```
using System;
using System.Collections.Generic;
using System.Text;

namespace reliability
{
    class Sistema
    {
        //private Graph<Objekt, bool> m_graphOfObjekt;
        private Objekt.ObjektState m_CurrentSystemState, m_LastSystemState;

        private System.Collections.ArrayList m_arrObjekt; //Массив объектов
        private const uint m_NumOfObjects = 7; //Количество объектов

        private uint m_ChisloOtkazov; //Число отказов
        private uint m_Otrezkov; //Количество отрезко разбиения
        private double m_Vremya, m_Tm, m_ShagVrem, m_VremaPoslOtkaza, m_VremaPoslVosst; // ↙
    Праметры времени
        private Mathematika.Funktzya m_calcFuncOtk; //Функция для отказов
        private Mathematika.Funktzya m_calcFuncVosst; //Функция для восстановлений
        private Mathematika.Parametry.ExpZakona m_calcParam; //Параметры экспоненциального ↙
    закона

        private double[] m_Totkaz; //Время наработки на отказ
        private double[] m_Tvosst; //Время восстановления

        private double[] m_SrednTotkazNaOtrezke; //Среднее время наработки на отказ
        private double[] m_SrednTvosstNaOtrezke; //Среднее время восстановления
        private double[] m_KoefGot; //Коэффициент готовности

        //Инициализация системы
        public Sistema()
        {
            //Массив из m_NumOfObjects объектов
            m_arrObjekt = new System.Collections.ArrayList((int)m_NumOfObjects);

            //Массивы времён из максимум 20000000 элементов
            m_Totkaz = new double[20000000];
            m_Tvosst = new double[20000000];

            //Количество отрезков берём 10
            m_SrednTotkazNaOtrezke = new double[10];
            m_SrednTvosstNaOtrezke = new double[10];
            m_KoefGot = new double[10];

            m_VremaPoslOtkaza = 0;
            m_VremaPoslVosst = 0;
            m_CurrentSystemState = Objekt.ObjektState.Pa6otaet;
            m_LastSystemState = Objekt.ObjektState.Pa6otaet;
        }

        //Для доступа к функции отказа
        public Mathematika.Funktzya FuncOtk
        {
            get { return m_calcFuncOtk; }
            set { m_calcFuncOtk = value; }
        }

        //Для доступа к параметрам экспоненциального закона
        public Mathematika.Parametry.ExpZakona calcParam
        {
            get { return m_calcParam; }
            set { m_calcParam = value; }
        }

        //Для доступа к шагу времени
        public double ShagVrem
        {
            get { return m_ShagVrem; }
            set { m_ShagVrem = value; }
        }

        //Для доступа к временам ..
        public double[] GetArrraySrVrNaOtkaz
        {
            get { return m_SrednTotkazNaOtrezke; }
            set { }
        }
```

```
public double[] GetArrraySrVrVosst
{
    get { return m_SrednTvosstNaOtrezke; }
    set { }
}

//Для доступа к коэффициенту готовности
public double[] GetArrrayKoefGot
{
    get { return m_KoefGot; }
    set { }
}

//Для доступа к Tm
public double Tm
{
    get { return m_Tm; }
    set { m_Tm = value; }
}

//Для доступа к числу отказов системы
public uint ChisloOtkazovSystemy
{
    get { return m_ChisloOtkazov; }
    set { m_ChisloOtkazov = value; }
}

//Для доступа к числу отрезков
public uint Otrezkov
{
    get { return m_Otrezkov; }
    set { m_Otrezkov = value; }
}

internal Objekt Objekt
{
    get
    {
        throw new System.NotImplementedException();
    }
    set
    {
    }
}

internal Mathematika Mathematika
{
    get
    {
        throw new System.NotImplementedException();
    }
    set
    {
    }
}

//Изменение состояния системы
public void SystemStateChange(Objekt.ObjektState newSystemState)
{
    switch (newSystemState) //Конечный автомат элемента
    {
        case Objekt.ObjektState.Pa6otaet:
            //Переключение в состояние работы
            if (m_CurrentSystemState == Objekt.ObjektState.Pa6otaet)
                return;
            break;
        case Objekt.ObjektState.Slomalosj:
            //Переключение в состояние сбоя
            if (m_CurrentSystemState == Objekt.ObjektState.Slomalosj)
                return;
            break;
        default:
            throw new SystemException("Что происходит?");
    }
    //Установка последнего и текущего состояний системы
    m_LastSystemState = m_CurrentSystemState;
    m_CurrentSystemState = newSystemState;
```

```
    }

    //Процедура добавления объектов в массив
    public void DobavitObjekty()
    {
        for (uint i = 0; i < m_NumOfObjects + 1; i++)
        {
            //Создаём экземпляр класса объекта
            Objekt o = new Objekt();

            //Указываем параметры текущего объекта
            o.calcParam = m_calcParam;
            o.calcFunc = m_calcFuncOtk;
            o.ObjectID = i;
            o.Tm = m_Tm;
            o.ShagVrem = m_ShagVrem;

            //Запускаем симуляцию работы объекта
            o.Simulate();

            //Добавляем объект в массив объектов системы
            m_arrObjekt.Add(o);
        }
    }

    //Симуляция работы системы
    public void Simulate()
    {
        //Инициализация переменных
        double SredneeVremyaNarabNaOtkaz = 0;
        double SredneeVremyaVosst = 0;
        uint i = 0;
        long m = 0;
        m_Vremya = 0;

        //Цикл по всем отрезкам
        for (uint j = 1; j <= m_Otrezkov; j++)
        {
            //Инициализация переменных
            long k = 0;
            SredneeVremyaNarabNaOtkaz = 0;
            SredneeVremyaVosst = 0;

            //Цикл по заданному времени работы системы
            do
            {
                //Цикл определения работоспособности каждого из объектов системы
                foreach (Objekt o in m_arrObjekt)
                    o.RabotaetUluNet(m);

                //Определение глобальной работоспособности системы
                // по заданному закону связи объектов
                SystemStateChange(RabotaetUluNet());

                //В случае изменения состояния..
                if (m_CurrentSystemState != m_LastSystemState)
                {
                    if (m_LastSystemState == Objekt.ObjektState.Slomalosj)
                    {
                        //Записываем параметры изменения для отказа
                        m_Totkaz[i] = m_Vremya - m_VremaPoslVosst; //T.наработки.на.
отказ = T.текущее - T.послед.восст.
                        m_VremaPoslOtkaza = m_Vremya; //T.текущее
                        SredneeVremyaNarabNaOtkaz += m_Totkaz[i]; //Добавим T.наработки
.на.отказ к среднему времени наработки на отказ
                    }
                    else
                    {
                        //Записываем параметры изменения для восстановления
                        m_Tvosst[i] = m_Vremya - m_VremaPoslOtkaza; //T.восстановления
= T.текущее - T.послед.отказа.
                        m_VremaPoslVosst = m_Vremya; //T.текущее
                        SredneeVremyaVosst += m_Tvosst[i]; //Добавим T.восстановления к
 среднему времени восстановления
                    }
                    i++;
                }
```

```
                    //Переходим к следующему шагу по времени
                    m_Vremya += m_ShagVrem;
                    m++;
                }
                while (m_Vremya < j * m_Tm / m_Otrezkov); //Пока время в заданном интервале

                //Получим фактическое среднее время наработки на отказ
                SredneeVremyaNarabNaOtkaz /= (i - k);
                //Получим фактическое среднее время восстановления
                SredneeVremyaVosst /= (i - k);
                k = i;

                //Зафиксируем фактическое среднее время наработки на отказ
                m_SrednTotkazNaOtrezke[j - 1] = SredneeVremyaNarabNaOtkaz;
                //Зафиксируем фактическое среднее время восстановления
                m_SrednTvosstNaOtrezke[j - 1] = SredneeVremyaVosst;

                //Расчитаем коэффициент готовности
                m_KoefGot[j-1] = (SredneeVremyaNarabNaOtkaz + SredneeVremyaVosst) /
    SredneeVremyaNarabNaOtkaz;
            }
        }

        //Метод определяющий работоспособность элемента по выбранному нами закону
        public Objekt.ObjektState RabotaetUluNet()
        {
            if (// Исключительно для наглядности ))
                (((Objekt)m_arrObjekt[0]).CurrentObjektState.Equals(Objekt.ObjektState.
    Работает) &&
                ((Objekt)m_arrObjekt[1]).CurrentObjektState.Equals(Objekt.ObjektState.
    Работает)) ||
                (((Objekt)m_arrObjekt[2]).CurrentObjektState.Equals(Objekt.ObjektState.
    Работает) &&
                ((Objekt)m_arrObjekt[3]).CurrentObjektState.Equals(Objekt.ObjektState.
    Работает)) ||
                (((Objekt)m_arrObjekt[4]).CurrentObjektState.Equals(Objekt.ObjektState.
    Работает) &&
                ((Objekt)m_arrObjekt[5]).CurrentObjektState.Equals(Objekt.ObjektState.
    Работает)) &&
                ((Objekt)m_arrObjekt[6]).CurrentObjektState.Equals(Objekt.ObjektState.
    Работает) ||
                ((Objekt)m_arrObjekt[0]).CurrentObjektState.Equals(Objekt.ObjektState.
    Работает) ||
                ((Objekt)m_arrObjekt[2]).CurrentObjektState.Equals(Objekt.ObjektState.
    Работает) ||
                ((Objekt)m_arrObjekt[4]).CurrentObjektState.Equals(Objekt.ObjektState.
    Работает))
                return Objekt.ObjektState.Работает;
            else
                return Objekt.ObjektState.Slomalosj;
        }
    }
}
```