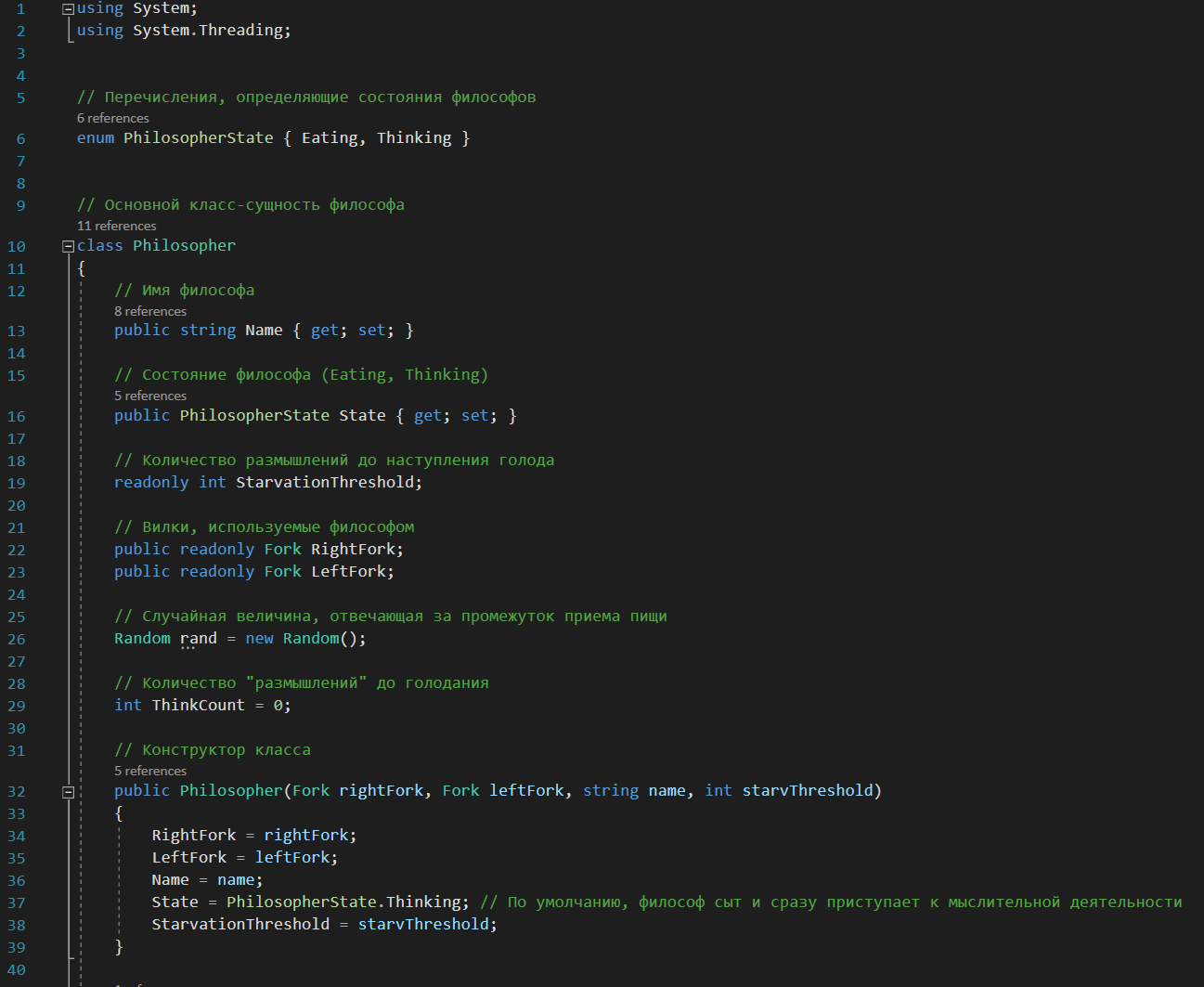
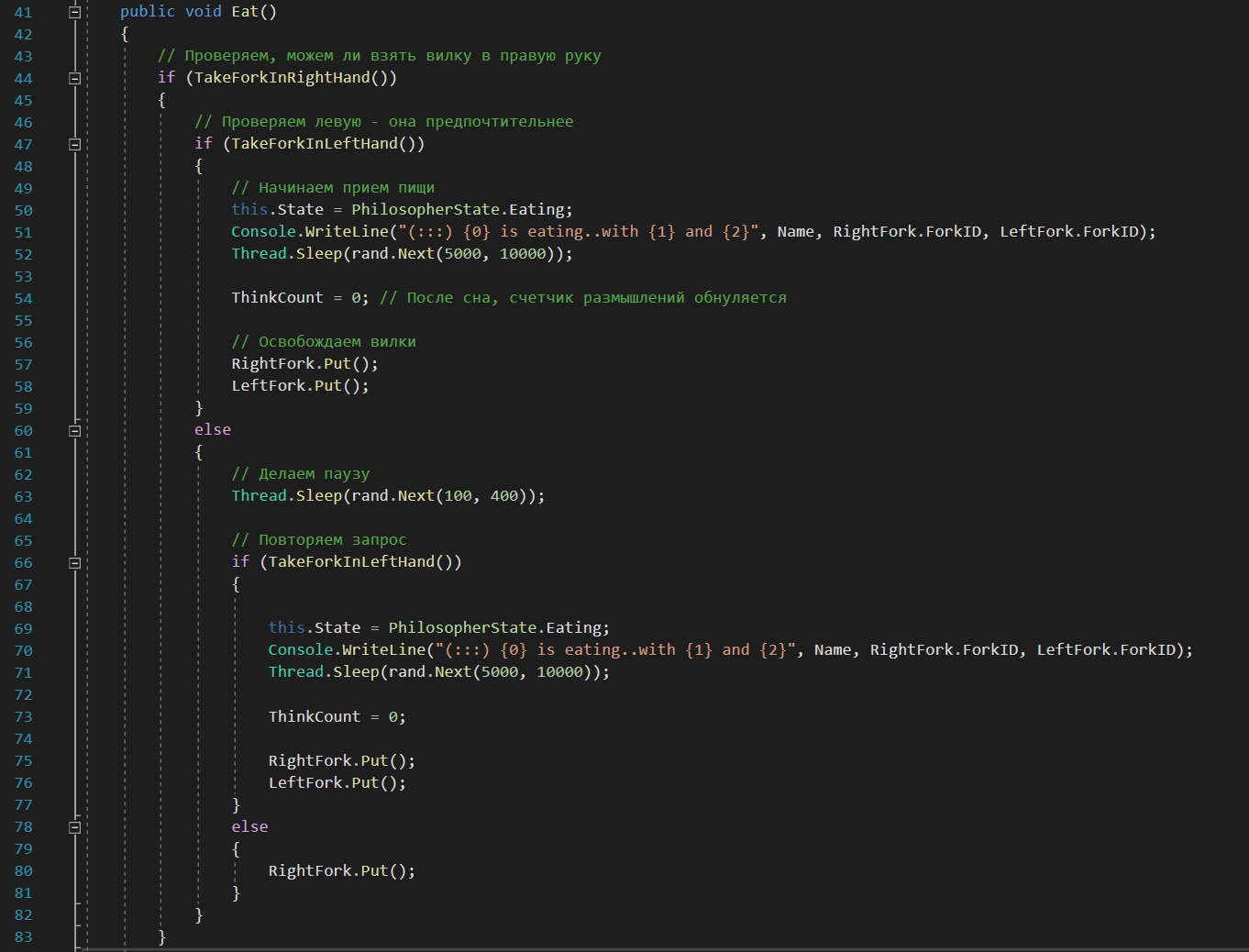
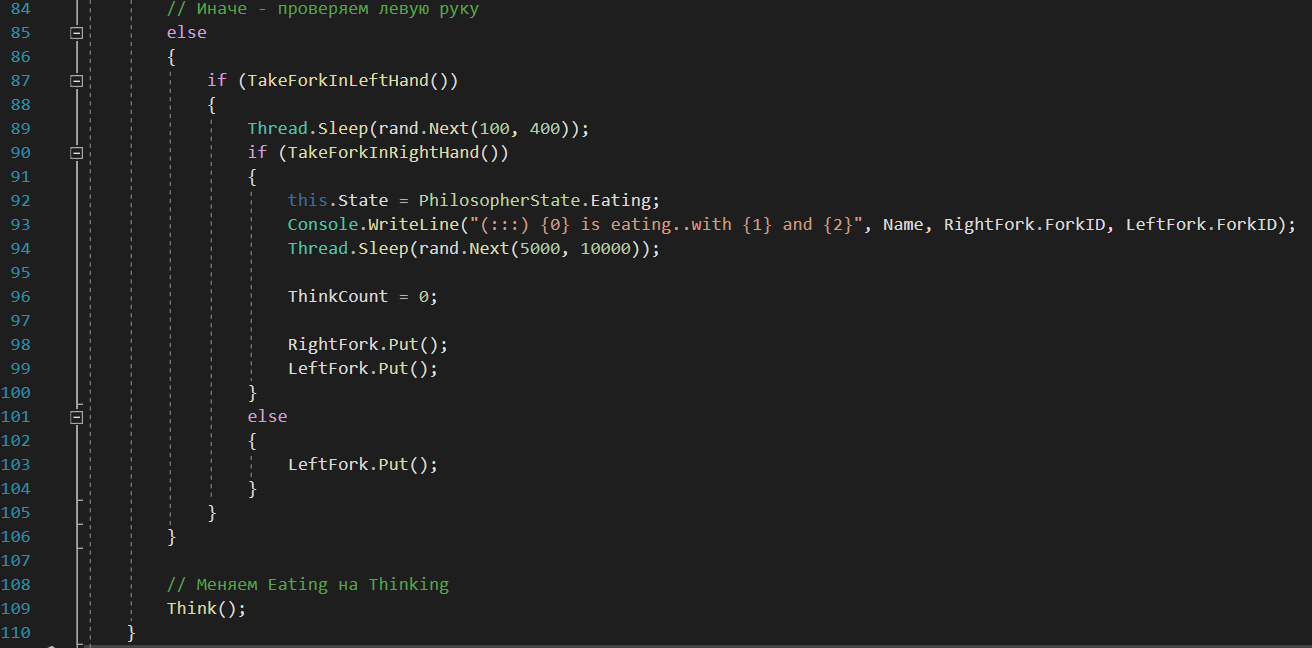
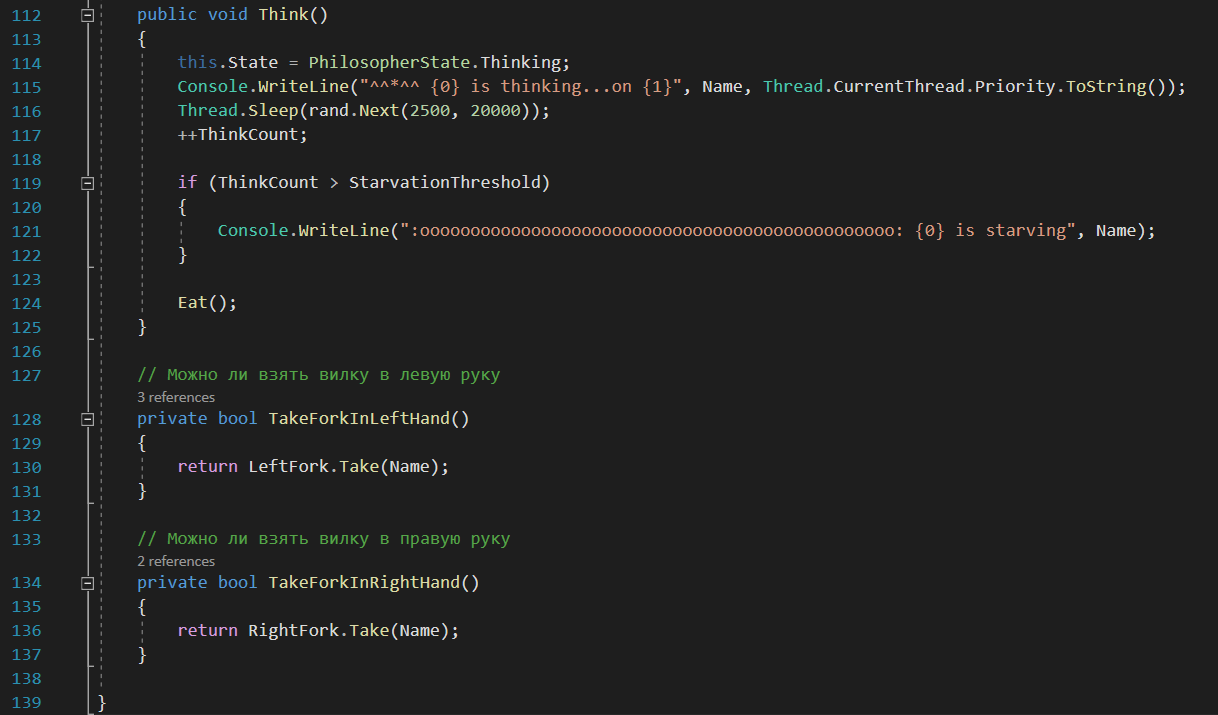
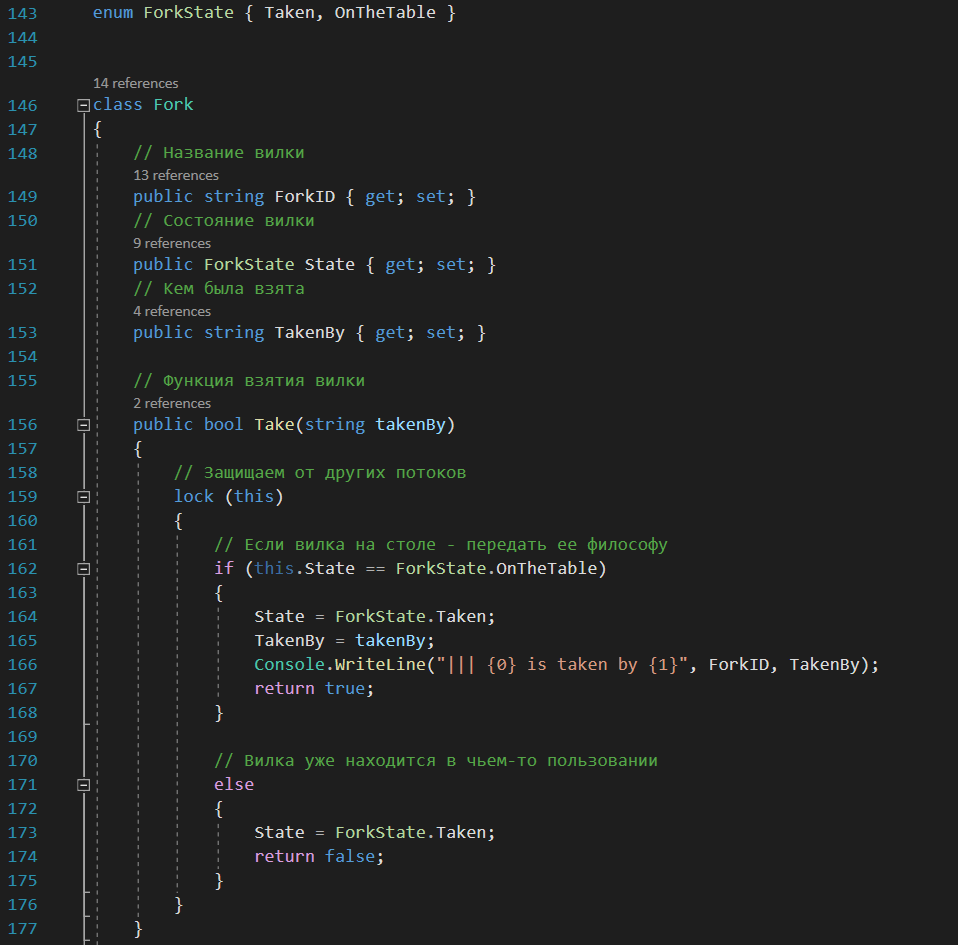
Отчет по лабораторной работе « Философы»

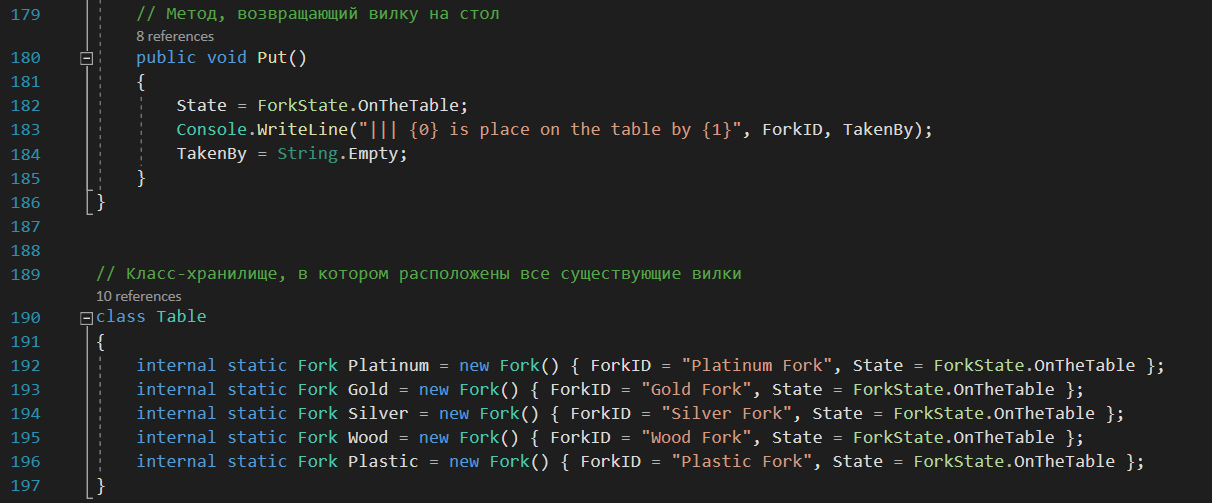




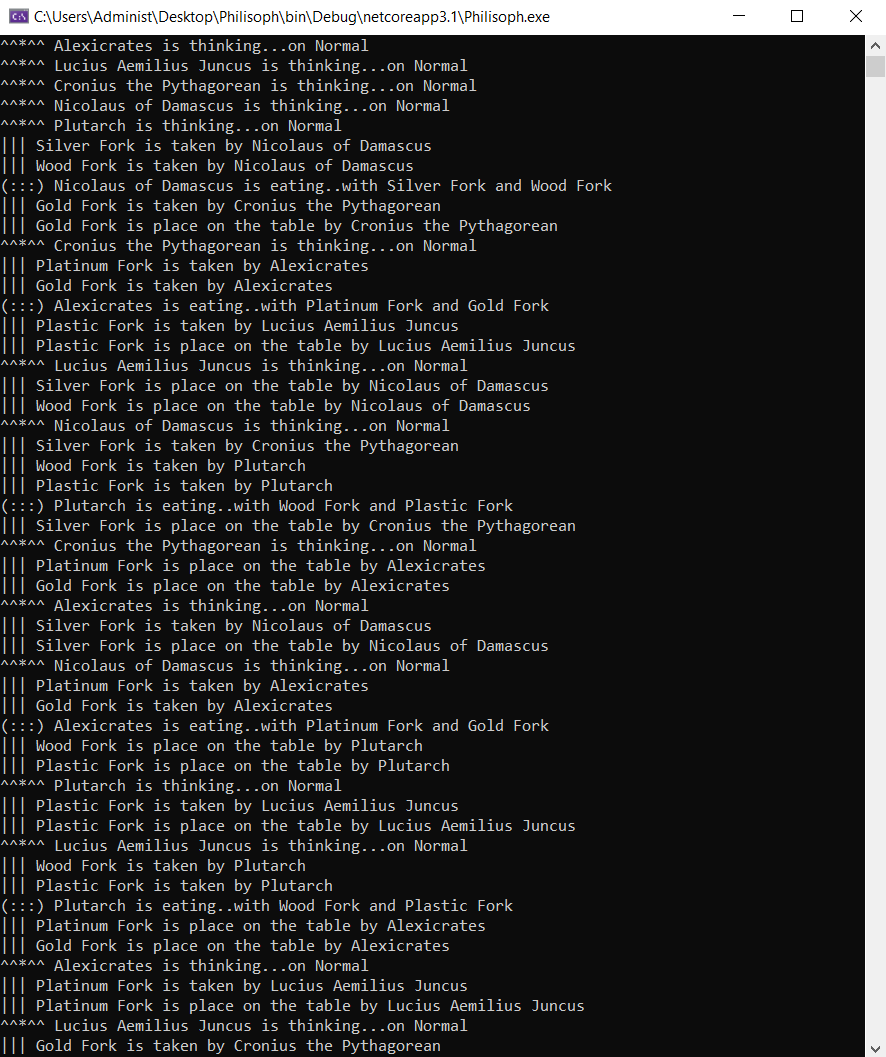












Код программы.

using System;

using System.Threading;

// Перечисления, определяющие состояния философов

enum PhilosopherState { Eating, Thinking }

// Основной класс-сущность философа

class Philosopher

{

// Имя философа

public string Name { get; set; }

// Состояние философа (Eating, Thinking)

public PhilosopherState State { get; set; }

// Количество размышлений до наступления голода

readonly int StarvationThreshold;

// Вилки, используемые философом

public readonly Fork RightFork;

public readonly Fork LeftFork;

// Случайная величина, отвечающая за промежуток приема пищи

Random rand = new Random();

// Количество "размышлений" до голодания

int ThinkCount = 0;

// Конструктор класса

public Philosopher(Fork rightFork, Fork leftFork, string name, int starvThreshold)

{

RightFork = rightFork;

LeftFork = leftFork;

Name = name;

State = PhilosopherState.Thinking; // По умолчанию, философ сыт и сразу приступает к мыслительной деятельности

StarvationThreshold = starvThreshold;

}

public void Eat()

{

// Проверяем, можем ли взять вилку в правую руку

if (TakeForkInRightHand())

{

// Проверяем левую - она предпочтительнее

if (TakeForkInLeftHand())

{

// Начинаем прием пищи

this.State = PhilosopherState.Eating;

Console.WriteLine("(:::) {0} is eating..with {1} and {2}", Name, RightFork.ForkID, LeftFork.ForkID);

Thread.Sleep(rand.Next(5000, 10000));

ThinkCount = 0; // После сна, счетчик размышлений обнуляется

// Освобождаем вилки

RightFork.Put();

LeftFork.Put();

}

else

{

// Делаем паузу

Thread.Sleep(rand.Next(100, 400));

// Повторяем запрос

if (TakeForkInLeftHand())

{

this.State = PhilosopherState.Eating;

Console.WriteLine("(:::) {0} is eating..with {1} and {2}", Name, RightFork.ForkID, LeftFork.ForkID);

Thread.Sleep(rand.Next(5000, 10000));

ThinkCount = 0;

RightFork.Put();

LeftFork.Put();

}

else

{

RightFork.Put();

}

}

}

// Иначе - проверяем левую руку

else

{

if (TakeForkInLeftHand())

{

Thread.Sleep(rand.Next(100, 400));

if (TakeForkInRightHand())

{

this.State = PhilosopherState.Eating;

Console.WriteLine("(:::) {0} is eating..with {1} and {2}", Name, RightFork.ForkID, LeftFork.ForkID);

Thread.Sleep(rand.Next(5000, 10000));

ThinkCount = 0;

RightFork.Put();

LeftFork.Put();

}

else

{

LeftFork.Put();

}

}

}

// Меняем Eating на Thinking

Think();

}

public void Think()

{

this.State = PhilosopherState.Thinking;

Console.WriteLine("^^\*^^ {0} is thinking...on {1}", Name, Thread.CurrentThread.Priority.ToString());

Thread.Sleep(rand.Next(2500, 20000));

++ThinkCount;

if (ThinkCount > StarvationThreshold)

{

Console.WriteLine(":ooooooooooooooooooooooooooooooooooooooooooooooo: {0} is starving", Name);

}

Eat();

}

// Можно ли взять вилку в левую руку

private bool TakeForkInLeftHand()

{

return LeftFork.Take(Name);

}

// Можно ли взять вилку в правую руку

private bool TakeForkInRightHand()

{

return RightFork.Take(Name);

}

}

// Перечисления, определяющее состояние сущности класса Fork

enum ForkState { Taken, OnTheTable }

class Fork

{

// Название вилки

public string ForkID { get; set; }

// Состояние вилки

public ForkState State { get; set; }

// Кем была взята

public string TakenBy { get; set; }

// Функция взятия вилки

public bool Take(string takenBy)

{

// Защищаем от других потоков

lock (this)

{

// Если вилка на столе - передать ее философу

if (this.State == ForkState.OnTheTable)

{

State = ForkState.Taken;

TakenBy = takenBy;

Console.WriteLine("||| {0} is taken by {1}", ForkID, TakenBy);

return true;

}

// Вилка уже находится в чьем-то пользовании

else

{

State = ForkState.Taken;

return false;

}

}

}

// Метод, возвращающий вилку на стол

public void Put()

{

State = ForkState.OnTheTable;

Console.WriteLine("||| {0} is place on the table by {1}", ForkID, TakenBy);

TakenBy = String.Empty;

}

}

// Класс-хранилище, в котором расположены все существующие вилки

class Table

{

internal statiс Fork Platinum = new Fork() { ForkID = "Platinum Fork", State = ForkState.OnTheTable };

internal static Fork Gold = new Fork() { ForkID = "Gold Fork", State = ForkState.OnTheTable };

internal static Fork Silver = new Fork() { ForkID = "Silver Fork", State = ForkState.OnTheTable };

internal static Fork Wood = new Fork() { ForkID = "Wood Fork", State = ForkState.OnTheTable };

internal static Fork Plastic = new Fork() { ForkID = "Plastic Fork", State = ForkState.OnTheTable };

}

namespace DinningPhilosophers

{

class Program

{

static void Main(string[] args)

{

// Создаем всех необходимых философов

Philosopher lucius = new Philosopher(Table.Plastic, Table.Platinum, "Lucius Aemilius Juncus", 4);

Philosopher alexicrates = new Philosopher(Table.Platinum, Table.Gold, "Alexicrates", 5);

Philosopher cronius = new Philosopher(Table.Gold, Table.Silver, "Cronius the Pythagorean", 6);

Philosopher nicolaus = new Philosopher(Table.Silver, Table.Wood, "Nicolaus of Damascus", 4);

Philosopher thomas = new Philosopher(Table.Wood, Table.Plastic, "Plutarch", 7);

// Начинаем выполнение в новых потоках

new Thread(lucius.Think).Start();

new Thread(alexicrates.Think).Start();

new Thread(cronius.Think).Start();

new Thread(nicolaus.Think).Start();

new Thread(thomas.Think).Start();

Console.ReadKey();

}

}

}