```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace DronokHarca
{
   abstract class Drone
   {
      ushort id;

      public ushort Id
      {
         get { return id; }
         private set
         {
            if (value <= 0)
               throw new BadIdException(value);
            id = value;
         }
      }

      private string name;

      public string Name
      {
         get { return name; }
         set
         {
            if (value.Length < 3)
               throw new BadNameException();
            name = value;
         }
      }

      PointF coordinates;

      public PointF Coordinates
      {
         get { return coordinates; }
      }

      internal protected bool isAlive;

      public bool IsAlive
      {
         get { return isAlive; }
      }

      public Drone(ushort id, string name, PointF coordinates)
      {
          Id = id;
         Name = name;
         this.coordinates = coordinates;
         isAlive = true;
       }

       public void Fly(float x, float y)
       {
          //coordinates = new PointF(x, y);
          coordinates.X = x;
           coordinates.Y = y;
        }
    }
}
```

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

namespace DronokHarca
{
    enum Spectrum { UltraViolet, Visible, InfraRed, Thermal }

    interface Iscout
    {
        void TakePhoto(Spectrum spectrum);
    }
}
```

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

namespace DronokHarca
{
    interface Iwarrior
    {
        void Bomb();
        void Shoot(Drone enemy);
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace DronokHarca
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DronokHarca
{
    class ScoutDrone: Drone, Iscout
    {
        private bool thermalCapable;

        public bool ThermalCapable
        {
            get { return thermalCapable; }
            set { thermalCapable = value; }
        }

        private byte[] numOfPhotos;

        public void TakePhoto(Spectrum spectrum)
        {
            if (spectrum == Spectrum.Thermal && !thermalCapable)
                throw new NotThermalCapableException();
            numOfPhotos[(int)spectrum]++;
        }

        public ScoutDrone(ushort id, string name, PointF coordinates, bool thermalCapable)
            : base(id, name, coordinates)
        {
            this.thermalCapable = thermalCapable;
            numOfPhotos = new byte[Enum.GetValues(typeof(Spectrum)).Length];
        }
    }
}
```

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

namespace DronokHarca
{
    class WarriorDrone : Drone, IWarrior, Iscout
    {
        byte numOfThermalPhotos;

        public void TakePhoto(Spectrum spectrum)
        {
            if (spectrum != Spectrum.Thermal)
                throw new BadSpectrumException();
            numOfThermalPhotos++;
        }

        byte numOfBombs;

         public void Bomb()
         {
             if (numOfBombs == 0)
                 throw new NoBombsException();
             numOfBombs--;
         }
```

```
        private float range;

        public float Range
        {
            get { return range; }
            set
            {
                if (value < 0)
                    throw new BadRangeException();
                range = value;
            }
        }

        public void Shoot(Drone enemy)
        {
            if (Math.Sqrt(
                Math.Pow(this.Coordinates.X - enemy.Coordinates.X, 2)
                +
                Math.Pow(this.Coordinates.Y - enemy.Coordinates.Y, 2)
                )
                > range)
                throw new OutOfRangeException();
            enemy.isAlive = false;
        }
    }
}
```

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

namespace Edesszajuak
{
   enum Csokifajta {kerek, szogletes, hosszu, rovid, gombolyu, lapos }
   class Csokolade:Edesseg
   {
      int kakaotartalom;
      public int Kakaotartalom
      {
         get
         {
            return kakaotartalom;
         }
         set
         {
             kakaotartalom = value;
         }
      }
      Csokifajta fajta;
      public Csokifajta Fajta
      {
         get
         {
            return fajta;
         }

      }
      public Csokolade(string nev, float cukortartalom, int tomeg, int kakaotartalom, Csokifajta fajta) : base(nev,
cukortartalom, tomeg)
      {
         this.Kakaotartalom = kakaotartalom;
         this.fajta = fajta;
      }
          public override string ToString()
        {
             return string.Format("Nev: {0}\n\tCukortaralom: {1}\n\t Tömeg: {2}\n\tKakaótartalom: {3}\n\tFajta: {4}",
Nev,Cukortartalom,Tomeg,kakaotartalom,fajta);
           }

      }
   }
-----------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Edesszajuak
{
   enum Drazseszin {piros,kek,zold,sarga}
   class Drazse:Edesseg
   {
      Drazseszin szin;
      public Drazseszin Szin
      {
         get
         {
            return szin;
         }
      }
      public Drazse(string nev, float cukortartalom, int tomeg, Drazseszin szin) : base(nev, cukortartalom, tomeg)
      {
         this.szin = szin;
      }
   }
```

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

namespace Edesszajuak
{
    class Edeseegbolt
    {
        List<Edesseg> Edessegek = new List<Edesseg>();
        public void Addedesseg(Edesseg ujedesseg)
        {
            Edessegek.Add(ujedesseg);
        }
        public bool VaneEdesseg
        {
            get
            {
                bool vane = true;
                foreach (Edesseg x in Edessegek)
                {
                    if (x.Diabetikus == true)
                        vane = true;
                    else
                        vane = false;
                }
                return vane;
            }
        }
        public int OsszesDrazse(Drazseszin szin)
        {
            int osszes = 0;
            Drazse d;
            foreach (Edesseg x in Edessegek)
            {
                if (x is Drazse)
                {
                    d = x as Drazse;

                    if (d.Szin == szin)
                        osszes += d.Tomeg;
                }
            }
            return osszes;
        }
        public List<Csokolade> Csoki(Csokifajta fajta, int kakaotart)
        {
            List<Csokolade> csoki = new List<Csokolade>();
            Csokolade cs;
            foreach (Edesseg x in Edessegek)
            {
                if (x is Csokolade)
                {
                    cs = x as Csokolade;
                    if(cs.Fajta==fajta&& cs.Kakaotartalom==kakaotart)
                    csoki.Add(cs);
                }

            }
            return csoki;
        }
    }
}
```

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

namespace Edesszajuak
{
    class Edesseg
    {
        string nev;
        public string Nev
        {
            get
            {
                return nev;
            }
            set
            {
                nev = value;
            }
        }
        float cukortartalom;
        public float Cukortartalom
        {
            get
            {
                return cukortartalom;
            }
            set
            {
                cukortartalom = value;
            }
        }
        public bool Diabetikus
        {
            get
            {
                if (cukortartalom < 0.5)
                    return true;
                else
                    throw new Exception("Nem megfelelő a cukortartalom");
            }
        }
        int tomeg;
        public int Tomeg
        {
            get
            {
                return tomeg;
            }
            set
            {
                tomeg = value;
            }
        }
        public Edesseg(string nev, float cukortartalom, int tomeg)
        {
            this.Nev = nev;
            this.Cukortartalom = cukortartalom;
            this.Tomeg = tomeg;
        }
    }
}
```

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

namespace Edesszajuak
{
    class Edeseegbolt
    {
        List<Edesseg> Edessegek = new List<Edesseg>();
        public void Addedesseg(Edesseg ujedesseg)
        {
            Edessegek.Add(ujedesseg);
        }
        public bool VaneEdesseg
        {
            get
            {
                bool vane = true;
                foreach (Edesseg x in Edessegek)
                {
                    if (x.Diabetikus == true)
                        vane = true;
                    else
                        vane = false;
                }
                return vane;
            }
        }
        public int OsszesDrazse(Drazseszin szin)
        {
            int osszes = 0;
            Drazse d;
            foreach (Edesseg x in Edessegek)
            {
                if (x is Drazse)
                {
                    d = x as Drazse;

                    if (d.Szin == szin)
                        osszes += d.Tomeg;
                }
            }
            return osszes;
        }
        public List<Csokolade> Csoki(Csokifajta fajta, int kakaotart)
        {
            List<Csokolade> csoki = new List<Csokolade>();
            Csokolade cs;
            foreach (Edesseg x in Edessegek)
            {
                if (x is Csokolade)
                {
                    cs = x as Csokolade;
                    if(cs.Fajta==fajta&& cs.Kakaotartalom==kakaotart)
                    csoki.Add(cs);
                }

            }
            return csoki;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace Edesszajuak
{
    class Program
    {
        static void Main(string[] args)
        {
            Edeseegbolt bolt = new Edeseegbolt();
            StreamReader reader = new StreamReader("edessegbolt.txt");

            while (!reader.EndOfStream)
            {
                string sor = reader.ReadLine();
                string[] szo = sor.Split(';');
                if (szo.Length == 3)
                    bolt.Addedesseg(new Edesseg(szo[0], float.Parse(szo[1]), int.Parse(szo[2])));
                else if (szo.Length == 4)
                    bolt.Addedesseg(new Drazse(szo[0], float.Parse(szo[1]), int.Parse(szo[2]),
(Drazseszin)Enum.Parse(typeof(Drazseszin), szo[3]));
                else if (szo.Length == 5)
                    bolt.Addedesseg(new Csokolade(szo[0], float.Parse(szo[1]), int.Parse(szo[2]),
int.Parse(szo[3]), (Csokifajta)Enum.Parse(typeof(Csokifajta), szo[4]));
                else
                    throw new Exception("Nem megfeleló file");
            }
            foreach (Csokolade x in bolt.Csoki(Csokifajta.lapos, 80))
            {

                Console.WriteLine(x);
            }
    //        Console.WriteLine("{} kg sárga drazsé van a boltban", bolt.OsszesDrazse(Drazseszin.sarga) /
1000);



            Console.ReadKey();
        }
    }
}
```

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

namespace Mehek
{
   class Darazs : Rovar, Ibeporzo
   {
      public int Nektargyujtes()
      {
         return 0;
      }
      public void Beporoz(Virag virag)
      {
         virag.Beporoztak();

      }
      public Darazs(string rovarnev) : base(rovarnev)
      { }
   }
```

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

namespace Mehek
{
   interface Ibeporzo
   {
      void Beporoz(Virag virag);
      int Nektargyujtes();
   }
}
```

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

namespace Mehek
{
   class Meh:Rovar,IBeporzo
   {
      int begyujtottnektar;
      int osszes=0;
      public int Nektargyujtes()
      {
         return osszes;
      }
      public void Beporoz(Virag virag)
      {
         virag.Beporoztak();
         virag.NektartAd();
         begyujtottnektar = virag.Nektarszam - virag.NektartAd();
         osszes += begyujtottnektar;
      }
      public Meh(string rovarnev) : base(rovarnev)
      { }
   }
}
```

```
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace Mehek
{
    class Program
    {
        static void Main(string[] args)
        {

            List<Virag> Viragok = new List<Virag>();
            List<Rovar> Rovarok = new List<Rovar>();
            try
            {
                StreamReader reader = new StreamReader("data.txt");
                while (!reader.EndOfStream)
                {
                    string[] szavak = reader.ReadLine().Split(';');
                    switch (szavak[0])
                    {
                        case "virág":
                            Viragok.Add(new Virag(szavak[1], int.Parse(szavak[2])));
                            break;
                        case "darázs":
                            Rovarok.Add(new Darazs(szavak[1]));
                            break;
                        case "méh":
                            Rovarok.Add(new Meh(szavak[1]));
                            break;

                    }
                }
                reader.Close();
            }

            catch (RosszNektarszamException)
            {
                Console.WriteLine("Rossz (negatív) nektarszamot adtál meg.");
            }
            catch (RosszRovarnevException)
            {
                Console.WriteLine("Rossz rovarnevet adtál meg. Minimum 3 karakter kell.");
            }
            catch (RosszViragnevException)
            {
                Console.WriteLine("Rossz virágnevet adtál meg. Minimum 3 karakter kell.");
            }

            Random rnd = new Random();

            int osszes = 0;
            foreach (Rovar x in Rovarok)
              {
                if (x is Darazs)
                {
                    (x as Darazs).Beporoz(Viragok[rnd.Next(Viragok.Count)]);
                    osszes += (x as Darazs).Nektargyujtes();
                }
                if (x is Meh)
                {
                    (x as Meh).Beporoz(Viragok[rnd.Next(Viragok.Count)]);
                      osszes += (x as Meh).Nektargyujtes();
                }
```

```
            }
            byte a=0;
            foreach (Virag x in Viragok)
            {
                if (x.Beporzasszam > a)
                    a = x.Beporzasszam;
            }
            foreach (Virag x in Viragok)
            {
                if (x.Beporzasszam == a)
                    Console.WriteLine(x);

            }

                Console.WriteLine("{0} milliliter nektárt gyűjtöttek a rovarok.", osszes);
                Console.ReadKey();
            }
        }
    }
```

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

namespace Mehek
{
    class RosszNektarszamException:Exception
    {
    }
}
```

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

namespace Mehek
{
    class RosszRovarnevException:Exception
    {
    }
}
```

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

namespace Mehek
{
    class RosszViragnevException: Exception
    {
    }
}
```

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

namespace Mehek
{
    abstract class Rovar
    {
        string rovarnev;

        public string Rovarnev
        {
            get
            {
                return rovarnev;
            }

            set
            {
                if (value.Length >= 0)
                    rovarnev = value;
                else
                    throw new RosszRovarnevException();
            }
        }
        public Rovar(string rovarnev)
        {
            this.Rovarnev = rovarnev;
        }
    }
}
```

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

namespace Mehek
{
    class Virag
    {
        string viragnev;

        public string Viragnev
        {
            get
            {
                return viragnev;
            }

            set
            {
                if (value.Length >= 3)
                    viragnev = value;
                else
                    throw new RosszViragnevException();

            }
        }
```

```
  private byte beporzasszam=0;
      public byte Beporzasszam
      {
         get
         {
            return beporzasszam;
         }

      }
      int nektarszam;
      public int Nektarszam
      {
         get
         {
            return nektarszam;
         }

         set
         {
            if (value >= 0)
               nektarszam = value;
            else
               throw new RosszNektarszamException();
         }
      }

      public byte Beporoztak()
      {
         return beporzasszam++;
      }
      public int NektartAd()
      {
         Random rnd = new Random();
         int nektarlevon = rnd.Next(nektarszam + 1);

         return nektarszam - nektarlevon;
      }
      public Virag(string viragnev, int nektarszam)
      {
         this.Viragnev = viragnev;
         this.Nektarszam = nektarszam;
      }
      public override string ToString()
      {
         return string.Format("{0} virágot porozták be ennyiszer {1}", viragnev,beporzasszam);
      }
   }
}
```

```
virág;margaréta;22
méh;poszméh
darázs;kecskedarázs
méh;háziméh
virág;kankalin;17
darázs;lódarázs
virág;szegfű;19
virág;ibolya;26
```

DRONOK TXT:
1;alfa1;38.45;137.12;0
2;alfa2;170.12;45.16;1
3;alfa3;45.15;278.1;1
4;beta1;90.12;167.45;23;200.12
5;beta2;120.34;150.57;12;150.15
6;beta3;119.01;148.12;35;230.17

EDES TXT:
cukor;60;5000
milka tej;50;120;30;lapos
milka et;40;150;70;kerek
dunakavics;87;500;piros
franciadrazse;68;1000;sarga
tibi;50;350;80;lapos
