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

namespace reliability
{
    class Objekt : Object, IEquatable<Objekt>
    {
        //Перечисление состояний объекта
        public enum ObjektState
        {
            Slomalosj = 0,
            Pa6otaet = 1
        }

        private ObjektState m_CurrentObjektState; //Текущее состояние
        private Mathematika.Funktzya m_calcFuncOtk; //Функция определения времени наработки
     на отказ
        private Mathematika.Funktzya m_calcFuncVosst; //Функция определения времени
    восстановления
        private Mathematika.Parametry.ExpZakona m_calcParam; //Параметры функции
        private uint m_ObjectID; //Номер объекта
        private uint m_ChisloOtkazov; //Число отказов
        private double m_Tm; //Время работы
        private double m_ShagVrem; //Шаг времени
        private double[] m_Totkaz; //Времена наработки на отказ
        private double[] m_Tvosst; //Времена восстановления
        private bool[] m_Sostoyanie; //Булево состояние объекта

        //Инициализация объекта
        public Objekt()
        {
            m_Totkaz = new double[65536];
            m_Tvosst = new double[65536];
            m_Sostoyanie = new bool[20000000];
            //Изначально объект работает
            m_ChisloOtkazov = 0;
            m_CurrentObjektState = ObjektState.Pa6otaet;
        }

        //Для доступа к функциям распределений
        public Mathematika.Funktzya calcFunc
        {
            set
            {
                m_calcFuncOtk = value;
                m_calcFuncVosst = value;
            }
        }

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

        //Для доступа к текущему состоянию объекта
        public ObjektState CurrentObjektState
        {
            get { return m_CurrentObjektState; }
            set { }
        }

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

        //Для доступа к номеру объекта
        public uint ObjectID
        {
            get { return m_ObjectID; }
            set { m_ObjectID = value; }
        }
```

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

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

public reliability.Mathematika.Funktzya Funktzya
{
    get
    {
        throw new System.NotImplementedException();
    }
    set
    {
    }
}

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

//Функция сравнения объектов
public bool Equals(Objekt other)
{
    if(m_ObjectID == other.ObjectID)
        return true;
    return false;
}

//Изменение состояния объекта
public void ObjektStateChange(ObjektState newObjektState)
{
    switch (newObjektState) //Конечный автомат объекта
    {
        case ObjektState.Pa6otaet:
            //Переключение в состояние работы
            if (m_CurrentObjektState == ObjektState.Pa6otaet)
                return;
            break;
        case ObjektState.Slomalosj:
            //Переключение в состояние сбоя
            if (m_CurrentObjektState == ObjektState.Slomalosj)
                return;
            break;
        default:
            throw new SystemException("Что происходит?");
    }
    m_CurrentObjektState = newObjektState;

}

//Симуляция работы объектов
public void Simulate()
{
    //Инициализация переменных
    double t = 0;
    uint i = 0;

    //Цикл по заданному времени работы объекта
    do
    {
        //В зависимости от выбранной функции рассчитываем время наработки на отказ
        if (m_calcFuncOtk.Equals(Mathematika.Funktzya.Exponenta))
```

```
                    //m_Totkaz[i] =  Mathematika.Veibull(0.8f, 1000);
                    m_Totkaz[i] = Math.Round(Mathematika.Exponential(m_calcParam.lambda));

                //В зависимости от выбранной функции рассчитываем время восстановления
                if (m_calcFuncVosst.Equals(Mathematika.Funktzya.Exponenta))
                    //m_Tvosst[i] = Mathematika.ExponentialVosst(m_calcParam.mu);
                    m_Tvosst[i] = Math.Round(Mathematika.ExponentialVosst(m_calcParam.mu));

                //Определяем суммарное время событий
                t += m_Totkaz[i] + m_Tvosst[i];
                i++;
            }
            while (t <= m_Tm);

            double steptime = 0;
            double current = 0;
            double interval;
            uint j = 0;
            i = 0;
            m_Sostoyanie[0] = true;

            //Цикл по заданному времени работы объекта
            do
            {
                steptime += m_ShagVrem;
                interval = m_Totkaz[i] + m_Tvosst[i];

                //Заполняем массив состояний объекта на заданных интервалах времени
                //На интервале попали в отрезок работоспособности
                if (steptime <= current + m_Totkaz[i])
                {
                    m_Sostoyanie[j] = true;
                }
                //На интервале попали в отрезок восстановления
                else if (steptime <= current + interval)
                {
                    m_Sostoyanie[j] = false;
                }
                //Вышли за пределы интервала времени
                else if (steptime > current + interval)
                {
                    current += interval;
                    i++;
                }
                j++;
            }
            while (current <= m_Tm || i < 65535);/**/
        }



        public bool RabotaetUluNet(long index)//double time)
        {
            //По массиву состояний объекта определяем работоспособность
            // объекта на интервале времени, указанному по номеру index
            if (m_Sostoyanie[index])
                ObjektStateChange(ObjektState.Pa6otaet);
            else
                ObjektStateChange(ObjektState.Slomalosj);

            return m_Sostoyanie[index];
        }
    }
}
```