TİP DÖNÜŞÜMLERİ

İÇİNDEKİLER

- Bilinçsiz Tip Dönüşümü
- Bilinçli Tip Dönüşümü
- Dönüşüm Metotları
  - ToString Metodu
  - Parse Metodu
  - Convert Metotları

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Tip dönüşümleri hakkında,
  - Tip dönüşümlerinin türleri hakkında,
  - Dönüşüm için kullanılan metotlar hakkında bilgi sahibi olabilir ve programlarınızı yazarken tip dönüşümünü uygulayabilirsiniz.

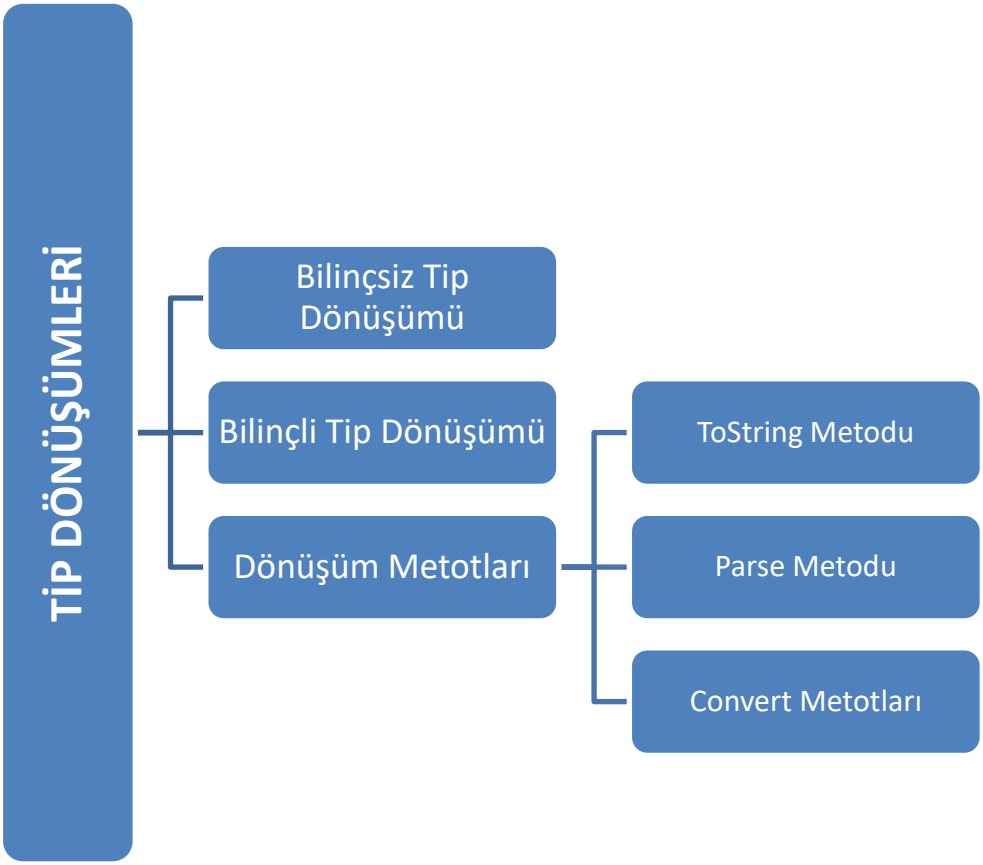
Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

Seda KAYA

ÜNİTE

6


TİP DÖNÜŞÜMLERİ
Bilinçsiz Tip Dönüşümü
Bilinçli Tip Dönüşümü
Dönüşüm Metotları
ToString Metodu
Parse Metodu
Convert Metotları

GİRİŞ

Programlama dillerinde, daha önceki ünitelerde de gördüğünüz üzere birçok veri tipi mevcuttur. Aynı veri tipleri arasında yapılabilecek birçok işlem vardır. Aynı veri tipleriyle işlem yaptığımız gibi bazı durumlarda farklı veri tipleriyle de işlem yapmak isteyebiliriz. Normalde farklı veri tipleri arasında işlem yapılmazken işlem yapılması gereken durumlarda veri tipinin dönüştürülmesi gerekmektedir. Örneğin, güncel yaşı hesaplamak için *int* veri tipindeki bir yıl sayısından, textbox' tan alacağımız doğum yılını çıkarmak isteyebiliriz. Bu durumda textbox' tan string veri tipinde bir veri geleceği için bu değeri önce *int* veri tipine dönüştürmemiz gerekir. Ya da *date* tipindeki bir tarihi ekrana *string* veri tipinde yazdırmak isteyebiliriz.

Herhangi bir veri tipindeki bir veriyi, bir değişkende saklamak istediğimiz zaman verinin veri tipinde bellekte bir yer açılır. Bu yüzden bir değişkenin veri tipini dönüştürmek istediğimiz zaman, verinin bellekteki yerini değiştiririz. Veri küçük bellek adresinden büyük bellek adresine veya büyük bellek adresinden küçük bellek adresine taşınabilir. Büyük bellekten küçük bellek adresine taşındığında da veri kayıpları meydana gelir(Mastar, Vural, & Eriş, 2012). Bu konu ünite içerisinde ayrıntılı açıklanacaktır.

Tip dönüşümleri, yapılma şekline göre ikiye ayrılır. Bunlar; bilinçli ve bilinçsiz tip dönüşümüdür. Bazı kaynaklarda bilinçli dönüşüm için açık dönüşüm, bilinçsiz dönüşüm için de kapalı dönüşüm ifadesini görebilirsiniz. Her iki ifade de aynı anlamı taşımaktadır. Bu ünitede veri tipi dönüşümleri detaylı bir şekilde ele alınacaktır.

BİLİNÇSİZ (IMPLICIT) TİP DÖNÜŞÜMÜ

Herhangi bir operatör kullanmadan derleyicinin kendisinin yaptığı dönüşüme "bilinçsiz dönüşüm" denir.

Herhangi bir operatör kullanmadan derleyicinin kendisinin yaptığı tip dönüşümüne bilinçsiz tip dönüşümü ya da kapalı dönüşüm denir. Bilinçsiz tip dönüşümünde küçük bellekli bir veri tipinden büyük bellekli bir veri tipine dönüşüm yapılabilirken, büyük bellekli veri tipinden küçük bellekli veri tipine dönüşüm yapılamaz. Çünkü derleyici veri kaybının olacağını bilir ve buna izin vermez. Bu durumda program çalışmayı durdurur ve hata alırız. Örneklere geçmeden önce hangi veri tipinin hangi veri tipine dönüştürülebileceğini Tablo 6.1' de inceleyelim.

Tablo 6.1. Veri Tipleri Arasında Dönüştürülebilirlik

	ushort	**short**	**uint**	**int**	**ulong**	**long**	**float**	**decimal**	**double**
byte	+	+	+	+	+	+	+	+	+
sbyte		+		+		+	+	+	+
ushort			+	+	+	+	+	+	+
short				+		+	+	+	+
char	+		+	+	+	+	+	+	+
uint					+	+	+	+	+
int						+	+	+	+

ulong							+	+	+
long							+	+	+
float									+

Veri tiplerinin hepsinin birbirine bilinçsiz dönüşümü mümkün değildir.

Tablo 6.1' de görüldüğü gibi veri tiplerinin hepsinin birbirine dönüşümü mümkün değildir. Büyük veri tipleri küçük veri tiplerine dönüştürülürken veri kaybı olacağından derleyici hata verir. Küçük veri tipleri büyük veri tiplerine dönüştürülürken; büyük tipten dolayı eklenen fazla bitler "sıfır" ile doldurulur. Bu durum değişken içinde bulunan veriyi etkilemeyeceği için veri kaybı olmaz (Algan, 2013).

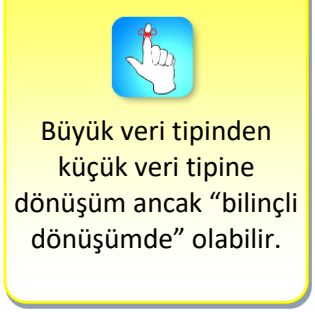
Örnek

- int veri tipindeki bir değişken byte veri tipine dönüştürülemez. Çünkü int veri tipi bellekte 32 bit yer kaplarken byte veri tipi 8 bit yer kaplar.

Şimdi bir örnek uygulama ile tip dönüşümlerini daha detaylı bir şekilde inceleyelim. Aşağıdaki örnekte bir form açılmış ve bir buton yerleştirilmiştir. Butona tıkladığımız zaman kodlar çalışacak ve değerleri label' a basacaktır. Farklı veri tiplerinde a, b ve c değişkenlerine değerler atanmıştır. Bu değişkenlerden *byte* tipindeki a değişkeninin değeri *int* tipindeki x değişkenine, *short* tipindeki b değişkeninin değeri *long* tipindeki y değişkenine, *uint* tipindeki c değişkeninin değeri z değişkenine atanmıştır. Program çalıştırıldığında hiçbir hata almadan farklı "label" lara bu değerler yazdırıldı. Bunu Şekil 6.1' de görebilirsiniz.

```
public Form1()
{
    InitializeComponent();
}

private void Form1_Load(object sender, EventArgs e)
{


}

private void button1_Click(object sender, EventArgs e)
{
    byte a = 2;
    short b=-3;
    uint c = 5;

    int x;
    long y;
    float z;

    x = a;
    y = b;
    z = c;
```

```
            label1.Text = x.ToString();
            label2.Text = y.ToString();
            label3.Text = z.ToString();


        }
```



Büyük veri tipinden küçük veri tipine dönüşüm ancak "bilinçli dönüşümde" olabilir.

Şekil 6.1. Program Sonucu

Aynı kod içinde büyük veri tipindeki bir veriyi küçük veri tipindeki bir değişkene atamayı deneyelim. Yukarıdaki kod bloğu içinde z değişkeninin veri tipini *byte* yapıp *uint* veri tipinde olan c değişkenini z' ye atadığımız zaman derleyicimiz programı derlemeyecek ve Şekil 6.2' deki gibi c değişkeninin altını çizip "Cannot implicitly convert type 'uint' to 'byte'." hatasını verecektir. Bu hata da " 'uint' tipi 'byte' tipine bilinçsiz olarak dönüştürülemedi." anlamına gelmektedir. Büyük veri tipinden küçük veri tipine dönüşüm ancak bilinçli dönüşümde olabilir.

```
private void button1_Click(object sender, EventArgs e)
{
    byte a = 2;
    short b=-3;
    uint c = 5;

    int x;
    long y;
    byte z;

    x = a;
    y = b;
    z = c;

    labe
    labe
    label3.Text = z.ToString();


}
```

(local variable) uint c

Cannot implicitly convert type 'uint' to 'byte'. An explicit conversion exists (are you missing a cast?)

Şekil 6.2. Program Sonucu

Yaptığımız bu uygulamada dikkat etmemiz gereken bir hususta a, b ve c değişkenlerinin veri tiplerinin değişmemiş olmasıdır. Sadece x, y ve z veri tiplerine değerler atanırken bu değerlerin veri tipi değişir.

Hesaplama işlemi yaparken tek tek verilerin veri tiplerini değiştirmemize gerek yoktur. Önemli olan sonucun doğru veri tipi ile tanımlanmış olmasıdır. Bu durumu daha iyi anlamak için aşağıdaki kodları inceleyelim. Toplama işlemindeki tüm değişkenler *float* veri tipine dönüştürülebildiği için programı çalıştırdığımızda herhangi bir hata almadık. Eğer değişkenlerden biri *float'* a dönüştürülemeseydi yine Şekil 6.2'deki gibi hata alırdık.

```
private void button1_Click(object sender, EventArgs e)
 {
     byte a = 2;
     short b= -3;
     uint c = 5;

     float x;

     x = a + b + c;

     label1.Text = x.ToString();

 }
```

Büyük veri tiplerini küçük veri tiplerine dönüştürmek için bir dönüştürme operatörü kullanmalıyız.

BİLİNÇLİ (EXPLICIT) TİP DÖNÜŞÜMÜ

"Bilinçsiz dönüşüm" de veri tipleri uygun veri tiplerine derleyici tarafından dönüştürülebiliyordu. Ama her veri tipi herhangi bir veri tipine dönüştürülemiyordu. Büyük veri tiplerini küçük veri tiplerine dönüştürmeye çalıştığımız zaman hata ile karşılaşıyorduk. Bazı durumlarda büyük veri tiplerini küçük veri tiplerine dönüştürmek isteyebiliriz. O zaman yapmamız gereken bir dönüştürme operatörü kullanmaktır. Bilinçli tip dönüşümünde, kendisine dönüştürülmesi amaçlanan veri tipi parantez içine alınır ve yanına değişken ya da sabit değer yazılır. Bu durumda veri kaybı göze alınarak verimiz istediğimiz türe dönüşmüştür. Tabi bu durum ancak sayısal değerler içinde yapılabilir.

Örnek

- Float veri tipindeki 10.5 sayısını int veri tipine dönüştürürken bu sayı 10 olarak değiştirildikten sonra dönüştürülecektir.

Aşağıdaki kodları ve Şekil 6.3' ü inceleyiniz. Veri dönüştürülürken virgülden sonraki kısımda veri kaybı meydana gelmiştir. *Float* sayının sadece tam kısmını almak istediğimiz için bu dönüşüm yapılmıştır. Ama veri kaybının yazdığımız kodları etkileyeceği olumsuz bir durum varsa buna dikkat etmeliyiz. Dönüşüm yaptıktan sonra elimizde doğru bir değer kalmayabilir. Dolayısıyla çok gerekmedikçe bilinçli dönüşüm işlemi yapılmamalıdır.

```
private void button1_Click(object sender, EventArgs e)
 {
     float a=10.5f;

     int x=(int)a;

     label1.Text = x.ToString();

 }
```

Şekil 6.3. Program Sonucu

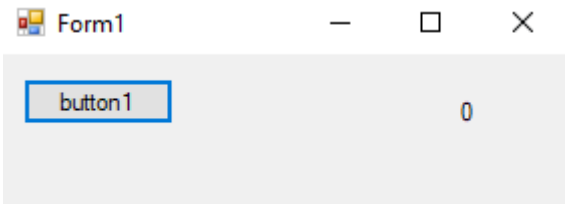
Veri kaybının kodlarımızın çalışmasında herhangi bir olumsuzluğa sebep olmasının istenmeyen bir sonuç olduğu söylenmiştir. Yukarıdaki örnekte sadece virgülden sonraki değerde veri kaybı oldu. Bazı durumlarda değerin tamamında veri kaybı olabilir.

Örnek

- Ushort veri tipi değeri en fazla 65535 sayısını alabilir.
- Biz int veri tipindeki 65536 sayısını ushort veri tipine dönüştürmek istersek sonuç 0 olacaktır. Çünkü ushort veri tipinin boyutu aşılmıştır. Yani veri tamamen yanlış olacaktır.
- Aşağıdaki kod bloğunu ve şekil 6.4' ü inceleyiniz.

```
private void button1_Click(object sender, EventArgs e)
 {
     int a=65536;
     ushort x= (ushort)a;
     label1.Text = x.ToString();
 }
```

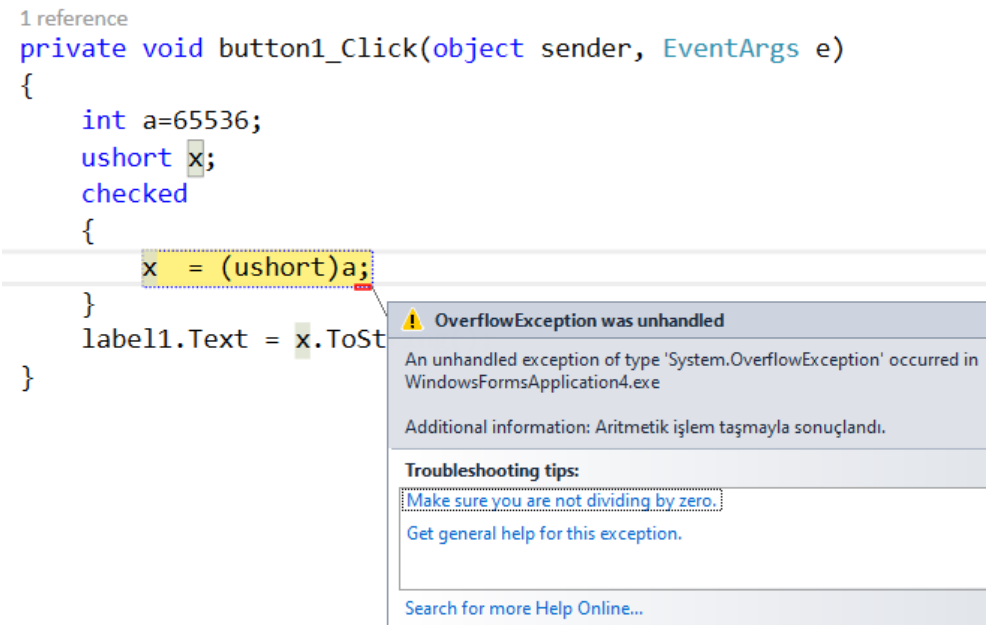
Veri kaybının olacağı durumlarda "checked "kontrolü ile "tip dönüşümü" engellenebilir.



Şekil 6.4. Program Sonucu

Veri kaybının olacağı durumlarda bir kontrol sağlayıp tip dönüşümünün yapılmasını engellemek isteyebiliriz. Bunu da *checked* ile yapabiliriz. Yukarıdaki kod bloğuna *checked* kontrolünü eklediğimiz zaman program hata verecektir. Böylelikle veri kaybı olmaması için veri tipi dönüşümü engellenecektir.

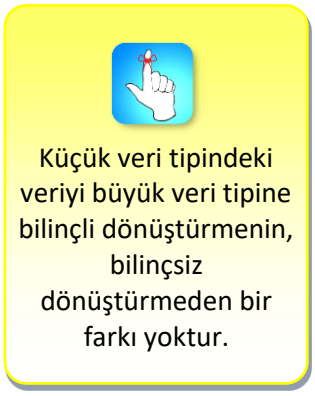
```
 private void button1_Click(object sender, EventArgs e)
 {
     int a=65536;
     ushort x;
     checked
     {
         x  = (ushort)a;
     }
     label1.Text = x.ToString();
 }
```

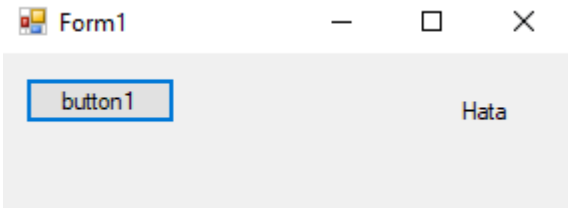


Şekil 6.5. Program Sonucu

Şekil 6.5'teki programın hata vermesini engellemek için de bir *try- catch* kontrolü ekleyerek kodları biraz daha işlevsel hale getirebiliriz. Hata verdiğinde programın durdurmasını engelleyip hatayı form üzerinde bir mesaj olarak verelim. Böylece veri kaybı olduğu durumda ekrana "Hata" mesajı yazdırılacak, olmadığı durumda değer yazdırılacaktır.

```
private void button1_Click(object sender, EventArgs e)
 {
     int a=65536;
     ushort x;
     checked
     {

         try
         {
             x = (ushort)a;
             label1.Text = x.ToString();
         }
         catch (OverflowException )
         {
             label1.Text = "Hata";
         }
     }
 }
```





Şekil 6.6. Program Sonucu

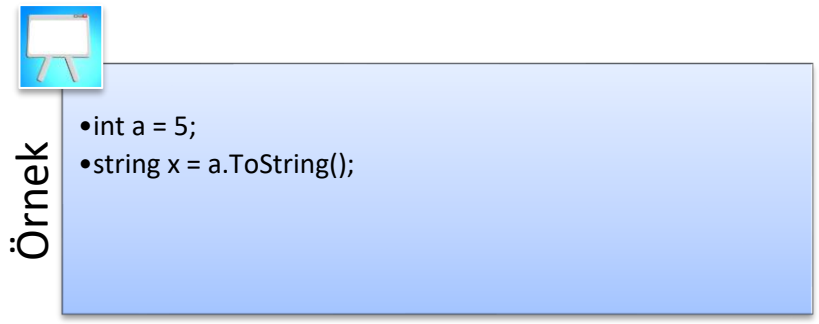
Unchecked kontrolü checked kontrolünün tam tersidir. Normal bir kod ile unchecked kontrolü içinde kalan kodun bir farkı yoktur. Sadece checked bloğu içinde bazı yerleri unchecked durumuna getirmek istendiğinde unchecked kontrolü kullanılır (Algan, 2013).

Küçük veri tipindeki veriyi büyük veri tipine bilinçli dönüştürmenin, bilinçsiz dönüştürmeden bir farkı yoktur. Bir dönüşüm operatörü kullansak da kullanmasak da aynı sonucu verecektir.

DÖNÜŞÜM METOTLARI

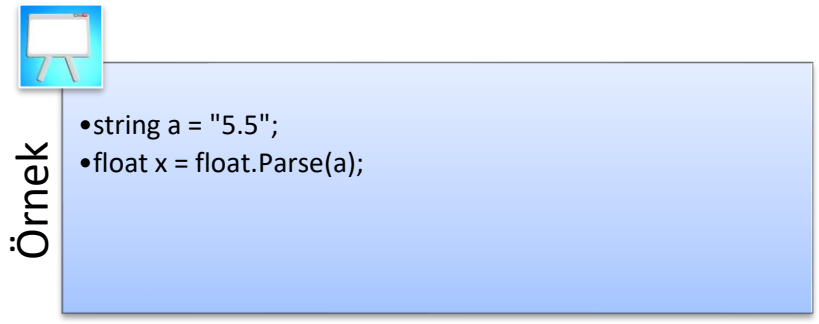
ToString Metodu

ToString metodu ile herhangi bir veri tipi *string* veri tipine dönüştürülebilir. Ekrana basılacak değer *string* veri tipinde olması gerektiği için bu metot sıklıkla kullanılır. Yukarıdaki kodları tekrar incelediğinizde label' a yazdırdığımız tüm değerleri önce *string'* e çevirdiğimizi görebilirsiniz. `degiskenAdi.ToString()` şeklinde yazarak dönüşüm yapabilirsiniz.



Parse Metodu

Parse metodu ile *string* veri tipi herhangi bir veri tipine dönüştürülebilir. `hedefVeriTipi.Parse(degiskenAdi)` şeklinde yazarak dönüşüm yapabilirsiniz.



Parse metodunda dikkat edilmesi gereken bazı hususlar vardır:

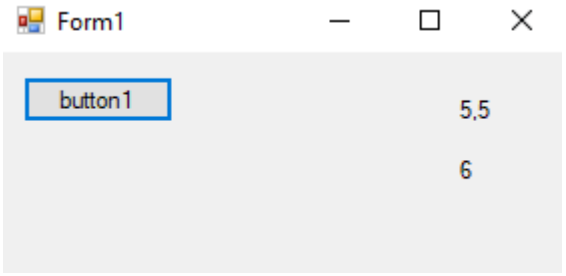
- Dönüştürülecek değişken null olursa,
- *String* veri tipinin haricindeki veri tiplerini dönüştürmek için kullanılırsa,
- Dönüştürülecek *string* değer hedef veri tipinin bellek sınırları içinde olmazsa, derleyici hata verir (Mastar, Vural, & Eriş, 2012).

.Net' te Convert sınıfının dönüşüm metotları anlamlı dönüşümlerde tüm veri tipleri için kullanılabilir.

Convert Metotları

Önceki konularda anlatılan C# dönüştürme metotlarının haricinde .NET sınıf kütüphanesindeki özel metotlar da tip dönüştürme için kullanılır. Convert isimli bu sınıf, dönüşümün anlamlı olacağı tüm veri tiplerinin dönüştürülmesinde kullanılabilir (Algan, 2013). Aşağıdaki kod bloğunda; Convert sınıfının metotlarına örnek verilmiştir. *Float* veri tipinde tanımladığımız "a" değişkeni önce *int* veri tipine dönüştürülmüş ve "b" değişkenine atanmıştır. *Int* veri tipine dönüştürme esnasında veri kaybı olacak ve "b" değişkenine 5 değeri atanacaktır. Daha sonra "b" değişkeni *string* veri tipine dönüştürülerek "c" değişkenine atanacaktır. Butona tıklandığında iki ayrı label öğesine "a" ve "c" değerleri atanacaktır. Butona tıklandığında Şekil 6.7' de görüldüğü üzere ekrana 5,5 ve 5 değerleri yazdırılacaktır.

```
private void button1_Click(object sender, EventArgs e)
{
    float a = 5.5f;
    int b = Convert.ToInt32(a);
    string c = Convert.ToString(b);
    label1.Text = Convert.ToString(a);
    label2.Text = c;
}
```



Şekil 6.7. Program Sonucu

Tablo 6.2' de Convert sınıfının dönüşüm metotları ve açıklamaları listelenmiştir. Metotlar herhangi bir veri tipinde değişken alabilir. Tabloda dikkat edilmesi gereken bazı önemli noktalar bulunmaktadır. Bunlar şu şekildedir;

- *Short* ve *ushort* veri tipleri için Convert.ToInt16(x) ve Convert.ToUInt16(x) dönüşümleri kullanılır.
- *Long* ve *ulong* veri tipleri için Convert.ToInt64(x) ve Convert.ToUInt64(x) dönüşümleri kullanılır.
- *Float* veri tipi için Convert.ToSingle(x) dönüşümü kullanılır.

Tablo 6.2. Dönüşüm Metotları ve Açıklamaları

Metot	Açıklama
Convert.ToBoolean(x)	x nesnesini bool tipine dönüştürür.
Convert.ToByte(x)	x nesnesini 8 bitlik byte tipine dönüştürür.
Convert.ToSbyte(x)	x nesnesini 8 bitlik sbyte tipine dönüştürür.
Convert.ToInt16(x)	x nesnesini 16 bitlik short tipine dönüştürür.
Convert.ToUInt16(x)	x nesnesini 16 bitlik ushort tipine dönüştürür.
Convert.ToInt32(x)	x nesnesini 32 bitlik int tipine dönüştürür.
Convert.ToUInt32(x)	x nesnesini 32 bitlik uint tipine dönüştürür.
Convert.ToInt64(x)	x nesnesini 64 bitlik long tipine dönüştürür.
Convert.ToUInt64(x)	x nesnesini 64 bitlik ulong tipine dönüştürür.
Convert.ToSingle(x)	x nesnesini float tipine dönüştürür.
Convert.ToDouble(x)	x nesnesini double tipine dönüştürür.
Convert.ToDecimal(x)	x nesnesini decimal tipine dönüştürür.
Convert.ToChar(x)	x nesnesini char tipine dönüştürür.
Convert.ToString(x)	x nesnesini string tipine dönüştürür.
Convert. ToDateTime(x)	x nesnesini date time tipine dönüştürür.

Bireysel Etkinlik

- Siz de kendi Visual Studio Programınızda ünite içinde yazdığımız kodları ayrı ayrı deneyerek çıktılarını karşılaştırınız.
- Ayrıca örneklerini yapmadığımız metotlar için de yeni örnekler deneyiniz.

Özet

- **Tip Dönüşümü Nedir?**
- Her veri tipinin kendi içinde yapılabilecek birçok işlem vardır. Aynı veri tipleriyle işlem yaptığımız gibi bazı durumlarda farklı veri tipleriyle de işlem yapmak isteyebiliriz. Normalde farklı veri tipleri arasında işlem yapılmazken işlem yapılması gereken durumlarda veri tipinin dönüştürülmesi gerekmektedir.
- Tip dönüşümleri, yapılma şekline göre ikiye ayrılır. Bunlar; bilinçli tip dönüşümü ve bilinçsiz tip dönüşümüdür. Bazı kaynaklarda bilinçli dönüşüm için açık dönüşüm, bilinçsiz dönüşüm için de kapalı dönüşüm ifadesini görebilirsiniz. Aynı manaya gelirler.
- **Bilinçsiz(Implicit) Tip Dönüşümü**
- Herhangi bir operatör kullanmadan derleyicinin kendisinin yaptığı dönüşüme bilinçsiz dönüşüm denir. Bilinçsiz dönüşümde küçük bellekli bir veri tipinden büyük bellekli bir veri tipine dönüşüm yapılabilirken büyük bellekli veri tipinden küçük bellekli veri tipine dönüşüm yapılamaz. Bu durumda hata alırız.
- Veri tiplerinin hepsinin birbirine dönüşümü mümkün değildir. Büyük veri tipleri küçük veri tiplerine dönüştürülürken veri kaybı olacağından derleyici hata verir. Küçük veri tipleri büyük veri tiplerine dönüştürülürken; büyük tipten dolayı eklenen fazla bitler sıfır ile doldurulur. Bu durum değişken içinde bulunan veriyi etkilemeyeceği için veri kaybı olmaz.
- **Bilinçli(Explicit) Tip Dönüşümü**
- Bilinçsiz dönüşümde veri tipleri uygun veri tiplerine derleyici tarafından dönüştürülebiliyordu. Ama her veri tipi her veri tipine dönüştürülemiyordu. Büyük veri tiplerini küçük veri tiplerine dönüştürmeye çalıştığımız zaman hata ile karşılaşıyorduk. Bazı durumlarda büyük veri tiplerini küçük veri tiplerine dönüştürmek isteyebiliriz. O zaman yapmamız gereken bir dönüştürme operatörü kullanmaktır. Bilinçli tip dönüşümünde, kendisine dönüştürülmesi amaçlanan veri tipi parantez içine alınır ve yanına değişken ya da sabit değer yazılır. Bu durumda veri kaybını göze almış ve verimizi istediğimiz türe dönüştürebilmiş olacağız. Tabi bu ancak sayısal değerler içinde yapılabilir.
- Küçük veri tipindeki veriyi büyük veri tipine bilinçli dönüştürmenin, bilinçsiz dönüştürmeden bir farkı yoktur. Bir dönüşüm operatörü kullansak da kullanmasak da aynı sonucu verecektir.
- **Dönüşüm Metotları**
- ToString() Metodu: ToString metodu ile herhangi bir veri tipi string veri tipine dönüştürülebilir. Ekrana basılacak değer string veri tipinde olması gerektiği için bu metot çokça kullanılır. Yukarıdaki kodları tekrar incelediğinizde label' a yazdırdığımız tüm değerleri önce string' e çevirdiğimizi görebilirsiniz. degiskenAdi.ToString() şeklinde yazarak dönüşüm yapabilirsiniz.
- Parse Metodu: Parse metodu ile string veri tipi herhangi bir veri tipine dönüştürebilir. hedefVeriTipi.Parse(degiskenAdi) şeklinde yazarak dönüşüm yapabilirsiniz.
- Convert Metotları: Önceki konularda anlatılan C# dönüştürme metotlarının haricinde .NET sınıf kütüphanesindeki özel metotlar da tip dönüştürme için kullanılır. Convert isimli bu sınıf, dönüşümün anlamlı olacağı tüm veri tiplerinin dönüştürülmesinde kullanılabilir.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi tip dönüşümü hakkında söylenemez?
   a) Farklı veri tipleri arasında işlem yapılabilmesini sağlar.
   b) Bir değişkenin dönüşümü yapılırken bellekteki yeri değişir.
   c) Tip dönüşümü sadece küçük veri tipinden büyük veri tipine doğru yapılır.
   d) Tip dönüşümleri bilinçli ve bilinçsiz dönüşüm diye ikiye ayrılır.
   e) Büyük veri tipinden küçük veri tipine dönüşümde veri kaybı olabilir.

2. Herhangi bir operatör kullanmadan derleyicinin kendisinin yaptığı dönüşüme ne denir?
   a) Bilinçsiz Tip Dönüşümü
   b) Bilinçli Tip Dönüşümü
   c) Açık Tip Dönüşümü
   d) Yarı Tip Dönüşümü
   e) Tam Tip Dönüşümü

3. Uint veri tipi aşağıdaki veri tiplerinden hangisine bilinçsiz dönüştürülemez?
   a) ulong
   b) long
   c) float
   d) short
   e) double

```
float a=10.5f;
int x=(int)a;
```

4. Yukarıda verilen kod bloğuna göre x aşağıdakilerden hangisine eşit olur?
   a) 10.5
   b) “10,5”
   c) 10.5f
   d) “10”
   e) 10

```
float a = 5.6f;
string b= "5";
int c = 6;
```

5. Yukarıdaki değişken ataması göz önüne alındığında aşağıdakilerden hangisinin sonucu bir int değer değildir?
   a) (int)a
   b) Convert.ToInt32(a)
   c) int.Parse(b)
   d) c
   e) c.ToString()

6. Convert sınıfının veri tipi dönüşüm metotları ve eşleştirmelerinden hangisi yanlıştır?
   a) Convert.ToInt16(x) - short veri tipine dönüşüm
   b) Convert.ToDouble(x) –decimal veri tipine dönüşüm
   c) Convert. ToDateTime(x) - date time veri tipine dönüşüm
   d) Convert.ToSingle(x) - float veri tipine dönüşüm
   e) Convert.ToUInt64(x) – ulong veri tipine dönüşüm

```
byte a = 2;
short b = -3;
uint c = 5;
float x;
x = a + b + c;
```

7. Yukarıda verilen kod bloğuna göre x aşağıdakilerden hangisine eşit olur?
   a) 10
   b) “4”
   c) x hesaplanamaz
   d) 4
   e) “10”

8. Aşağıda verilen dönüştürülen veri tipi– hedef veri tipi eşleştirmelerinden hangisi bilinçsiz tip dönüşümü için yanlıştır?
   a) byte - ushort
   b) sbyte - int
   c) long - ulong
   d) ulong - float
   e) long - double

9. Parse metodu için aşağıda verilen bilgilerden hangisi yanlıştır?
   a) Söz dizimi hedefVeriTipi.Parse(degiskenAdi) şeklindedir.
   b) int veri tipini dönüştürebilir.
   c) Dönüştürülecek değer null olamaz.
   d) Sadece string veri tipini dönüştürür.
   e) Dönüştürülecek değer, hedef veri tipinin bellek sınırları içinde olmalıdır.

10. Tip dönüşümü ile ilgili aşağıda verilen ifadelerden hangisi doğrudur?
    a) Büyük bellekli veri tipinden küçük bellekli veri tipine bilinçsiz dönüşüm yapılabilir.
    b) Herhangi bir operatör kullanmadan derleyicinin kendisinin yaptığı dönüşüme bilinçsiz dönüşüm denir.
    c) Veri tiplerinin hepsinin birbirine bilinçsiz dönüşümü mümkündür.
    d) Bilinçli dönüşümde veri kaybı asla yaşanmaz.
    e) Bilinçli dönüşüm yaparken operatör kullanmaya gerek yoktur.

Cevap Anahtarı
1.c, 2.a, 3.d, 4.e, 5.e, 6.b, 7.d, 8.c, 9.b, 10.b

YARARLANILAN KAYNAKLAR

Algan, S. (2013). *Her Yönüyle C#.* İstanbul: Pusula.

Mastar, M., Vural, M. T., & Eriş, S. (2012). *Visual Studio 2011.* Kodlab.