

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**204 DATA STRUCTURES (3+1)**

**2020–2021 FALL SEMESTER**

**PROJECT-1 REPORT**

**(Arrays, Matrices, Methods, Random Numbers)**

**DELIVERY DATE**

14/12/2020

**PREPARED BY**

05180000028 Kaan Caner Kurtcephe

05190000818 Metehan Akmeşe

05190000835 Oğuzhan Demir

İçindekiler

[1) POINTS IN A 2D PLANE 2](#_Toc57390051)

[1.a Rastgele Nokta Üretimi 2](#_Toc57390052)

[1.a.1 Kodlar 2](#_Toc57390053)

[1.a.2 Ekran görüntüleri 2](#_Toc57390054)

[1.a.3 Açıklama 2](#_Toc57390055)

[1.b Uzaklık Matrisi 2](#_Toc57390056)

[1.b.1 Kodlar 2](#_Toc57390057)

[1.b.2 Ekran görüntüleri 3](#_Toc57390058)

[1.b.3 Açıklama 3](#_Toc57390059)

[2) CLASSIFICATION USING K-NEAREST NEIGHBORS (KNN) ALGORITHM 3](#_Toc57390060)

[2.a KNN ile sınıflandırma 3](#_Toc57390061)

[2.a.1 Algoritma sözde-kodu 3](#_Toc57390062)

[2.b Banknot sınıflandırma 6](#_Toc57390063)

[2.b.1 Kodlar 6](#_Toc57390064)

[2.b.2 Ekran görüntüleri 7](#_Toc57390065)

[2.b.3 Açıklama 7](#_Toc57390066)

[2.c Başarı ölçümü 7](#_Toc57390067)

[2.c.1 Kodlar 7](#_Toc57390068)

[2.c.2 Ekran görüntüleri 9](#_Toc57390069)

[2.c.3 Açıklama 11](#_Toc57390070)

[2.d Listeleme 11](#_Toc57390071)

[2.d.1 Kodlar 11](#_Toc57390072)

[2.d.2 Ekran görüntüleri 11](#_Toc57390073)

[2.d.3 Açıklama 12](#_Toc57390074)

[Özdeğerlendirme Tablosu 12](#_Toc57390075)

# 1) POINTS IN A 2D PLANE

Visual Studio, 16.7.7, C#

## 1.a Rastgele Nokta Üretimi

### 1.a.1 Kodlar

public static double GetRandomNumber(int minimum, int maximum) //Rastgele double sayı döndüren metod

{

return random.NextDouble() +random.Next(minimum,maximum);

### }

for (int i = 0; i < n; i++) //Rastgele noktaların oluşturulduğu ve matrisin içine atanan döngü

{

double x = GetRandomNumber(0, width);

double y = GetRandomNumber(0, heigth);

nokta nokta = new nokta(x, y);

noktalar[i] = nokta;

noktalarmatrisi[i, 0] = nokta.x;

noktalarmatrisi[i, 1] = nokta.y;

}

### 1.a.2 Ekran görüntüleri

### 

### 1.a.3 Açıklama

Kullanılan veri yapıları:Random numbers,Array,Class

0-1 arasında random double ve 0-100 arasında random integer oluşturup topladık.Bu sayede 0-100 arasında double elde ettik.

## 1.b Uzaklık Matrisi

### 1.b.1 Kodlar

public double UzaklikHesapla(nokta n1, nokta n2) //Verilen iki noktanın uzaklığını hesaplayıp döndürür

{

return Math.Sqrt(Math.Pow((n2.x - n1.x), 2) + Math.Pow((n2.y - n1.y), 2));

}

double[,] uzaklikmatris = new double[n, n];

for (int i = 0; i < n; i++) //Uzaklık matrisinin oluşturulduğu döngü

{

for (int j = 0; j < n; j++)

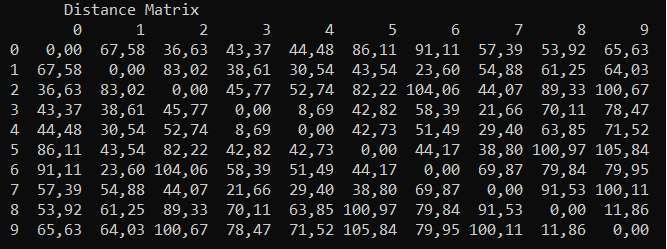
{

uzaklikmatris[i, j] = noktalar[i].UzaklikHesapla(noktalar[i], noktalar[j]);

}

}

### 1.b.2 Ekran görüntüleri



### 1.b.3 Açıklama

Kullanılan veri yapıları: Array, Matris,Metot,Class

Random olarak üretilen noktaların birbirine olan uzaklıklarını hesapladık. Bu uzaklıkları da Matris olarak yazdırdık.

# 2) CLASSIFICATION USING K-NEAREST NEIGHBORS (KNN) ALGORITHM

Visual Studio, 16.7.7, C#

## 2.a KNN ile sınıflandırma

### 2.a.1 Algoritma

Türü bilinmeyen banknot bilgilerini al

Verisetindeki diğer banknotlarla uzaklıklarını hesapla ve dizi içinde tut

Bu diziyi mesafeye göre sırala

Bu dizideki ilk k tane banknotun türlerine bak ve kaç tane sahte kaç tane gerçek para olduğuna bak

Bu sayıya göre en çok hangisinden varsa parayı sahte veya gerçek olarak tahminle

int ToplamElemanSayisi = ToplamSayiBul(textFile);

Banknot[] OrnekBanknotlar = new Banknot[ToplamElemanSayisi];

DosyaOku(OrnekBanknotlar);

Console.WriteLine("Gerçekliğini Kontrol Etmek İstediğiniz Paranın; ");

Console.WriteLine("Varyans Değerini Giriniz : ");

double varyansDegeri = DoubleCevir(Console.ReadLine());

Console.WriteLine("Çarpıklık Değerini Giriniz : ");

double carpiklikDegeri = DoubleCevir(Console.ReadLine());

Console.WriteLine("Basıklık Değerini Giriniz : ");

double basiklikDegeri = DoubleCevir(Console.ReadLine());

Console.WriteLine("Entropi Değerini Giriniz : ");

double entropiDegeri = DoubleCevir(Console.ReadLine());

Banknot bilinmeyenBanknot = new Banknot(varyansDegeri, carpiklikDegeri, basiklikDegeri, entropiDegeri);

BelirliBirDegereGoreMesafeBulma(bilinmeyenBanknot, OrnekBanknotlar);

Sıralama(OrnekBanknotlar);

Console.WriteLine("Kıyaslamak istediğiniz miktarı giriniz : ");

int k = Int32.Parse(Console.ReadLine());

Yazdir(OrnekBanknotlar, k);

int gercekmiSahtemi2=0;

GercekMiSahteMi(OrnekBanknotlar, k,gercekmiSahtemi2);

public static double MesafeHesapla(Banknot b1, Banknot b2) //İki banknot arasındaki benzerliği hesaplayan formül

{

return Math.Sqrt(Math.Pow((b2.VaryansDegeri - b1.VaryansDegeri), 2) + Math.Pow((b2.CarpiklikDegeri - b1.CarpiklikDegeri), 2) + Math.Pow((b2.BasiklikDegeri - b1.BasiklikDegeri), 2) + Math.Pow((b2.EntropiDegeri - b1.EntropiDegeri), 2));

}

public static int ToplamSayiBul(string s) //Örnek olarak verilen banknot textinin satır sayısı için (banknot adeti)

{

int deger = 0;

if (File.Exists(s))

{

string[] lines = File.ReadAllLines(s);

foreach (string line in lines)

{

deger++;

}

}

return deger;

}

public static Banknot[] DosyaOku(Banknot[] banknotDizisi) //Text içindeki banknotları, Banknot nesnesi halinde tutan arrayi

{ //oluşturan metot

int sayac = 0;

if (File.Exists(textFile))

{

string[] lines = File.ReadAllLines(textFile);

foreach (string line in lines)

{

string[] nitelikler = line.Split(',');

Banknot banknot = new Banknot(DoubleCevir(nitelikler[0]), DoubleCevir(nitelikler[1]), DoubleCevir(nitelikler[2]), DoubleCevir(nitelikler[3]), Convert.ToInt32(nitelikler[4]));

banknotDizisi[sayac] = banknot;

sayac++;

}

}

return banknotDizisi;

}

public static Banknot[] BelirliBirDegereGoreMesafeBulma(Banknot b1, Banknot[] banknotDizisi)

{

for (int i = 0; i < banknotDizisi.Length; i++) //Özellikleri girilen banknotun, text içindeki banknotlarla olan benzerliğini

{ //kaydeden metot

banknotDizisi[i].SetMesafe(Banknot.MesafeHesapla(b1, banknotDizisi[i]));

}

return banknotDizisi;

}

public static Banknot[] Sıralama(Banknot[] banknotDizisi) //Arrayi benzerliğe göre sıraladığımız metot

{

int x = 0;

while (x < banknotDizisi.Length)

{

int y = 0;

while (y < banknotDizisi.Length)

{

if (banknotDizisi[x].Mesafe < banknotDizisi[y].Mesafe)

{

Banknot temp = new Banknot(banknotDizisi[x].VaryansDegeri, banknotDizisi[x].CarpiklikDegeri, banknotDizisi[x].BasiklikDegeri, banknotDizisi[x].EntropiDegeri, banknotDizisi[x].Tur, banknotDizisi[x].Mesafe);

banknotDizisi[x] = banknotDizisi[y];

banknotDizisi[y] = temp;

}

y++;

}

x++;

}

return banknotDizisi;

}

public static int GercekMiSahteMi(Banknot[] b, int k,int gercekmiSahtemi) //Bir banknotun, banknot arrayine kıyasla gerçek veya

{ //sahte olduğunu tahminlediğimiz metot

int gercekSayisi = 0;

int sahteSayisi = 0;

for (int i = 0; i < k; i++)

{

if (b[i].Tur == 1)

{

gercekSayisi++;

}

else

{

sahteSayisi++;

}

}

if (gercekSayisi > sahteSayisi)

{

Console.WriteLine("Girdiğiniz para gerçek");

gercekmiSahtemi = 1;

}

else if (sahteSayisi > gercekSayisi)

{

Console.WriteLine("Girdiğiniz para sahte");

gercekmiSahtemi = 0;

}

else

{

if (b[0].Tur == 1)

{

Console.WriteLine("Girdiğiniz para gerçek");

gercekmiSahtemi = 1;

}

else

{

Console.WriteLine("Girdiğiniz para sahte");

gercekmiSahtemi = 0;

}

}

return gercekmiSahtemi;

}

## 2.b Banknot sınıflandırma

### 2.b.1 Kodlar

public static int GercekMiSahteMi(Banknot[] b, int k,int gercekmiSahtemi) //Bir banknotun, banknot arrayine kıyasla gerçek veya

{ //sahte olduğunu tahminlediğimiz metot

int gercekSayisi = 0;

int sahteSayisi = 0;

for (int i = 0; i < k; i++)

{

if (b[i].Tur == 1)

{

gercekSayisi++;

}

else

{

sahteSayisi++;

}

}

if (gercekSayisi > sahteSayisi)

{

Console.WriteLine("Girdiğiniz para gerçek");

gercekmiSahtemi = 1;

}

else if (sahteSayisi > gercekSayisi)

{

Console.WriteLine("Girdiğiniz para sahte");

gercekmiSahtemi = 0;

}

else

{

if (b[0].Tur == 1)

{

Console.WriteLine("Girdiğiniz para gerçek");

gercekmiSahtemi = 1;

}

else

{

Console.WriteLine("Girdiğiniz para sahte");

gercekmiSahtemi = 0;

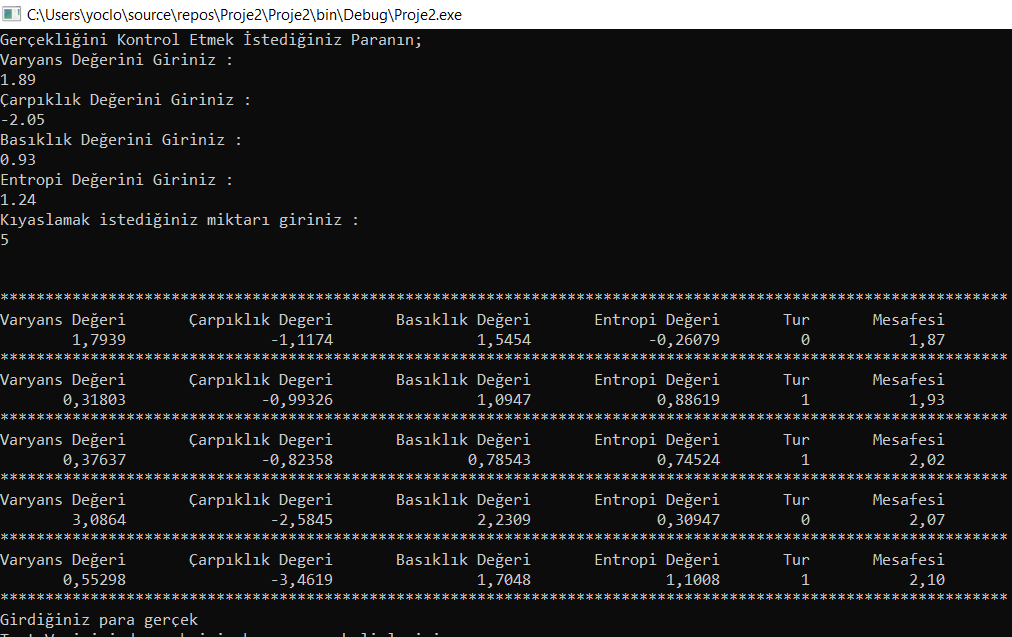
}

}

return gercekmiSahtemi;

}

### 2.b.2 Ekran görüntüleri



### 2.b.3 Açıklama

Kullanılan veri yapıları: Array, Metot, Class

Bilgileri alınan türü bilinmeyen banknotun verisetindeki diğer banknotlarla formülü uygulayarak bulduğumuz yakınlık ölçütüne bakarak sıralıyoruz. Sıraladığımız banknotların ilk k tanesine bakarak türü bilinmeyen banknote tahminliyoruz.

## 2.c Başarı ölçümü

### 2.c.1 Kodlar

int index = IndexBulma(OrnekBanknotlar);

Banknot[] SifirTestiVerisi = TestVerisiOlusturma(OrnekBanknotlar, index, 100); //Test verisi arrayi

Banknot[] BirTestiVerisi = TestVerisiOlusturma(OrnekBanknotlar, OrnekBanknotlar.Length, 100); //Test verisi arrayi

Banknot[] VeriSeti = VeriSetiOlusturma(OrnekBanknotlar, 100, index);

int dogrulukSayisi = 0;

dogrulukSayisi=BasariOlcumu(VeriSeti, SifirTestiVerisi, k1,dogrulukSayisi);

dogrulukSayisi=BasariOlcumu(VeriSeti, BirTestiVerisi, k1,dogrulukSayisi);

public static int IndexBulma(Banknot[] b)

{

int index = 0;

for (int i = 0; i < b.Length; i++)

{

if (b[i].Tur == 1)

{

break;

}

index++;

}

return index;

}

public static Banknot[] TestVerisiOlusturma(Banknot[] b, int index, int k) //Test verisini oluşturan metot

{

int sayac = index - k;

Banknot[] testVerisi = new Banknot[k];

for (int i = 0; i < k; i++)

{

testVerisi[i] = b[sayac];

sayac++;

}

return testVerisi;

}

public static Banknot[] VeriSetiOlusturma(Banknot[] b, int k, int index) //Verisetini oluşturan metot

{

int sayac = 0;

Banknot[] VeriSeti = new Banknot[b.Length - (2 \* k)];

for (int i = 0; i < b.Length; i++)

{

if (i >= index - k && i < index)

{

continue;

}

if (i >= b.Length - k && i < b.Length)

{

continue;

}

VeriSeti[sayac] = b[i];

sayac++;

}

return VeriSeti;

}

public static int BasariOlcumu(Banknot[] VeriSeti, Banknot[] OlcumVerisi, int k1,int dogrulukSayisi)

{ //Test verisindeki her banknotu tahminleyen ve başarı ölçümünü hesaplayan metot

for (int i = 0; i < OlcumVerisi.Length; i++)

{

BelirliBirDegereGoreMesafeBulma(OlcumVerisi[i], VeriSeti);

Sıralama(VeriSeti);

Yazdir(VeriSeti, k1);

int gercekmiSahtemi = 0;

gercekmiSahtemi=GercekMiSahteMi(VeriSeti, k1,gercekmiSahtemi);

if (gercekmiSahtemi == OlcumVerisi[i].Tur)

{

dogrulukSayisi++;

}

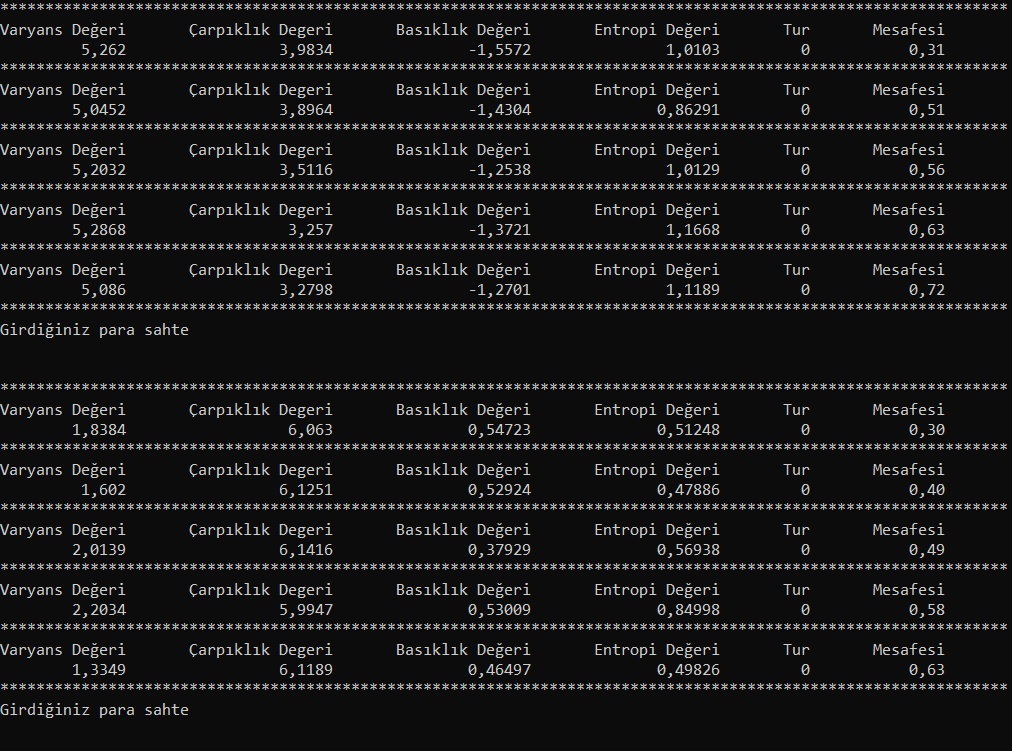
}

return dogrulukSayisi;

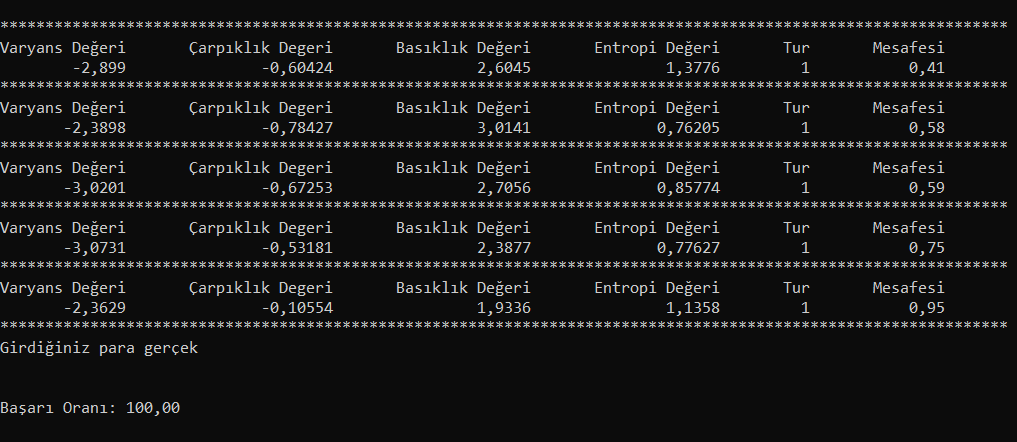
}

### 2.c.2 Ekran görüntüleri

### 



(uzuyor)…………………………………………………………………………………..………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………



### 2.c.3 Açıklama

Kullanılan veri yapıları: Array, Metot, Class

## 2.d Listeleme

### 2.d.1 Kodlar

for (int i = 0; i < VeriSeti.Length; i++)

{

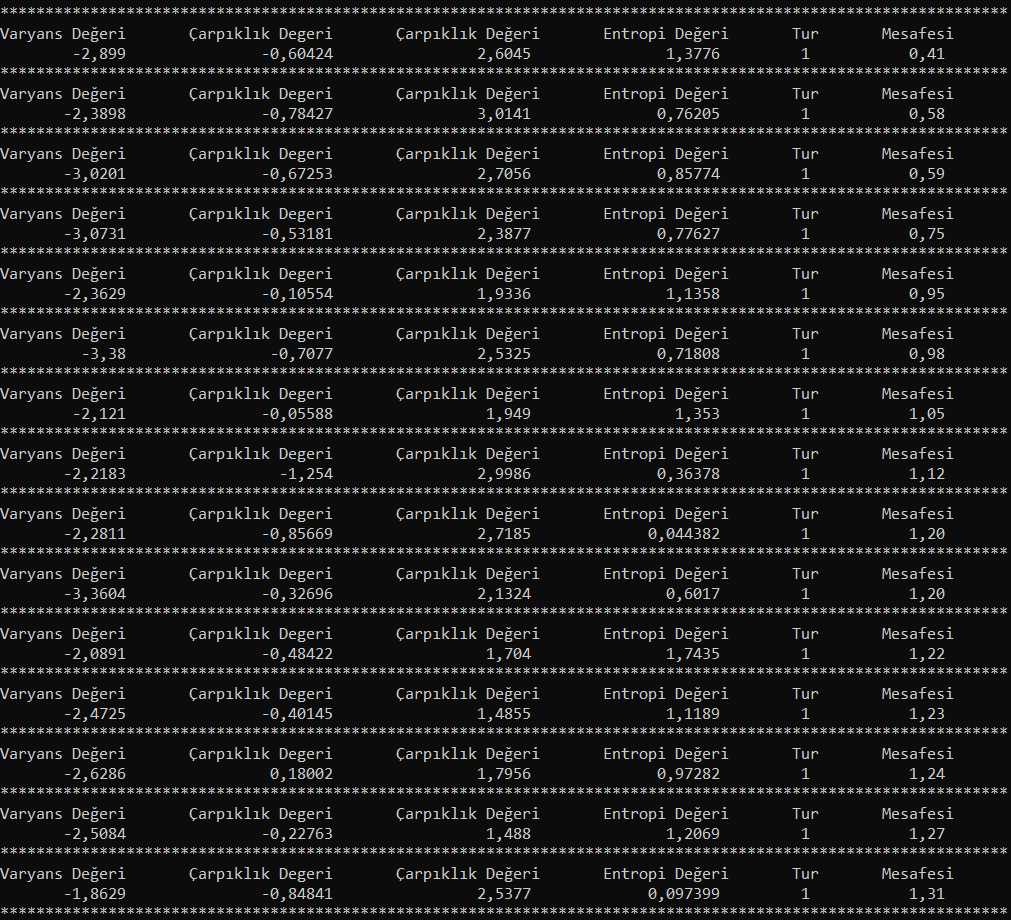
Console.WriteLine("Varyans Değeri Çarpıklık Degeri Çarpıklık Değeri Entropi Değeri Tur Mesafesi");

Console.WriteLine(string.Format("{0,14} {1,16} {2,15} {3,14} {4,3} {5,8:0.00}", VeriSeti[i].VaryansDegeri, VeriSeti[i].CarpiklikDegeri, VeriSeti[i].BasiklikDegeri, VeriSeti[i].EntropiDegeri, VeriSeti[i].Tur, VeriSeti[i].Mesafe));

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

### 2.d.2 Ekran görüntüleri



(uzuyor…)……………………………………………….………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

### 2.d.3 Açıklama

Kullanılan veri yapıları: Array, Metot, Sınıf

Verisetinin içindeki nesnelerin özelliklerini yazdırdık.

# Özdeğerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 1 Maddeleri** | **Not** | **Tahmini Not** | **Açıklama** |
| 1.a | 15 | 15 | Yapıldı |
| 1.b | 15 | 15 | Yapıldı |
| Bölüm 1. Rapor | 10 | 10 | Yapıldı |
| 2.a | 10 | 10 | Yapıldı |
| 2.b | 10 | 10 | Yapıldı |
| 2.c | 10 | 10 | Yapıldı |
| 2.d | 10 | 10 | Yapıldı |
| Bölüm 2. Rapor | 10 | 10 | Yapıldı |
| Özdeğerlendirme Tablosu | 10 | 10 | Yapıldı |

**Açıklama kısmında yapıldı, yapılmadı bilgisi veya hangi maddelerin nasıl yapıldığı kısaca yazılabilecektir.**