DENIS RAMSKIY II 11

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

namespace BagApp
{
    public class Gen
    {
        private List<Item> hromosom;
        public double fitness{set;get;}

        public Gen(Random rnd)
        {
            hromosom = new List<Item>();
            int CalcFitness = 0, CalcVol = 0;;
            for(int i=0;i<20;i++)
            {
                Item temp = new Item(rnd.Next(10) +1 , rnd.Next(10) +1,
rnd.Next()%60);
                CalcFitness += (temp.count * temp.price) / temp.volume;
                CalcVol += temp.count * temp.volume;
                hromosom.Add(temp);
            }

            if (CalcVol <= 1000)
                fitness = CalcFitness;
            else
                fitness = 0;
        }
        public Gen(Gen arg)
        {
            this.hromosom = new List<Item>();
            for (int i = 0; i < 20; i++)
                this.hromosom.Add(arg[i]);

            this.fitness = arg.fitness;
        }
        public Gen(List<Item> arg)
        {
            this.hromosom = new List<Item>();
            foreach (Item a in arg)
            {
                this.hromosom.Add(a);
                this.fitness += (a.count * a.price)/a.volume;
            }
        }
        static public Gen operator +(Gen arg1, Gen arg2)
        {
            arg1.hromosom = new List<Item>();
            for (int i = 0; i < 20; i++)
                arg1.hromosom.Add(arg2[i]);

            arg1.fitness = arg2.fitness;
            return arg1;
        }
        public Item this[int index]
        {
            get { return hromosom[index]; }
        }
        public void mutation(Random rnd)
        {
            for (int i = 0; i < 20; i++)
                if (rnd.Next() % 2000 < 1)
                    hromosom[i].count = rnd.Next() % 60;
```

```
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BagApp
{
    public class Generation
    {
        private List<Gen> generation;
        public double fitness{set;get;}

        public Generation(Random rnd)
        {
            generation = new List<Gen>();
            while (true)
            {
                Gen temp = new Gen(rnd);
                if (temp.fitness != 0)
                {
                    for (int i = 0; i<40;i++ )
                        generation.Add(temp);
                    break;
                }
            }
            this.Sort();
        }
        public void Sort()
        {
            for (int i = 0; i < 40; i++)
                for (int j = 0; j < 39; j++)
                    if (generation[j].fitness < generation[j + 1].fitness)
                    {
                        Gen temp = new Gen(generation[j]);
                        generation[j] += generation[j + 1];
                        generation[j + 1] += temp;
                    }
        }
        public List<Gen> get_parents(Random rnd)
        {
            List<Gen> TEMP = new List<Gen>();
            TEMP.Add(generation[rnd.Next() % 20]);
            TEMP.Add(generation[rnd.Next() % 20]);
            return TEMP;
        }
        public Gen get_child(Random rnd, List<Gen> par)
        {
            List<Item> TEMP = new List<Item>();
            for (int i = 0; i < 20; i++)
                if (rnd.Next() % 2 == 0)
                    TEMP.Add(par[0][i]);
                else
                    TEMP.Add(par[1][i]);
            Gen child = new Gen(TEMP);

            child.mutation(rnd);
            return child;
        }
        public void new_generation(Random rnd)
        {
            List<Gen> ngeneration = new List<Gen>();
            double nfitness = 0;
            for (int i = 0; i < 40; i++)
```

```
            {
                ngeneration.Add(this.get_child(rnd, this.get_parents(rnd)));
                nfitness += ngeneration[i].fitness / 40;
            }

            for (int i = 0; i < 40; i++)
                this.generation[i] += ngeneration[i];
            this.fitness = nfitness;
            this.Sort();
        }
        public Gen this[int index]
        {
            get { return generation[index]; }
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BagApp
{
    public class Item
    {
        public int price { set; get; }
        public int volume { set; get; }
        public int count { set; get; }

        public Item(int pr, int vol, int n)
        {
            this.price = pr;
            this.volume = vol;
            this.count = n;
        }
        public Item(Item arg)
        {
            this.price = arg.price;
            this.volume = arg.volume;
            this.count = arg.count;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BagApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Random rnd = new Random(DateTime.Now.Millisecond);
            Generation work = new Generation(rnd);
            while (true)
            {
                Console.WriteLine(work.fitness);
                for (int i = 0; i < 20; i++)
                {
                    Console.WriteLine("   Price: {0,6} Vol: {1,6} Count: {2,6}",
                    work[0][i].price,work[0][i].volume,work[0][i].count);
                }
                Console.ReadKey();
```

```
                work.new_generation(rnd);
            }
        }
    }
}
```