

**T.C.
MİLLÎ EĞİTİM BAKANLIĞI**

BİLİŞİM TEKNOLOJİLERİ

NESNE TABANLI PROGRAMLAMADA DİZİ DEĞİŞKENLER VE KOLEKSİYONLAR

Ankara, 2013

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- **PARA İLE SATILMAZ.**

İÇİNDEKİLER

AÇIKLAMALAR	ii
GİRİŞ	1
ÖĞRENME FAALİYETİ-1	3
1. DİZİLER	3
1.1. Bir Dizi Oluşturma.....	3
1.2. Diziye İlk Değer Verme.....	4
1.3. Dizi Elemanlarını Erişme.....	4
1.4. Foreach İfadesi.....	5
1.5. Dizi Kopyalama	6
UYGULAMA FAALİYETİ	8
ÖLÇME VE DEĞERLENDİRME	10
ÖĞRENME FAALİYETİ-2	12
2. KOLEKSİYONLAR	12
2.1. Koleksiyon Sınıfları	12
2.1.1 ArrayList.....	12
2.1.2. Queue - Stack.....	14
2.1.3. Hashtable	18
2.1.4. SortedList.....	19
2.2. Koleksiyon Başlatıcıları	21
UYGULAMA FAALİYETİ	22
ÖLÇME VE DEĞERLENDİRME	24
ÖĞRENME FAALİYETİ-3	25
3. PARAMETRE DİZİLERİ	25
3.1. Dizi Bağımsız Değişkenleri	25
3.1.1. “params” Dizisi Tanımlama	26
3.1.2. “params Object []”	28
3.2. “params” Dizisini Kullanma	30
UYGULAMA FAALİYETİ	31
ÖLÇME VE DEĞERLENDİRME	33
MODÜL DEĞERLENDİRME	34
CEVAP ANAHTARLARI.....	36
KAYNAKÇA	38

AÇIKLAMALAR

ALAN	Bilişim Teknolojileri
DAL/MESLEK	Veritabanı Programcılığı
MODÜLÜN ADI	Nesne Tabanlı Programlamada Dizi Değişkenler ve Koleksiyonlar
MODÜLÜN TANIMI	Nesne tabanlı programlamada dizi değişkenler ve koleksiyonların kullanımını anlatan bir öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Nesne Tabanlı Programlamada Değer Türleri modülünü tamamlamış olmak
YETERLİK	Dizileri ve koleksiyonları kullanmak
MODÜLÜN AMACI	Genel Amaç Bu modül ile gerekli ortam sağlandığında; dizileri ve koleksiyonları kullanabileceksiniz. Amaçlar 1. Dizileri kullanabileceksiniz. 2. Koleksiyon sınıflarını kullanabileceksiniz. 3. Parametre dizilerini kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Ortam: Nesne tabanlı programlama yazılımı Donanım: Bilgisayar
ÖLÇME DEĞERLENDİRME VE	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen modül sonunda ölçme aracı kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

Şu ana kadar yaptığımız örneklerde değişkenlere tek değer atadık. Bu modülü bitirdiğinizde artık diziler kullanarak bir değişkende 1'den fazla değer tutabiliriz.

Bütün örneklerimizde diziler, koleksiyonlar ve parametre dizileri kullanarak değer kümeleri oluşturduk. Parametre dizileri kullanarak herhangi bir sayıda, farklı değer türlerini dizi olarak tanımlayabileceğiz. Koleksiyon sınıfları kullanarak dizilerimizi alfabetik sıraya sokabileceğiz.

Koleksiyonlar kullanarak diziler listesinde düzenlemeler yapabileceksiniz. Diziye istediğiniz elemanı ekleyebilecek ya da silebileceksiniz.

Bu sayede diziler oluşturup, dizi elemanlarını kolayca düzenleyebileceksiniz.

ÖĞRENME FAALİYETİ-1

AMAÇ

Dizileri kullanabileceksiniz.

ARAŞTIRMA

- Nesne tabanlı programlamada dizi kullanım avantajlarının hangileri olduğunu araştırınız.
- Farklı türdeki değişkenlerin nasıl tanımlandığını hatırlayarak tartışınız.

1. DİZİLER

1.1. Bir Dizi Oluşturma

Dizi (array), sıralanmamış elemanlar serisidir. Bir dizideki tüm elemanlar aynı türdedir. Bir dizi değişkeni; eleman türünün adını, ardından da bir çift köşeli parantez ve değişken adı yazılarak tanımlanabilir.

Örnek:

```
int[] tckimlik; //Kişisel kimlik numarası
```

Bir dizi oluşturmak için, new (yeni) anahtar sözcüğünü ardından da eleman türünün adı ve köşeli parantez içinde oluşturduğunuz dizinin boyutu yazılmalıdır. Bir dizi oluşturmak aynı zamanda dizinin elemanlarına varsayılan değerleri de (eleman türünün sayısal ya da boolean (doğru/yanlış) olmasına bağlı olarak 0,false (yanlış)) atar.

Örnek:

```
int[] tckimlik;  
  
tckimlik = new int[5];
```

Yukarıdaki örnekte tckimlik isimli bir dizi değişken tanımlanmıştır, tckimlik dizi değişkeni için yeni bir 5 tam sayıdan oluşan dizi oluşturulup tckimlik isimli dizi değişkenine atanmıştır.

1.2. Diziyeye İlk Değer Verme

Bir dizi oluşturulduğunda, oluşturulan dizinin tüm elemanları türlerine bağlı olarak varsayılan bir değerle başlatılır. Bir dizinin elemanlarını belirli değerlerle başlatmak için küme parantezi içerisinde virgülle ayırarak yazabiliriz, tckimlik dizi değişkenini, değerleri 7,2,5,3 ve 6 olan 5 int değişken dizisi olarak başlatmak için aşağıdaki örneği yazabiliriz.

Örnek:

```
int[] tckimlik = new int[5]{7,2,5,3,6};
```

Not: Küme parantezi arasındaki değerlerin sayısı, tam olarak oluşturulan dizinin boyutuna eşit olmalıdır.

Örnek:

```
int[] tckimlik = new int[3]{7,2,5,3,6}; //Yanlış kullanım  
int[] tckimlik = new int[4]{7,2,5,3}; //Doğru kullanım
```

Not: tckimlik dizisi int türünde tanımlanırsa, bu dizide double, string, struct(yapı) ya da int dışında herhangi bir tür saklanamaz. Bir dizi tanımlanırken eleman türü, dizi içinde saklayacağınız eleman türü ile eşleşmek zorundadır.

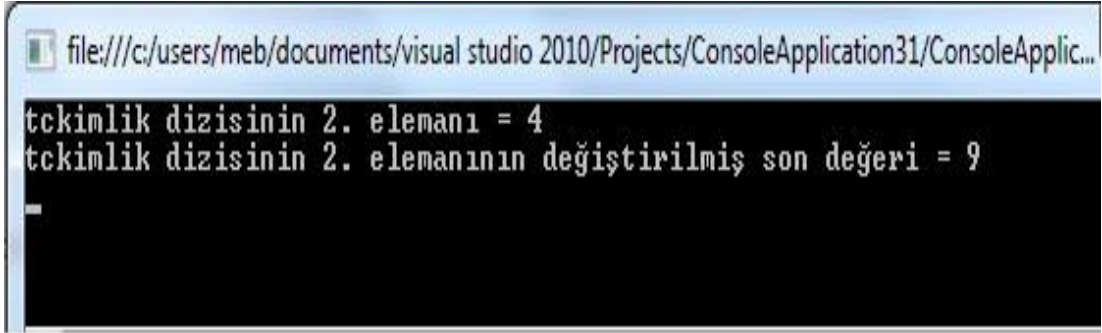
1.3. Dizi Elemanlarını Erişme

Dizi elemanlarına erişmek için, istediğiniz elemanın hangisi olduğunu gösteren ifadenin yazılması gerekir.

Aşağıdaki örnek tckimlik dizisinin 2. elemanının içeriğini tcno isimli int türündeki bir değişkene nasıl atandığını ve daha sonra tcno isimli değişkenin değiştirilmiş son değerini (9 sayı değeri) atayarak dizinin içeriğinin nasıl değiştirildiğini göstermektedir.

```
namespace ConsoleApplication31  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int[] tckimlik = { 2, 3, 4 };  
            int tcno;  
            tcno = tckimlik[2];  
            Console.WriteLine("tckimlik dizisinin 2. elemanı = "+tcno);  
            tcno = 9;  
            tckimlik[2] = tcno;  
            Console.WriteLine("tckimlik dizisinin 2. elemanının değiştirilmiş son değeri = " + tckimlik[2]);  
            Console.ReadKey();  
        }  
    }  
}
```

Resim1.1: Dizi elemanlarına erişme



```
file:///c:/users/meb/documents/visual studio 2010/Projects/ConsoleApplication31/ConsoleApplic...
tckimlik dizisinin 2. elemanı = 4
tckimlik dizisinin 2. elemanının değıştirilmiş son değeri = 9
```

Resim1.2: Ekran çıktısı

Not: Dizilerde indeks numarası olarak adlandırılan erişim numaraları sıfırdan başlar yani bir dizinin ilk elemanı 1. dizinde değil, 0. dizinde bulunur. 1. dizin kullanılarak dizinin 2. elemanına erişilebilir.

1.4. Foreach İfadesi

Döngüler program içerisinde tekrarlanması gereken ifadeler veya kod bloklarını tekrar tekrar yazmak yerine tek bir yapıda yazarak ifadelerin veya kod bloklarının tekrarlanmasını sağlar.

Foreach ifadesi (döngüsü), bir dizi belirten ifadenin veya bir koleksiyonun her elemanı için yapısındaki kodları çalıştıran ifadedir. Foreach ifadesinin kullanım şekli aşağıdaki gibidir.

```
foreach (degisken_tipi degisken in dizi)
{
    //kodlar;
}
```

Tanımlanan değişken, sırasıyla belirtilen bir dizi içerisindeki elemanları tutar. Foreach ifadesi (döngüsü) döngüye ait değişkeni kullanmamıza imkân sağlar. Değişkeni istediğimiz şekilde, istediğimiz işleme sokabiliriz. Ancak döngüye müdahale edemeyiz.

```

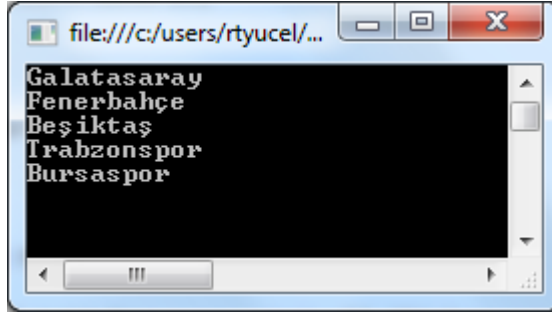
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] takimler = new string[] { "Galatasaray", "Fenerbahçe", "Beşiktaş", "Trabzonspor", "Bursaspor" };
            foreach (string tkm in takimler)
            {
                Console.WriteLine(tkm);
            }
        }
    }
}

```

Resim 1.3: Foreach ifadesi

Yukarıdaki Resim 1.3'te foreach ifadesi kullanılarak dizi içerisinde yer alan değerlerin, alt alta ekrana yazdırılmasını görüyoruz. Programın ekran çıktısı aşağıdaki gibidir.



Resim 1.4: Ekran çıktısı

1.5. Dizi Kopyalama

Bir dizinin kopyası oluşturmak istendiğinde, iki şey yapmak gerekir. İlk olarak kopyalanacak dizi ile aynı türde ve aynı uzunlukta yeni bir dizi oluşturulmalıdır.

Örnek:

```

int[] tckimlik = {7,5,3,2};
int[] kopya = new int[4];

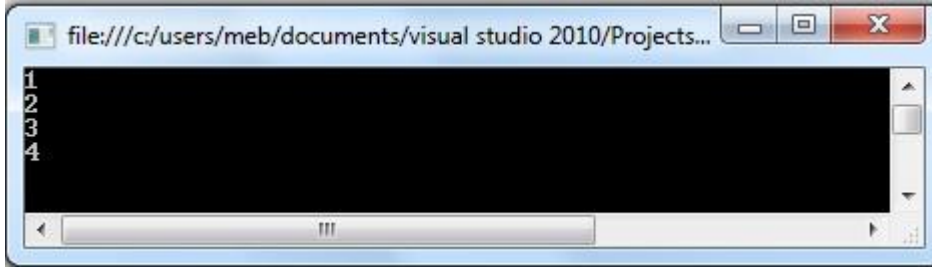
```

Yapılması gereken ikinci şey ise yeni dizinin içindeki değerleri orijinal dizinin içine eşitlemektir. Aşağıdaki örnekte gösterildiği gibi bu bir for ifadesi kullanılarak yapılabilir.

```
namespace ConsoleApplication30
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] num = { 1, 2, 3, 4 };
            int[] kopyanum = new int[4];
            for (int i = 0; i < 4; i++)
            {
                kopyanum[i] = num[i];
                Console.WriteLine(kopyanum[i]);
            }
            Console.ReadLine();
        }
    }
}
```

Resim 1.5: Dizi kopyalama

Yukarıdaki Resim 1.5'te num isimli 4 elemanlı bir diziye ait elemanlar, yeni oluşturulmuş kopyanum isimli 4 elemanlı bir diziye for ifadesi kullanılarak kopyalandığı gösterilmektedir. Ekran çıktısı aşağıdaki gibidir.



Resim 1.6: Ekran çıktısı

UYGULAMA FAALİYETİ

Dizi tanımlayarak foreach ifadesini kullanınız.

İşlem Basamakları	Öneriler
➤ Yeni bir proje oluşturunuz.	➤ Ctrl+N, File-New Project ya da Recent Project nesnelere kullanabilirsiniz.
➤ Konsol uygulaması seçiniz.	➤ Uygulamayı oluştururken dili seçmeyi unutmayınız.
➤ Program gövdesi Main metodu içinde bir dizi ve sayıların toplamının tutulacağı değişkeni tanımlayınız.	➤ <code>int[] sayi={1,2,3,4};</code> <code>int toplam;</code>
➤ Main metodu içinde foreach ifadesi kullanarak dizi elemanlarının toplamını bir değişken içinde tutunuz.	➤ <code>foreach(int i in sayi)</code> <code>{</code> <code> toplam=toplam+i;</code> <code>}</code>
➤ Dizi elemanlarının toplam sonucunu ekrana yazan kodu tanımlayınız.	➤ <code>Console.WriteLine(toplam);</code>
➤ Programı çalıştırarak test ediniz.	➤ F5 kısayol tuşunu kullanabilirsiniz.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadıklarınız için **Hayır** kutucuklarına (X) işareti koyarak öğrendiklerinizi kontrol ediniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Konsol uygulaması oluşturduğunuz mu?		
2. Gerekli komut satırlarını program gövdesine yazdınız mı?		
3. Dizi tanımladınız mı?		
4. Foreach ifadesini yazdınız mı?		
5. Dizi elemanlarının toplam sonucunu ekrana yazan kodu yazdınız mı?		
6. Ekran çıktısını adım adım incelediniz mi?		

DEĞERLENDİRME

Değerlendirme sonunda “**Hayır**” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “**Evet**” ise “Ölçme ve Değerlendirme” ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Dizilerle ilgili aşağıdakilerden hangisi yanlıştır?
A) Sıralanmamış öğeler serisidir.
B) Bir dizideki tüm öğeler farklı veri türündedir.
C) Bir dizi tanımlamak için new anahtar sözcüğü kullanılmalıdır.
D) Bir dizi, eleman türünün adının ardından bir çift köşeli parantez ve değişken adı yazılarak tanımlanabilir.
2. Aşağıdaki hangisi bir dizi tanımlaması olamaz?
A) `int[] no = new int[3]{1,2,3};`
B) `int[] no = new int[5]{4,5,2,1,3};`
C) `int no = new int[4]{4,5,2,1,3};`
D) `int[] no = new int[6]{12,13,3,2,17,8};`
3. Aşağıdakilerden hangisi yanlıştır?
A) Foreach ifadesi döngüye ait değişkeni kullanmamıza imkân sağlar.
B) Foreach ifadesinde döngüye müdahale edebiliriz.
C) Foreach ifadesi bir dizi belirten ifadenin yapısındaki kodları çalıştıran ifadedir.
D) Döngüler, program içerisinde tekrarlanması gereken ifadeleri tekrar tekrar yazmak yerine tek bir yapıda yazarak ifadelerin tekrarlanmasını sağlar.
4. Aşağıdakilerden hangisi yanlıştır?
A) `string[] isim = new string[3];`
B) `string[] isim = new string[3]{“ali”,“Mehmet”,“serdar”};`
C) `string[] isim = new string[4]{“ayşe”,“melek”,“Selma”,“seçil”};`
D) `int[] isim = new string[3]{“veli”,“zeki”,“hakan”};`
5. Aşağıdaki ifadelerden hangisi yanlıştır?
A) Bir dizi kopyalamak istendiğinde, kopyalanacak dizi ile aynı türde ve aynı uzunlukta yeni bir dizi oluşturulmalıdır.
B) Dizi elemanlarına erişmek için, istediğiniz elemanın hangisi olduğunu gösteren dizinin yazılması gerekir.
C) Bir dizinin ilk elemanı 1. dizisinde bulunur.
D) Bir dizi oluşturulduğunda, oluşturulan dizinin tüm öğeleri türlerine bağlı olarak varsayılan bir değerle başlatılır.

6. Aşağıdakilerden hangisi int türünde 3 elemanlı ve 10, 20, 30 değerlerini tutan kimlik dizisinin doğru tanımıdır?
- A) `int[] kimlik = new int[3]{10,20,30};`
 - B) `int kimlik = new int[3]{10,20,30};`
 - C) `int kimlik = int[3];`
 - D) `int[] kimlik = int[3]{10,20,30};`

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetini geçiniz.

ÖĞRENME FAALİYETİ-2

AMAÇ

Koleksiyon sınıflarını kullanabileceksiniz.

ARAŞTIRMA

- Nesne tabanlı programlamada diziler ile koleksiyonlar arasındaki farkların neler olduğunu araştırınız.
- Dizi tanımlama kurallarını hatırlayarak tartışınız.

2. KOLEKSİYONLAR

2.1. Koleksiyon Sınıfları

System.Collections veri saklama için kullanılan sınıflarımızın bulunduğu isim alanıdır. Bu öğrenme faaliyetinde Koleksiyon isim alanı içerisinde yer alan sınıfları inceleyeceğiz.

2.1.1 ArrayList

ArrayList, kısaca sınırları dinamik olarak değişebilen diziler olarak tanımlanır. Dizinin benzeridir fakat sadece object tipinden verileri saklar. Yani içeriğin bir kısmı int türünde, bir kısmı string türünde veya bool türünde olabilir. Belirli bir türde olma zorunluluğu yoktur.

Ayrıca belirli bir sınır vermemize gerek yoktur. Yeni nesne eklendikçe boyutunu otomatikman artırır. Add, Remove, Sort ve indeksleme metotlarına sahip olması işlem yapmayı kolaylaştırır.

Klasik bir dizide karşılaşılabileceğimiz sorunlar genel olarak şu şekildedir;

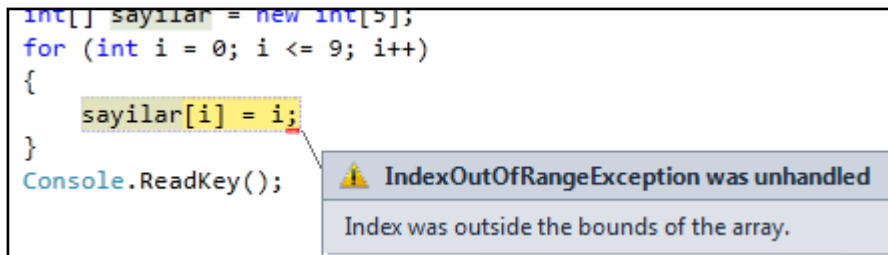
- Dizi boyutları sınırlıdır ve sabittir,
- Dizilerin tüm elemanları aynı türden olmalıdır,
- Kullanmadığımız dizi elemanlarından dolayı bellek alanları gereksiz yere işgal edilmektedir.

ArrayList ve klasik bir dizinin arasındaki farklılardan bazılarını inceleyelim.

- 5 elemanlık int türünde sayılar isminde bir dizi tanımlayalım ve bu dizi içerisine 10 adet veri girmeye çalışalım.

```
static void Main(string[] args)
{
    int[] sayilar = new int[5];
    for (int i = 0; i <= 9; i++)
    {
        sayilar[i] = i;
    }
    Console.ReadKey();
}
```

Yukarıdaki kodları çalıştırdığımız zaman Resim 2.1’de görülen *dizi sınır değeri aşımı* (*IndexOutOfRangeException*) hatasını alırız.



Resim 2.1: IndexOutOfRangeException hatası

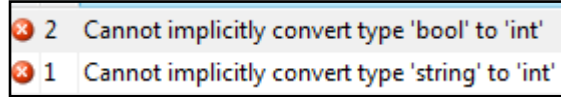
Oysa burada klasik bir dizi yerine dinamik bir yapıda bulunan ArrayList kullanmış olsaydık, dizi boyutu ile ilgili bir hata almazdık.

```
static void Main(string[] args)
{
    ArrayList sayilar=new ArrayList();
    for (int i = 0; i <= 9; i++)
    {
        sayilar.Add(i);
    }
    Console.ReadKey();
}
```

- Şimdi de bir önceki adımda tanımladığımız sayılar dizimizin içerisine string ve bool türünde veriler eklemeye çalışalım.

```
static void Main(string[] args)
{
    int[] sayilar=new int[5];
    sayilar[0] = "Türkiye";
    sayilar[1] = true;
    | Console.ReadKey();
}
```

Yukarıdaki kodları derlemeye çalıştığımız zaman “Türkiye” ve “true” ibarelerin altlarının kırmızı ile çizilmiş olduğunu ve burada bir hata olduğunu görürüz. Resim 2.2’de görünen int türünden string türüne (*Cannot implicitly convert type 'string' to 'int'*) ve int türünden bool türüne (*Cannot implicitly convert type 'bool' to 'int'*) dönüştürememe hatalarını alırız.



```
2 Cannot implicitly convert type 'bool' to 'int'
1 Cannot implicitly convert type 'string' to 'int'
```

Resim 2.2: Tür Dönüşüm hataları

Aynı atama işlemlerini bir de ArrayList kullanarak yapmayı deneyelim.

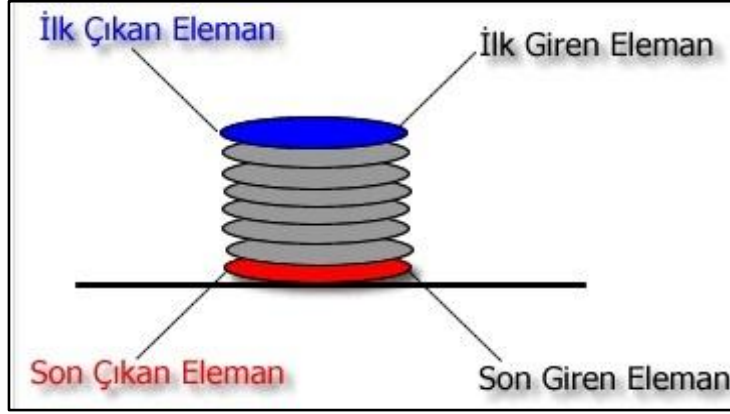
```
static void Main(string[] args)
{
    ArrayList sayilar = new ArrayList();
    sayilar.Add("Türkiye");
    sayilar.Add(true);
}
```

Bu kodların derlenmesi sırasında herhangi bir hata mesajı almaz ve verilerimizi başarılı bir şekilde *sayilar* isimli ArrayList içerisine aktarırız. Görüldüğü gibi ArrayListlere herhangi bir veri türünde veri ekleyebiliriz.

2.1.2. Queue - Stack

Queue (Kuyruk) ve Stack (Yığın) da içerisinde birden fazla veri depolayabildiğimiz yapılarıdır.

Queue (Kuyruk) sınıfının özelliği ilk giren ilk çıkar (first-in first-out, FIFO) prensibine göre çalışır. Bir eleman arkaya sıradan katılır (enqueue işlemi) ve sırayı önden terk eder(dequeue işlemi).



Resim 2.3: Queue (Kuyruk) sınıfı

Şekilde görüldüğü gibi Queue (Kuyruk) koleksiyon sınıfında elemanlar koleksiyona arkadan katılırlar ve ilk giren eleman kuyruktan ilk çıkan eleman olur.

```

using System;
using System.Collections;
namespace Queueornek1
{
    class Class1
    {
        static void Main(string[] args)
        {
            Queue ku = new Queue(6);
            ku.Enqueue("Ahmet");
            ku.Enqueue("Selim");
            ku.Enqueue("Zeki");
            ku.Enqueue("Yılmaz");
            ku.Enqueue(123);
            ku.Enqueue(false);
            Console.WriteLine("Çıkan eleman {0}", ku.Dequeue().ToString());
            Console.WriteLine("Çıkan eleman {0}", ku.Dequeue().ToString());
            Console.WriteLine("Çıkan eleman {0}", ku.Dequeue().ToString());
            Console.WriteLine("Çıkan eleman {0}", ku.Dequeue().ToString());
            Console.WriteLine("-----");
            IEnumerator dizi = ku.GetEnumerator();
            while (dizi.MoveNext())
            {
                Console.WriteLine("Güncel eleman {0}", dizi.Current.ToString());
            }
            Console.ReadKey();
        }
    }
}

```

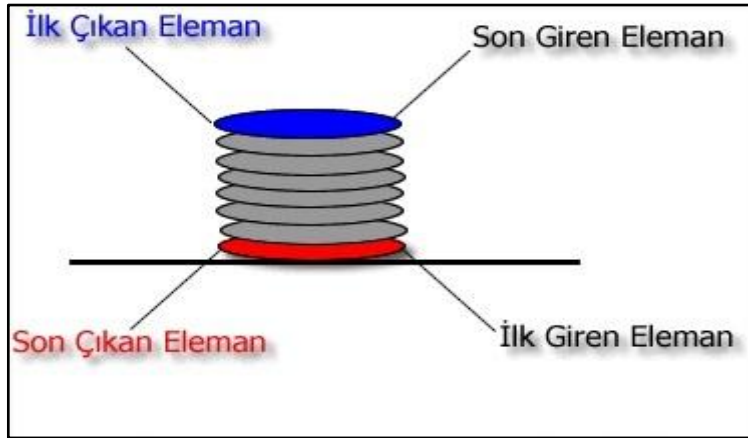
Resim 2.4: Queue (Kuyruk) sınıfı kullanımı

Resim 2.4'te ku isimli 6 elemanlı Queue (Kuyruk) dizisi tanımlanmış. Elemanları (Ahmet, Selim, Zeki, Yılmaz, 123, false) diziyeye eklemek için enqueue metodu kullanılmıştır. Dequeue metodu kullanılarak ilk giren (en alttaki) elemanı verirken, aynı zamanda bu elemanı ku dizisinden siler. Ku dizisinin elemanlarını IEnumerator ara yüzünden bir nesneye (dizi) aktarıyoruz. Current metodu ile dizi nesnesinde yer alan güncel elemanları elde ederiz. Böylelikle ku dizisindeki elemanlar koleksiyona arkadan katılır ve ilk giren eleman kuyruktan ilk çıkan eleman olur. Ekran çıktısı aşağıdaki gibidir.

```
file:///c:/users/meb/documents/visual studio 2010/Projects/Console...
Çıkan eleman Ahmet
Çıkan eleman Selim
Çıkan eleman Zeki
Çıkan eleman Yılmaz
-----
Güncel eleman 123
Güncel eleman False
-
```

Resim 2.5: Ekran çıktısı

Stack (yığın) sınıfının özelliği “son giren ilk çıkar (last-in first-out, LIFO)” prensibine göre çalışmalarıdır. Bir eleman yığına üstten katılır (push işlemi) ve yığını yine üstten terk eder (pop işlemi). Bu yöntemi mutfaktaki kirli tabaklara benzetebiliriz. Her zaman ilk önce en üste konan tabak yıkanır.



Resim 2.6: Stack (Yığın) sınıfı

Şekilde görüldüğü gibi stack koleksiyonunda yer alan elemanlardan son girene ulaşmak oldukça kolaydır. Oysaki ilk girdiğimiz elemana ulaşmak için, bu elemanın üstünde yer alan diğer tüm elemanları silmemiz gerekmektedir.

```

using System;
using System.Collections;
namespace stack
{
    class Class1
    {
        static void Main(string[] args)
        {
            Stack stek = new Stack(6);
            stek.Push("ALİ");
            stek.Push("MEHMET");
            stek.Push("ZEHRRA");
            stek.Push("ZEKİ");
            stek.Push(16);
            stek.Push(true);
            Console.WriteLine("Çıkan eleman {0}", stek.Pop().ToString());
            Console.WriteLine("Çıkan eleman {0}", stek.Pop().ToString());
            Console.WriteLine("Çıkan eleman {0}", stek.Pop().ToString());
            Console.WriteLine("Çıkan eleman {0}", stek.Pop().ToString());
            Console.WriteLine("-----");
            IEnumerator dizi = stek.GetEnumerator();
            while (dizi.MoveNext())
            {
                Console.WriteLine("Güncel eleman {0}", dizi.Current.ToString());
            }
            Console.ReadKey();
        }
    }
}

```

Resim 2.7: Stack (Yığın) sınıfı kullanımı

Resim 2.7’de stek isimli 6 elemanlı Stack (Yığın) dizisi tanımlanmış. Elemanları (ALİ, MEHMET, ZEHRRA, ZEKİ, 16, True) diziyeye eklemek için Push methodu kullanılmıştır. Pop methodu kullanılarak son giren (kalan) elemanı verirken, aynı zamanda bu elemanı stek dizisinden siler. Stek dizisinin elemanlarını IEnumerator arayüzünden bir nesneye (dizi) aktarıyoruz. Current methodu ile dizi nesnesinde yer alan güncel elemanları elde ederiz. Böylelikle stek dizisindeki elemanlara son katılan ilk çıkar, bir eleman yığına üstten katılır ve yığını yine üstten terk eder. Ekran çıktısı aşağıdaki gibidir.

```

file:///c:/users/meb/documents/visual studio 2010/Projects/ConsoleApplication7...
Çıkan eleman True
Çıkan eleman 16
Çıkan eleman ZEKİ
Çıkan eleman ZEHRRA
-----
Güncel eleman MEHMET
Güncel eleman ALİ

```

Resim 2.8: Ekran çıktısı

2.1.3. Hashtable

Hashtable (Karışık masa) koleksiyon sınıfında veriler key-value (anahtar-değer) çiftleri şeklinde tutulmaktadır. Hashtable koleksiyon sınıflarında key ve value değerleri herhangi bir veri türünde olabilir. Temel olarak bunların hepsi DictionaryEntry (Sözlük Giriş) nesnesidir. Key – value çiftleri hash tablosu adı verilen bir tabloda saklanır.

Key değerleri tektir ve değiştirilemez. Yani bir key-value çiftini koleksiyonumuza eklediğimizde, bu değer çiftinin value değerini değiştirebilirken, key değerini değiştiremeyiz.

Hashtable koleksiyonu verilere hızlı bir şekilde ulaşmamızı sağlayan bir kod yapısına sahiptir. Bu nedenle özellikle arama maliyetlerini düşürdüğü için tercih edilmektedir.

Örnek: Bir konsol uygulaması oluşturarak, Hashtable (Karışık Masa) koleksiyon sınıfı kullanılarak ali, serdar ve utku isiminde, yaşları sırasıyla 42, 28 ve 34 olan değerleri tablo şeklinde ekranda sırasıyla yazan program kodu yazılsın.

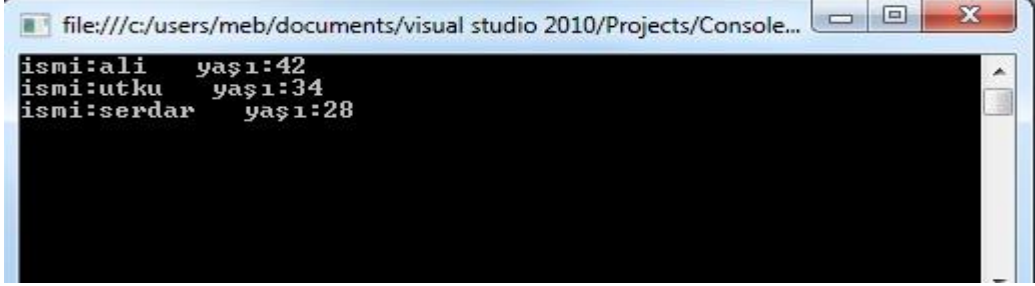
```
using System;
using System.Collections;
using System.Text;

namespace ConsoleApplication38
{
    class Program
    {
        static void Main(string[] args)
        {
            Hashtable yaslar = new Hashtable();
            yaslar["ali"] = 42;
            yaslar["serdar"] = 28;
            yaslar["utku"] = 34;
            foreach (DictionaryEntry element in yaslar)
            {
                string isim = (string)element.Key;
                int yas = (int)element.Value;
                Console.WriteLine("ismi:{0} yaşı:{1}", isim, yas);
            }
            Console.ReadKey();
        }
    }
}
```

Resim 2.9: Hashtable koleksiyon sınıfı

Resim 2.9’ da yaslar isimli bir hashtable(karışık masa) sınıfı tanımlanarak anahtar (ali, serdar, utku) ve değer (42, 28, 34) çifti tablosu oluşturulmuştur. DictionaryEntry (Sözlük giriş) nesnesi kullanılarak anahtar (key) ve değer (value) dizilerine erişim sağlanmıştır. İsim

ve yas isminde anahtar ve deęer çifti oluşturarak yaslar sınıfındaki tablo deęerleri ekrana yazdırılmıştır. Ekran çıktısı aşağıdaki gibidir.



```
file:///c:/users/meb/documents/visual studio 2010/Projects/Console...
ismi:ali yaşı:42
ismi:utku yaşı:34
ismi:serdar yaşı:28
```

Resim 2.10: Ekran çıktısı

2.1.4. SortedList

SortedList (Sıralı Liste) sınıfı, hashtable sınıfı ile benzerlik göstermektedir. SortedList sınıfı kullanarak anahtarları, deęerlerle ilişkilendirebiliriz. Hashtable'dan farklı olarak anahtarlar dizisinin her zaman sıralanmış olmasıdır.

Bir SortedList sınıfına bir anahtar-deęer çifti eklendiğinde anahtar, anahtarlar dizisinin sırasını bozmamak için doğru dizine alfabetik sıraya göre eklenir. Daha sonra deęerde, aynı dizinli deęerler dizisine eklenir. SortedList sınıfı, bir eleman eklediğinizde ya da bir elemanı kaldırdığınızda, otomatik olarak anahtarlar ve deęerleri eş zamanlı hale getirir. Buda anahtar-deęer çiftlerini SortedList'e istediğiniz sırada ekleyebileceğiniz anlamına gelir, her zaman anahtarlara göre sıralanır.

Hashtable sınıfında olduğu gibi, bir SortedList aynı anahtardan iki tane içermez. Bir SortedList boyunca yineleme yapmak için bir foreach ifadesi kullanıldığında DictionaryEntry (Sözlük giriş) elde edilir. DictionaryEntry nesnelere key (anahtar) özelliği tarafından sıralanmış olarak döner.

Örnek: Bir konsol uygulaması oluşturarak SortedList (Sıralı Liste) koleksiyon sınıfı kullanılarak semih, metin ve ahmet isminde, yaşları sırasıyla 12, 67 ve 26 olan deęerleri tablo şeklinde ekranda alfabetik sırayla yazan program kodu yazılsın.

```

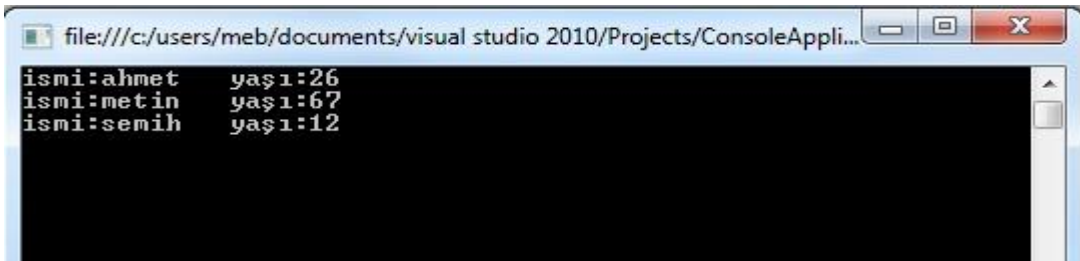
using System;
using System.Collections;
using System.Text;

namespace ConsoleApplication38
{
    class Program
    {
        static void Main(string[] args)
        {
            SortedList yaslar = new SortedList();
            yaslar["semih"] = 12;
            yaslar["metin"] = 67;
            yaslar["ahmet"] = 26;
            foreach (DictionaryEntry element in yaslar)
            {
                string isim = (string)element.Key;
                int yas = (int)element.Value;
                Console.WriteLine("ismi:{0} yaşı:{1}", isim, yas);
            }
            Console.ReadKey();
        }
    }
}

```

Resim 2.11: SortedList koleksiyon sınıfı

Resim 2.11’de yaslar isimli bir SortedList (Sıralı Liste) koleksiyon sınıfı tanımlanarak anahtar (semih, metin, ahmet) ve değer (12, 67, 26) çifti tablosu oluşturulmuştur. DictionaryEntry (Sözlükgiriş) nesnesi kullanılarak anahtar (key) ve değer (value) dizilerine erişim sağlanmıştır. İsim ve yas isminde anahtar ve değer çifti oluşturularak yaslar sınıfındaki tablo değerleri, isimler alfabetik sıraya göre ekrana yazdırılmıştır. Ekran çıktısı aşağıdaki gibidir.



```

file:///c:/users/meb/documents/visual studio 2010/Projects/ConsoleAppli...
ismi:ahmet yaşı:26
ismi:metin yaşı:67
ismi:semih yaşı:12

```

Resim 2.12: Ekran çıktısı

2.2. Koleksiyon Başlatıcıları

Koleksiyon sınıflarında yapılan örnekler, bir koleksiyona o koleksiyon için en uygun yöntemi (ArrayList için Add, Stack için Pop gibi) kullanarak tek tek elemanları nasıl ekleyeceğinizi gösterir. Ayrıca diziler tarafından desteklenen sözdizimine (komut) oldukça benzer bir söz dizimi kullanarak, bazı koleksiyon türlerini tanımlarken aynı zamanda başlatabiliriz. Örneğin aşağıdaki ifade, numaralar ArrayList dizisini oluşturur ve başlatır. Sürekli olarak Add yöntemi çağırmanıza gerek kalmaz.

```
ArrayList numaralar = new ArrayList(){10, 3, 4, 6, 9, 13};
```

Bu sözdizimini (komut satırı) sadece Add yöntemini destekleyen koleksiyonlar için kullanabiliriz (Stack ve Queue sınıfları desteklemez).

Hashtable gibi anahtar – değer çifti alan daha karmaşık koleksiyonlar için, aşağıdaki gibi her anahtar – değer çiftini başlatıcı listesinde anonim tür olarak belirleyebiliriz.

```
Hashtable yas = new Hashtable(){{"Serdar", 24}, {"Ahmet", 32}, {"Murat", 27}};
```

Her çiftteki ilk eleman anahtar, ikincisi ise değerdir.

UYGULAMA FAALİYETİ

Koleksiyon sınıflarını tanımlayarak ArrayList sınıfını kullanınız.

İşlem Basamakları	Öneriler
➤ Yeni bir proje oluşturunuz.	➤ Ctrl+N, File-New Project ya da Recent Project nesnelerini kullanabilirsiniz.
➤ Konsol uygulaması seçiniz.	➤ Uygulamayı oluştururken dili seçmeyi unutmayınız.
➤ Program gövdesi Main methodu içinde bir ArrayList koleksiyon sınıfı tanımlayınız.	➤ ArrayList sayi=new ArrayList();
➤ 10 elemanlı bir dizi tanımlayınız.	➤ İnt[] num=new int[10]{1,2,3,4,5,6,7,8,9,10};
➤ Tanımlanan dizideki elemanları tanımlı olan koleksiyon sınıfındaki diziyeye ekleyiniz.	➤ For (int i=0;i<10;i++) { Sayi.Add(num[i]); }
➤ Koleksiyon sınıfındaki 2. elemanı siliniz.	➤ Sayi.RemoveAt(2);
➤ Daha sonra koleksiyon sınıfındaki 5. Elemanın yerine, dizideki elemanlardan başka bir eleman değeri atayınız.	➤ Sayi.Insert(5, 2345);
➤ Tanımlı koleksiyon sınıfının en sonki halinde bulunan elemanların değerini ekrana yazan kodu tanımlayınız.	➤ for (int i=0;i<10;i++) { Console.WriteLine(sayi[i]); }
➤ Programı çalıştırarak test ediniz.	➤ F5 kısayol tuşunu kullanabilirsiniz.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadıklarınız için **Hayır** kutucuklarına (X) işareti koyarak öğrendiklerinizi kontrol ediniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Konsol uygulaması oluşturduunuz mu?		
2. Gerekli komut satırlarını program gövdesine yazdınız mı?		
3. ArrayList koleksiyon sınıfını tanımladınız mı?		
4. Dizi tanımlaması yaptınız mı?		
5. Tanımlı dizi elemanlarını, koleksiyon sınıfı dizisi elemanları olarak eklediniz mi?		
6. Add yöntemini kullandınız mı?		
7. RemoveAt yöntemini kullandınız mı?		
8. Insert yöntemini kullandınız mı?		
9. Ekran çıktısını adım adım incelediniz mi?		

DEĞERLENDİRME

Değerlendirme sonunda “Hayır” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınızı “Evet” ise “Ölçme ve Değerlendirme” ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Koleksiyonlarla ilgili aşağıdakilerden hangisi yanlıştır?
A) Dizinin elemanlarını özel yollardan bir araya toplayan sınıflardır.
B) Koleksiyonlar veriyi saklamak, gruplamak için kullanılır.
C) Koleksiyonlar dizilerin geliştirilmiş versiyonudur.
D) Değişkenler bir koleksiyon sınıfıdır.
2. Aşağıdakilerden hangisi bir koleksiyon sınıfı değildir?
A) ArrayList B) Method C) SortedList D) HashTable
3. Aşağıdakilerden hangisi ArrayList sınıfında kullanılan yöntemlerden değildir?
A) RemoveAt B) Insert C) Exit D) Add
4. HashTable ilgili olarak aşağıdakilerden hangisi yanlıştır?
A) Veriler key – value (anahtar – değer) çiftleri şeklinde tutulmaktadır.
B) Key ve value değerleri herhangi bir veri türünde olabilir.
C) Key ve value çiftleri hash tablosu adı verilen bir tabloda saklanır.
D) Key değerleri değiştirilebilen bilgilerdir.
5. Aşağıdakilerden hangisi yanlıştır?
A) HashTable sınıfı için Push yöntemi kullanılabilir.
B) Stack sınıfı için Pop yöntemi kullanılabilir.
C) Queue sınıfı için Dequeue yöntemi kullanılabilir.
D) ArrayList sınıfı için Add yöntemi kullanılabilir.
6. Queue sınıfı ile ilgili aşağıdakilerden hangisi yanlıştır?
A) Elemanlar koleksiyona arkadan katılır.
B) İlk giren ilk çıkar prensibine göre çalışmalarıdır.
C) Bir eleman sıraya arkadan katılır ve sırayı önden terk eder.
D) Bir eleman sıraya arkadan katılır ve yine sırayı arkadan terk eder.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetini geçiniz.

ÖĞRENME FAALİYETİ-3

AMAÇ

Parametre dizilerini kullanabileceksiniz.

ARAŞTIRMA

- Nesne tabanlı programlamada parametre dizilerinin kullanımının ne gibi avantajlar sağladığını araştırınız.
- Metot tanımlama kurallarını hatırlayarak tartışınız.

3. PARAMETRE DİZİLERİ

Parametre olarak herhangi bir sayıda farklı türlerde bağımsız değişkenler alabilen metotlar yazılmak istendiğinde, parametre dizileri kullanılabilir.

3.1. Dizi Bağımsız Değişkenleri

Parametre olarak geçirilmiş bir değerler kümesindeki en küçük değeri bulmak için metot yazılmak istendiğinde bir dizi kullanılabilir.

Örnek: Parametre dizisi kullanılarak dizinin en küçük elemanın bulan programın kodu yazılsın.

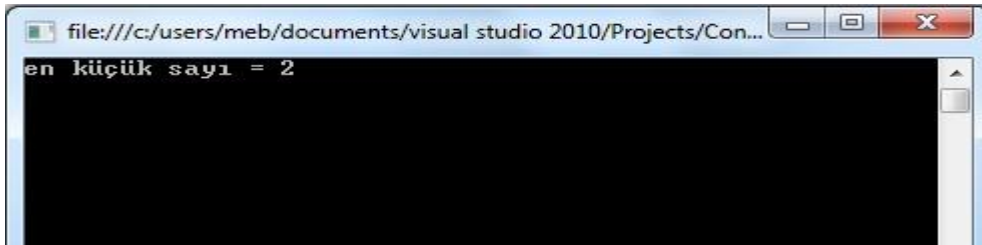
```

class Program
{
    public static int Min(params int[] paramsList)
    {
        int c = paramsList[0];
        foreach (int i in paramsList)
        {
            if (i < c)
            {
                c=i;
            }
        }
        return c;
    }
    static void Main(string[] args)
    {
        int[] dizi = new int[3];
        dizi[0] = 5;
        dizi[1] = 9;
        dizi[2] = 2;
        int enKucuk = Min(dizi);
        Console.WriteLine("En küçük sayı:" + enKucuk);
        Console.ReadLine();
    }
}

```

Resim 3.1: Parametre dizisi

Resim 3.1’de verilen 3 sayıdan (5, 9, 2) en küçüğünü bulmak için Min isimli, int türünde bir dizi biçiminde tek bir parametresi olan, bir statik metod yazılmıştır. Main programda, en küçüğü bulunmak istenen 3 sayı, dizi isimli 3 elemanlı diziye atanmıştır. Dizi değerleri en küçüğünün bulunması için Min isimli metoda gönderilmiştir. Metottan dönen en küçük değer ekrana yazdırılmıştır. Ekran çıktısı aşağıdaki gibidir.



Resim 3.2: Ekran çıktısı

3.1.1. “params” Dizisi Tanımlama

Params anahtar sözcüğü bir dizi parametresi düzenleyicisi olarak kullanılır. Bir metodun aldığı parametre sayısı çok sayıda olabilir önceden ne kadar parametre alınacağı bilinemeyebilir. Böyle durumlarda “params” anahtar kelimesi kullanılabilir. Bunun yanında

bir eleman kümesini parametre olarak geçirmek istiyorsanız params anahtar sözcüğü kullanılabilir.

- Params anahtar sözcüğünden sonra kullanılacak parametre tek boyutlu bir dizi gibi tanımlanmalıdır.
- Params anahtar sözcüğü parametre tanımlamada sadece bir kez kullanılır.
- Bir metoda birden fazla params anahtar sözcüğü kullanılamaz.
- Params ile tanımlanan değişkenden sonra herhangi bir değişken tanımlanamaz.
- Params anahtar sözcüğü ile tanımlanan değişken diğer değişken tanımlamalarından önce olamaz.
- Parametre dizisinin en sonunda yer alır.
- Metot çağrılırken parametre verileri virgül (,) ile ayırt edilerek gönderilir.

Örnek: Params anahtar sözcüğü tanımlanarak dizinin en küçük elemanın bulan programın kodu yazılsın.

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

namespace ConsoleApplication29
{
    class Program
    {
        public static int Min(params int[] paramList)
        {
            int c = paramList[0];
            foreach (int i in paramList)
            {
                if (i < c)
                {
                    c = i;
                }
            }
            return c;
        }
        static void Main(string[] args)
        {
            int min=Program.Min(7,8,9,4);
            Console.WriteLine("en küçük sayı = " + min);
            Console.ReadLine();
        }
    }
}
```

Resim 3.3: “params” Dizisi tanımlama

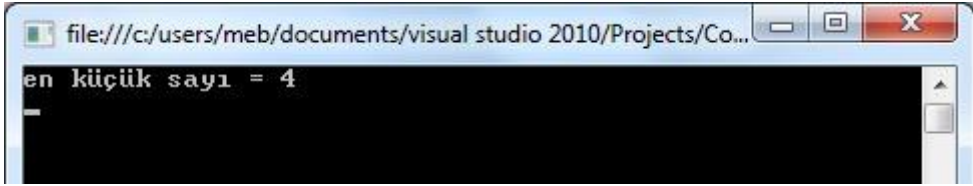
Resim 3.3’de verilen 4 sayıdan (7, 8, 9, 4) en küçüğünü bulmak için params anahtar sözcüğü kullanarak Min metodunu istediğiniz kadar bağımsız değişken olarak çağırabilirsiniz. Main programda 4 sayı, Min isimli metoda sayıların en küçüğü bulunmak üzere gönderilmiştir.

```
int min=Program.Min(7,8,9,4);
```

Derleyici bu kodu aşağıdaki koda çevirir.

```
int[] dizi=new int[3];  
Dizi[0]=7;  
Dizi[1]=8;  
Dizi[2]=9;  
Dizi[3]=4;  
int min=Program.Min(dizi);
```

Metottan dönen en küçük değer ekrana yazdırılmıştır. Ekran çıktısı aşağıdaki gibidir.



Resim 3.4: Ekran çıktısı

3.1.2. “params Object []”

“int” türü bir parametre dizisi bir metot çağrısında, herhangi bir sayıda int bağımsız değişkeni kullanımına izin verir. Params object türü, bağımsız değişkenlerin sayısı değil de türleri değişirse, değer türlerini nesnelere dönüştüren kodu üretir. Herhangi bir türde bağımsız değişken geçmesine olanak sağlayan herhangi bir sayıda object bağımsız değişkeni kabul eden bir yöntem bildirmek için object türü bir parametreler dizisi kullanılabilir.

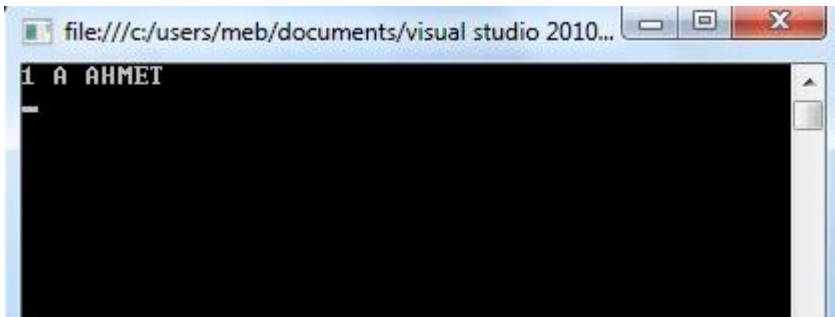
Örnek: Params object türü tanımlanarak sayı, harf ve kelime gibi farklı türde olan değişken değerlerini ekrana yazan programın kodu yazılsın.

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

namespace ConsoleApplication29
{
    class Program
    {
        public static void Use(params object[] list)
        {
            for (int i = 0; i < list.Length; i++)
            {
                Console.Write(list[i] + " ");
            }
            Console.WriteLine();
        }
        static void Main(string[] args)
        {
            Use(1, 'A', "AHMET");
            Console.ReadLine();
        }
    }
}
```

Resim 3.5: “params object []”

Resim 3.5’de verilen 3 değeri (1, A, AHMET) ekrana yazdırabilmek için params object türü kullanarak farklı değişken türlerinin geçişine izin verebilirsiniz. Main programda ki 3 değer(1, A, AHMET) , Use isimli metoda ekrana yazdırılmak üzere gönderilmiştir. Ekran çıktısı aşağıdaki gibidir.



Resim 3.6: Ekran çıktısı

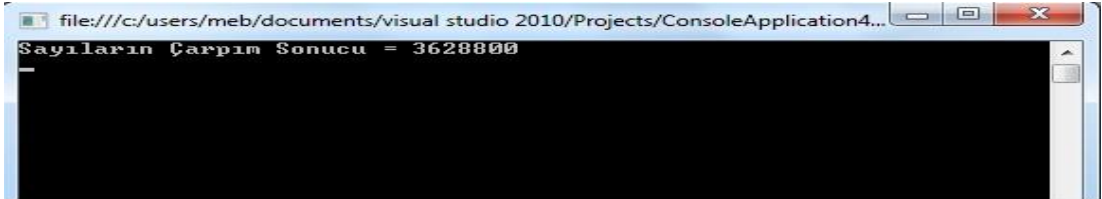
3.2. “params” Dizisini Kullanma

Aşağıdaki örnekte deneme.carpım isminde bir static yöntem yazacaksınız. Bu yöntemin amacı kendisine geçirilen int türü değişkenlerin çarpımlarını hesaplayıp, sonucu int olarak döndürmektir. Bunu deneme.carpım yöntemi params int[] parametresi alacak şekilde yazarak yapacaksınız.

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication29
{
    class deneme
    {
        public static int carpım(params int[] paramList)
        {
            int carpımsonucu = 1;
            foreach (int i in paramList)
            {
                carpımsonucu *= i;
            }
            return carpımsonucu;
        }
        static void Main(string[] args)
        {
            Console.WriteLine("Sayıların Çarpım Sonucu = "+deneme.carpım(1,2,3,4,5,6,7,8,9,10));
            Console.ReadLine();
        }
    }
}
```

Resim 3.7: “params” Dizisi kullanma

Resim 3.7 ‘de verilen 10 sayının (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) çarpımlarının sonucunu bulmak için params dizisi kullanarak carpım isimli int değerlerden oluşan dizi biçiminde bir statik metot yazılmıştır. Main programda, çarpımlarının sonucu bulunması için 10 sayı (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) carpım isimli metoda gönderilmiştir. Metottan dönen çarpım sonucu değeri ekrana yazdırılmıştır. Ekran çıktısı aşağıdaki gibidir.



Resim 3.8: Ekran çıktısı

UYGULAMA FAALİYETİ

Parametre dizileri tanımlayarak, “params ” dizisi kullanınız.

İşlem Basamakları	Öneriler
➤ Yeni bir proje oluşturunuz.	➤ Ctrl+N, File-New Project ya da Recent Project nesnelerini kullanabilirsiniz.
➤ Konsol uygulaması seçiniz.	➤ Uygulamayı oluştururken dili seçmeyi unutmayınız.
➤ Program gövdesi Main metodu içine int toplam=Program.toplam(9,4,5,6,19); Console.WriteLine(toplamen); Console.Read(); ifadelerini kullanınız	➤ Büyük küçük harf ayırımına ve noktalama işaretlerine dikkat ediniz. Kodları Main bloğundaki küme parantezleri içerisine yazınız.
➤ Bir statik int params dizi parametresi alan ve int değer döndüren “toplam” isminde metot tanımlayınız.	➤ public static int toplam(params int[] paramList)
➤ Metot içerisinde int türünde 2 değişken tanımlayıp, dizinin ilk değerlerini bu değişkenlere atayınız.	int c=paramList[0]; int d=paramList[0];
➤ Metot içerisinde params dizisi içerisindeki en küçük sayıyı bulan kodu tanımlayınız.	➤ foreach(int i in paramList) { if (i<c) c=i; }
➤ Daha sonra Metot içerisinde params dizisi içerisindeki en büyük sayıyı bulan kodu tanımlayınız.	➤ foreach(int i in paramList) { if (i>d) d=i; }
➤ Metotta son olarak c ve d değişkenlerini toplayarak geri gönderiniz.	➤ return c+d;
➤ Programı çalıştırarak test ediniz.	➤ F5 kısayol tuşunu kullanabilirsiniz.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadıklarınız için **Hayır** kutucuklarına (X) işareti koyarak öğrendiklerinizi kontrol ediniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Konsol uygulaması oluşturduunuz mu?		
2. Gerekli komut satırlarını program gövdesine yazdınız mı?		
3. Bir params dizi parametresi alan ve int değer döndüren metot tanımladınız mı?		
4. Metot içerisinde params dizisi içerisindeki en küçük sayıyı ve en büyük sayıyı bulan kodu tanımladınız mı?		
5. Metot içerisinde en küçük sayı ve en büyük sayının toplamını bulan kodu tanımladınız mı?		
6. Tanımladığınız params dizisi metodunu çağırdınız mı?		
7. Ekran çıktısını adım adım incelediniz mi?		

DEĞERLENDİRME

Değerlendirme sonunda “Hayır” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “Evet” ise “Ölçme ve Değerlendirme” ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise (D), yanlış ise (Y) yazınız.

1. () Parametre dizileri, parametre olarak herhangi bir sayıda farklı türlerde bağımsız değişkenler alabilen metotlar yazılmak istendiğinde kullanılabilir.
2. () Params anahtar sözcüğü, bir dizi parametresi düzenleyicisi olarak kullanılabilir.
3. () int türü bir parametre dizisi bir metot çağrısında sadece bir tane int bağımsız değişken kullanımına izin verir.
4. () Params object türü bağımsız değişkenlerin türleri değil de sayıları değişirse değer türlerini nesnelere dönüştüren kodu üretir.
5. () Public static int max(params int[] paramlist) bu kod satırında params anahtar sözcüğü doğru kullanılmıştır.
6. () Public static min(params int[] param) bu kod satırında params anahtar sözcüğü doğru kullanılmıştır.
7. () Bir eleman kümesini parametre olarak geçirmek istediğimizde params anahtar sözcüğünü kullanabiliriz.
8. () Parametre olarak geçireceğimiz değerler farklı veri tiplerine sahipse params anahtar sözcüğünü object tipinde bir dizi ile kullanırız.
9. () Params anahtar sözcüğü belirsiz sayıda parametrelerle işlem yapmamızı sağlar.
10. () Public int carpım(params int[] deger) bu kod satırında params anahtarı bu metoda bir kere integer değer geçirebileceğimizi ifade eder.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise “Modül Değerlendirme” ye geçiniz.

MODÜL DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Aşağıdaki dizi tanımlamalarından hangisi doğrudur?
A) `int[] tcno = new int[5];`
B) `int[] tcno = int[5];`
C) `[]int tcno = new int[5];`
D) `tcno int[] = new int[5]`
2. Dizi ile ilgili aşağıdakilerden hangisi yanlıştır?
A) Bir dizi tanımlanırken öge türü, dizi içinde saklayacağınız öge türü ile eşleşmek zorundadır.
B) Bir dizi `int` türünde tanımlanırsa, bu dizide `double`, `string`, `struct` (yapı) türündeki veriler saklanamaz.
C) Dizi elemanlarına erişmek için, istediğiniz elemanın hangisi olduğunu gösteren indeks numarasıyla çağrılması gerekir.
D) Dizi sıralanmış öğeler serisidir.
3. Numara isimli 5 elemanlı (1, 2, 3, 4, 5) bir dizi tanımlamak istiyorsak, aşağıdaki kod satırlarından hangisini yazmalıyız?
A) `int[] numara = {1, 2, 3, 4, 5};`
B) `int[] numara = new int[5]{1, 2, 3, 4, 5};`
C) `int numara = new[]{1, 2, 3, 4, 5};`
D) `int[] numara = new int[]`
4. `ArrayList` koleksiyon sınıfından `removeAt` yöntemi hangi amaç için kullanılır?
A) `ArrayList` 'in sonuna bir eleman eklemek için kullanılır.
B) `ArrayList` 'in ortasına bir eleman eklemek için kullanılır.
C) `ArrayList` 'den bir elemanı silmek için kullanılır.
D) `ArrayList` 'e bir eleman kopyalamak için kullanılır.
5. Aşağıdakilerden hangisi doğrudur?
A) `Queue` (kuyruk) sınıfının özelliği son giren ilk çıkar prensibine göre çalışmalarıdır.
B) `Queue` (kuyruk) koleksiyon sınıfında elemanlar koleksiyona önden katılır ve ilk giren eleman kuyruktan ilk çıkan elemandır.
C) `Stack` (yığın) sınıfının özelliği son giren ilk çıkar, prensibine göre çalışmalarıdır.
D) `Stack` (yığın) sınıfında bir eleman yığına yandan katılır ve yığını üstten terk eder.
6. `Hashtable` koleksiyon sınıfı ile ilgili aşağıdakilerden hangisi yanlıştır?
A) `Hashtable` (karışık masa) koleksiyon sınıfında veriler `key – value` (anahtar – değer) çiftleri şeklinde tutulmaktadır.
B) `Key` değerleri çifttir ve değiştirilebilir.
C) `Hashtable` koleksiyon sınıfı verilere hızlı bir biçimde ulaşmamızı sağlayan kod yapısına sahiptir.
D) `Hashtable` sınıfı arama maliyetlerini düşürdüğü için tercih edilmektedir.

7. Aşağıdaki yazımlardan hangisi yanlıştır?
- A) Hashtable deneme = new hashtable(){{“sema”, 12}, {“serkan”, 27}};
B) SortedList deneme = new sortedList();
C) ArrayList deneme = new arrayList(){1, 2, 3, 4, 5};
D) Stack deneme = Stack (4);
8. Aşağıdakilerden hangisi bir koleksiyon sınıfı değildir?
- A) Hashtable B) SortedList C) Params D) Queue
9. Aşağıdakilerden hangisi yanlıştır?
- A) Parametre dizileri farklı türlerde bağımsız değişkenler alabilen metotlar yazılmak istendiğinde kullanılabilir.
B) Params anahtar sözcüğü bir dizi parametresi düzenleyicisi olarak kullanılabilir.
C) Params nesnesi bağımsız değişkenlerin türleri değişirse, değer türlerini nesnelere dönüştüren kodu üretir.
D) Parametre dizilerinde herhangi bir türde bağımsız değişken kullanımı sınırlıdır.
10. Aşağıdaki parametre dizi tanımlamalarından hangisi doğrudur?
- A) int denem = paramList[0]; C) denem int = paramList[0];
B) paramList = int denem[0]; D) int denem = param;

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmenimize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

1	B
2	C
3	B
4	D
5	C
6	A

ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

1	D
2	B
3	C
4	D
5	A
6	D

ÖĞRENME FAALİYETİ-3'ÜN CEVAP ANAHTARI

1	Doğru
2	Doğru
3	Yanlış
4	Yanlış
5	Doğru
6	Yanlış
7	Doğru
8	Doğru
9	Doğru
10	Yanlış

MODÜL DEĞERLENDİRMENİN CEVAP ANAHTARI

1	A
2	D
3	B
4	C
5	C
6	B
7	D
8	C
9	D
10	A

KAYNAKÇA

- Sharp John(Çeviri:Ümit TEZCAN), **Adım Adım Microsoft Visual C# 2008**, Arkadaş Yayınevi, Ankara, 2008.