**T.C.**
**MİLLÎ EĞİTİM BAKANLIĞI**

# BİLİŞİM TEKNOLOJİLERİ

# ETKİLEŞİMLİ ANİMASYONLAR
**482BK0140**

**Ankara, 2012**

- 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

| | |
|---|---|
| **KOD** | **482BK0140** |
| **ALAN** | **Bilişim Teknolojileri** |
| **DAL/MESLEK** | **Web Programcılığı** |
| **MODÜLÜN ADI** | **Etkileşimli Animasyonlar** |
| **MODÜLÜN TANIMI** | Bu modül, animasyon yazılımı kullanarak etkileşimli animasyonlar hazırlamak için temel bilgi ve becerilerin kazandırıldığı bir öğrenme materyalidir. |
| **SÜRE** | 40/32 |
| **ÖNKOŞUL** | "Programlama Temelleri" dersi modülleri ve web ortamı için animasyon hazırlama" modülünü tamamlamış olmak |
| **YETERLİK** | Web sayfaları için animasyonlar hazırlamak |
| **MODÜLÜN AMACI** | **Genel Amaç**<br>Bu modülle gerekli ortam sağlandığında; eylem kodlarını kullanarak etkileşimli animasyonlar hazırlayabileceksiniz.<br>**Amaçlar**<br>**1.** Temel programlama işlemlerini yapabileceksiniz.<br>**2.** Fonksiyon işlemlerini gerçekleştirebileceksiniz.<br>**3.** Olaylarla (event) ilgili düzenlemeleri yapabileceksiniz.<br>**4.** Sınıf(class) işlemlerini gerçekleştirebileceksiniz.<br>**5.** Karar yapılarını kullanabileceksiniz. |
| **EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI** | **Ortam:** Bilişim Teknolojileri laboratuvarı, işletme ortamı<br>**Donanım:** Bilgisayar |
| **ÖLÇME VE DEĞERLENDİRME** | Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış testi, boşluk doldurma, eşleştirme vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir. |

# GİRİŞ

**Sevgili Öğrenci,**

İnternet üzerinde çalışan siteler, oyunlar, birçok özel uygulamalar animasyon yazılımıyla yapılmaktadır. Animasyon yazılımı, programlama yetenekleriyle bir çok teknolojiye entegre olabilir. Örneğin, cep telefonları için programlar yazarken tablet bilgisayarlar için oyunlar geliştirebilirsiniz. Animasyon yazılımı, diğer programlama dillerinden farklı olarak animasyon ve programlamayı bir araya getirir. Bu sebeple her geçen gün kullanıcı sayısı artmaktadır.

Ürettiğiniz uygulamalar, siteler, oyunlar, simülasyonlar çok az yer kaplayacakları için internette yayınlayabilir, kolayca insanlarla paylaşabilirsiniz.

Bu modülle animasyon yazılımında değişken tanımlayabilecek değişkenlerle işlemler yapabilecek, fonksiyon üretip sahnede olan herşeye hükmedebileceksiniz. Kodlarla nesnelerin biçimlerini değiştiren ve hareket ettiren animasyonlar oluşturup nesneye yönelik programlama yapabileceksiniz. Yazdığınız kodlara, karar yapıları ekleyip şartlara göre değişen kod yapıları oluşturabileceksiniz.

# ÖĞRENME FAALİYETİ–1

## AMAÇ

Bu faaliyet sonunda temel programlama işlemlerini yapabileceksiniz.

## ARAŞTIRMA

- Web sayfalarında kullanılan etkileşimli animasyonları araştırın ve sınıfta paylaşınız.
- Etkileşimli animasyon ile animasyon arasındaki fark nedir?

# 1. ACTIONSCRIPT TEMELLERİ

## 1.1. ActionScript

1995 yılında ilk olarak ortaya çıkan animasyon yazılımında kullanılan komutlar temel animasyon ve yönlendirme komutları olarak (toplamda 12 temel komut) kullanıcıya sunulmuştur. Her yeni versiyonla bu komutlara, yeni komut setleri eklenmiş ve her versiyonda farklı ihtiyaçlara karşılık vermiştir. Animasyon yazılımının 5. versiyonunda tam olarak bir uygulama diline dönüşmüş ve ActionScript ismini almıştır.

Animasyon yazılımı geliştikçe ActionScript dili de gelişmiş, ActionScript 2.0 ve ActionScript 3.0 versiyonuna ulaşmıştır. ActionScript dili ECMAscript (İnternette istemci üzerinde çalışan programlar için geliştirilmiş dil standardı) üzerine inşa edilmiştir, ECMAscript, JavaScript'in esasını oluşturduğu için birçok geliştirici için ActionScript'i anlamak kolaydır. Programlama özellikleriyle animasyon yazılımı üzerinde her türlü kontrol sağlanmış internet siteleri, internet siteleri için uygulamalar, oyunlar ve buna benzer birçok uygulama geliştirilebilir.

ActionScript ile zaman çizgisini kontrol edebilirsiniz. Yaptığınız animasyonlara etkileşim veya çeşitli şartlar ekleyebilir, animasyonun istediğiniz şekilde akmasını sağlayabilirsiniz. Sahneye düğmeler, yazı kutuları vb. form bileşenleri ekleyip kullanıcı ile etkileşime girebilirsiniz. Yapılabilecekleriniz hayal gücünüzle sınırlıdır.

ActionScript 3.0 diliyle ActionScript yeni bir boyuta taşınmış ve güçlü bir programlama dili olmuştur. ActionScript 3.0 nesne tabanlı (OOP) bir programlama dilidir. Eski tarz prosedürel kodlama olanağı da bulunmaktadır. Bu sayede isteyen OOP programlama yapabildiği gibi isteyenler prosedürel programlama da yapabilmektedir. Esnek bir dildir. İsteyen herkes kendi sınıfını oluşturup animasyon yazılımı için de kullanabilir. Sunucu tarafında çalışan çeşitli internet programlama dilleri (php,asp,asp.net,java vb.) ile uyumlu çalışabilmektedir.

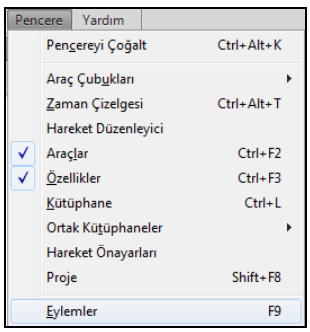
## 1.2. Eylemler Paneli

Eylemler paneli, animasyon yazılımı içerisinde Script yazmak için geliştirilmiş editördür. ActionScript 3.0 ile kod tamamlama özelliği daha da gelişmiştir. Programlama yapıldığı esnada kodları tamamlamak için çeşitli seçenekler çıkar ve programcıya kod yazarken büyük kolaylıklar sağlar.

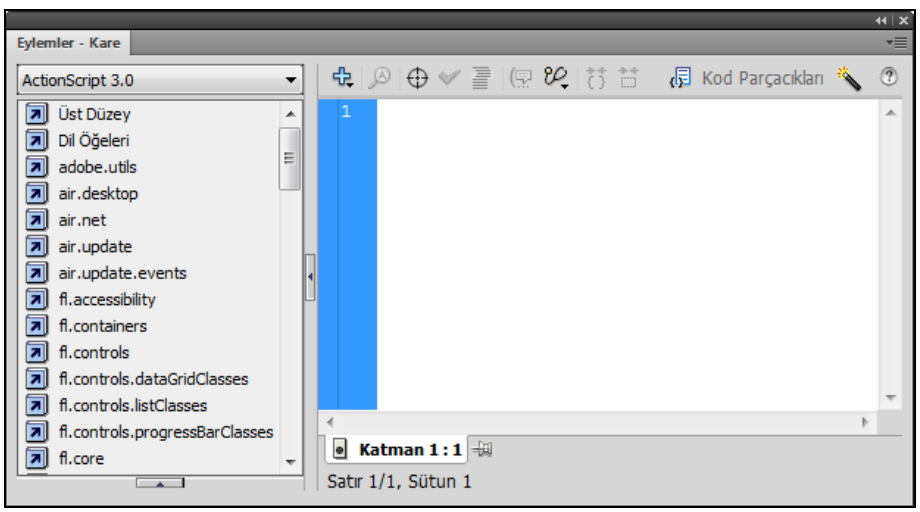
ActionScript'in eski versiyonlarında sahnemiz üzerine, film klipleri ve düğmelere kod yazabilirrken eylemler panelini 3.0 versiyonuyla birlikte, bu nesneler üzerinde açmaya çalışırsak hata verdiği görülecektir. Artık ActionScript kodları sadece sahneye yazılabilir.

Eylem panelini açmak için:

- Pencere > Eylemler (F9) komutunu kullanabilirsiniz.




**Resim 1.1: Pencere menüsü**



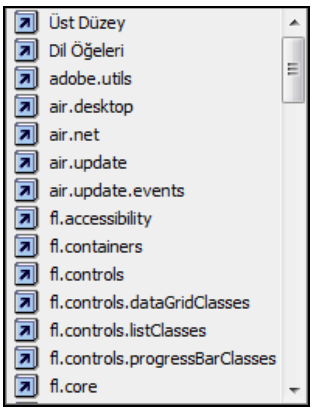

**Resim 1.2: Eylemler paneli**

Eylemler paneli 3 bölümden oluşmaktadır.

### 1.2.1. Panel Bölümleri

- **Script elemanları**

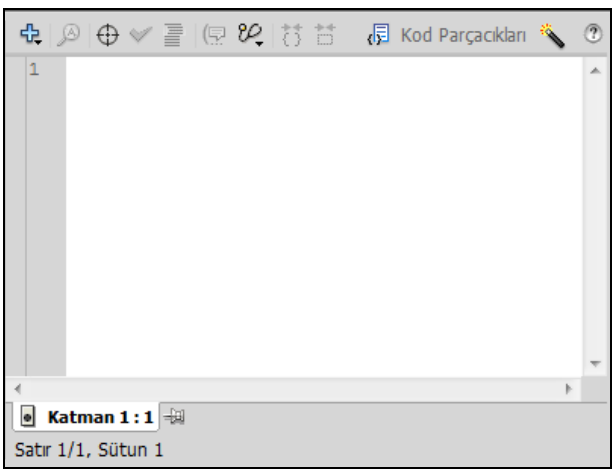
Bu bölümde,tüm ActionScript sınıfları gruplandırılmış olarak bulunur. İstediğimiz komut seçilerek yazma alanına eklenebilir. Uygulama içerisinde kullanılabilen tüm özellik ve metotlar bu pencere altında saklanır.




**Resim 1.3: Script elemanları menüsü**

- **Script Ekleme - Düzenleme Alanı**

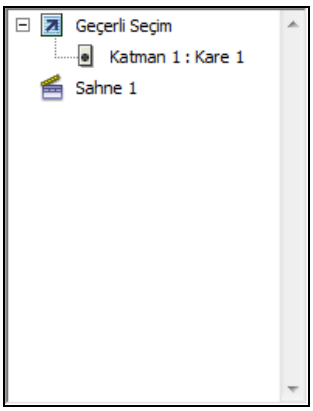
Kodların yazıldığı ve daha önceden yazılmış kodların düzenlenebildiği bölümdür.




**Resim 1.4: Script ekleme- düzenleme alanı**

- **Script gezinme alanı**

Karmaşık animasyon ve uygulamalarda kod yazılan yerler arasında hızlı geçiş yapmak için ActionScript 2.0 ile geliştirilmiş ve halen kullanılan özelliktir. Kod yazılan tüm nesneler bu bölümde listelenir.




**Resim 1.5: Script gezinme alanı**

## 1.2.2. Panel Araçları

Eylemler panelinde kod yazarken kullanımı kolaylaştıran doğruluk denetimi yapan araçlar bulunmaktadır.


**Resim 1.6: Panel araçları**

➢ **Otomatik düzenleme**

Otomatik düzenleme çalıştırıldığında kod penceresinde yazılmış kodlar standarda göre yeniden düzenlenir, kurallara uygun girinti verilir, satır sonunda kullanılmayan noktalı virgülleri (;) gerekli yerlerde otomatik olarak yerleştirir.

Script'i kullanmak için:

- Panel içerisine aşağıdaki kodları ekleyin.

```
1  import flash.events.MouseEvent;
2  stage.addEventListener(Event.ENTER_FRAME,gez);
3  var i:uint = 0;
4  function gez(event:Event):void
5  {
6  if (mouseX>200)
7      {
8  nesnem.x -=  1;
9  }
10 if (mouseX<200)
11 {
12 nesnem.x +=  1;
13     }
14     if (mouseY<200)
15 {
16 nesnem.y +=  1;
17 }
18 if (mouseY>200)
19     {
20 nesnem.y -=  1;
21 }
22 }
```

**Resim 1.7: Düzensiz yazılmış kodlar**

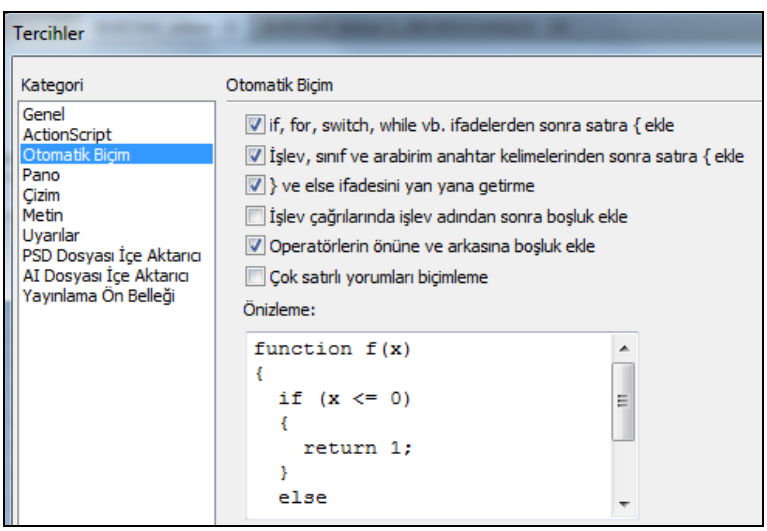
- CTRL+Shift+F komutunu ya da  simgesini kullanın.
- Uygulama otomatik olarak ActionScript söz dizilimine göre düzenleyecektir.

```
1  import flash.events.MouseEvent;
2  stage.addEventListener(Event.ENTER_FRAME,gez);
3  var i:uint = 0;
4  function gez(event:Event):void
5  {
6      if (mouseX>200)
7      {
8          nesnem.x -=  1;
9      }
10     if (mouseX<200)
11     {
12         nesnem.x +=  1;
13     }
14     if (mouseY<200)
15     {
16         nesnem.y +=  1;
17     }
18     if (mouseY>200)
19     {
20         nesnem.y -=  1;
21     }
22 }
```

**Resim 1.8: Otomatik düzenleme yapılmış kodlar**

Kendi otomatik düzeltme ayarlarınızı yapmak için:

- Panelin sağ tarafındaki  menüye tıklayarak tercihler paneli açın.




**Resim 1.9: Tercihler paneli**

- İstenilen biçim ayarlandığında onaylanarak paneli kapatın.

➢ **Satır numaralarini gösterme**

Karmaşık kod yapılarında hata bulmak için çok işe yarayan bir özelliktir.

Satır numaraları göstermek için:

- CTRL+Shift+L komutunu kullanabilir ya da panel özelliklerinden bu özelliği açıp kapatabilirsiniz.

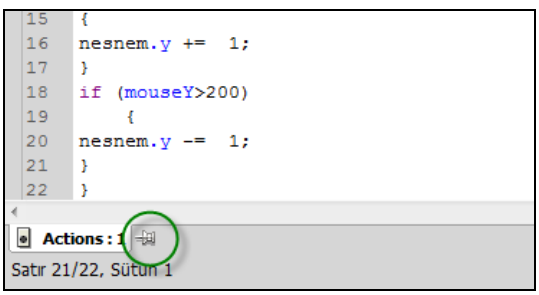

**Resim 1.10: Özellik menüsü**

➢ **Satır kaydırma**

Bu özellik aktifken yazılan kod, kod penceresine sığmadığında alt satıra geçer. Satır kaydırma işlemi yapılmayacağı için hızlı şekilde hatasız kodlar yazılabilir.

➢ **Script iğneleme**

Birden fazla frame ile uğraşırken zaman çizgisi üzerinde frameleri seçip kod yazmaya çalışmak bazen çok sıkıcı bir hal alabilir, bu gibi durumlarda bu özellik kullanılarak kod yazılan frameler panel altında bir tab haline getirilir. Frame seçmek yerine tablar arasında geçiş yapmak daha kolay olacaktır.



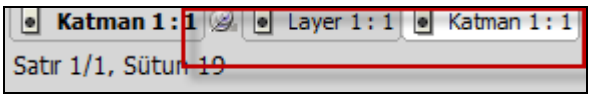

**Resim 1.11: Script iğneleme**

Script iğneleme yapmak için:

- Kod yazdığımız ilk frame seçiliyken eylemler paneli açılır. Script iğnele düğmesine basın.


**Resim 1.12: Script iğneleme ikonu tıklanmış hali**

- Diğer kod yazdığımız frame seçin ve script iğnele düğmesine tekrar tıklayın.



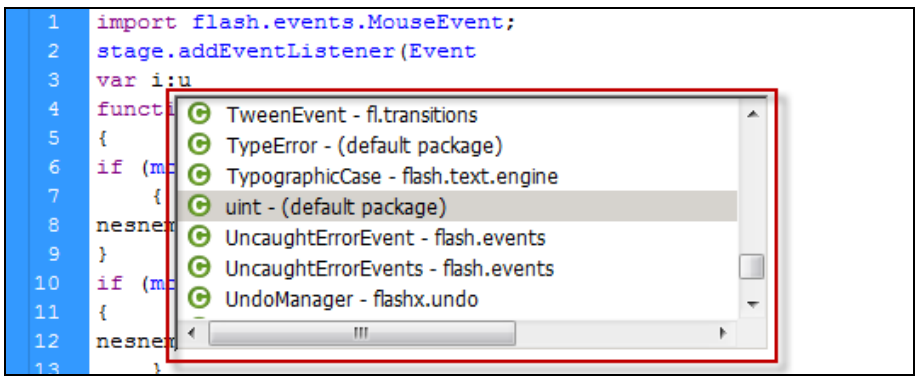

**Resim 1.13: Diğer script alanının iğlenemiş hali**

➢ **Kod ipuçları**

Yazılan kodlar için ipucu kutusunu çalıştırır. Kod ile ilgili parametreler bu kutu içerisinde gösterilir.

Kod ipuçlarını göstermek için:

- Eylemler panelini açın.
- *var i:* yazın.
- Açılan pencereden uint veri türü seçilir.




**Resim 1.14: Kod ipuçları**

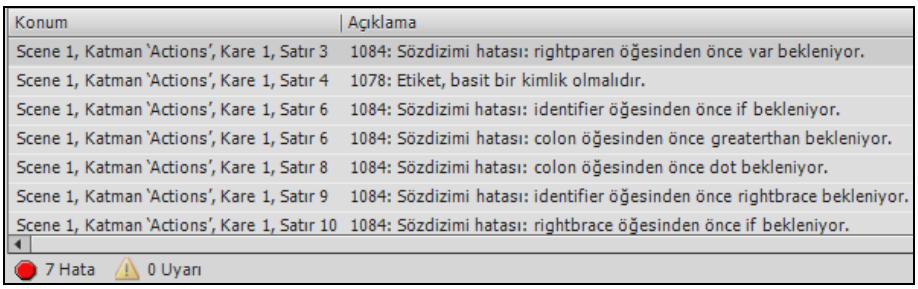
- Esc tuşu ile bu pencereyi yok edebilir CTRL+Boşluk ile yeniden açabilirsiniz.

➢ **Hataları denetle**

Yazılan kodların editör aracılığı ile kontrol ettirmek istenildiğinde kullanılan bu özellik, kontrol sonucunu iletişim kutusunda detaylı olarak sunacaktır.

Hataları denetleyi çalıştırmak için:

- CTRL+T komutunu ya da  simgesini kullanın.




**Resim 1.15: Hata raporu**

## 1.3. Yazım kuralları

### 1.3.1. Açıklamalar

Karmaşık ve yoğun kodlamanın olduğu projelerde, kod parçalarının ne iş yaptığını hatırlamak ya da takım çalışması yaparken takım üyelerinin kod blokları, değişkenler vb. bilgilendirmek için kullanılır. Kodların işleyişini etkilemez.

Açıklama ifadesi eklemek için:

- Eylemler paneli açılır ve ( // ) ifadesi yazılır. Buradan sonra yazılanlar tüm satırı açıklama satırı haline getirir. Satır bittiğinde açıklamada bitmiş olur.

- /* …….. */ ifadesi yazıldığında, arasına yazılan her şey açıklama olarak alınır.

```
1  // ifadesi yazıldığı satırı açıklama satırı haline getirir.
2  Satır bittiğinde açıklamada bitmiş olur.
3
4  /*ifadesi arasına yazılan
5
6  her şey açıklama olarak alınır.*/
```

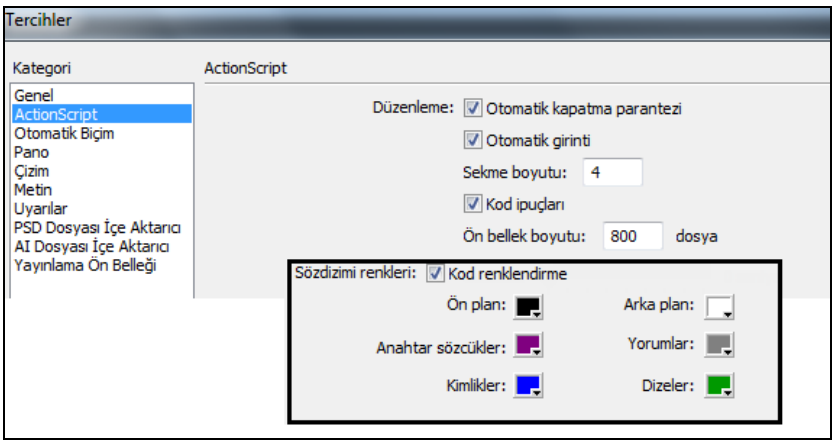
**Resim 1.16: Açıklama ifadeleri**

Açıklama satırları, program yazarken bazı kodları iptal etmek ve yeniden yazmaktan kurtulmak için sıkça kullanılır. Açıklama satırı olarak eklenen satır, panel renklendirmesi değiştirilmedikçe gri ve silik olarak yazılacaktır.

Panel renklerini değiştirmek için:

- Düzen > Tercihler (CTRL+U) ya da "Eylemler" panelinden köşedeki Tercihler simgesini tıklayın.

- Tercihler iletişim penceresinden, ActionScript kategorisi seçildiğinde sağ taraftaki kod renklendirme bölümünden isteğinize uygun ayarı yapın.




**Resim 1.17: Kodlama renkleri**

### 1.3.2. Bloklar

{ } parantezleri (küme parantezi, süslü parantez, güzel parantez) arasında yazılan her kod birlikte çalışmak için bloklanır.

```
1  var a:uint=0;
2  while(a<=10)
3  {
4  trace("merhaba");
5  a++;
6  }
```

Yukarıdaki döngüde güzel parantezler arasındaki kod bloğu her döngü basamağı için birlikte çalışacaktır.

### 1.3.3. Nokta(.)

Nesnelerin yol tanımlarını yapabilmek ya da sınıfların özellik ve metotlarına ulaşmak için kullanılan işarettir.

```
1  /* Mc sınıfının x özelliğine (koordinat) ulaşmak için nokta
2  işareti kullanılmalıdır.*/
3  mc.x = 100;
```

### 1.3.4. Noktalı Virgül (;)

Yazılan kodlarda ifadenin bittiğini göstermek amacıyla ifade sonuna konulur.

```
1  trace("Volkan GÖK");
```

Satır sonundaki (;) ifadesi trace komutunun sonlandığını göstermektedir. Otomatik düzelt seçeneği çalıştırıldığında otomatik olarak ifadeyi yerleştirecektir. Noktalı virgül yazılmasa da program hata mesajı vermez, fakat okunuşu kolaylaştırmak ve düzeni sağlamak için yazılması gereklidir.

## 1.4. ActionScript ile İletişim

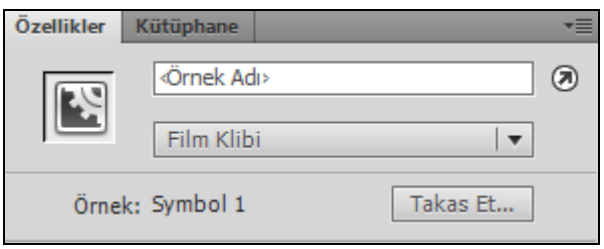
### 1.4.1. Film Klipleri ile Çalışma

Animasyon yazılımında en fazla kullanılan film klipleri, ActionScript içinde olmazsa olmaz bileşenlerdir. Film Klipleri butonların, grafiklerin yaptığı her işi yapabilirler. Kendi başlarına bir animasyon çalışma dosyası gibidirler; yani her film klibinin kendi zaman çizgisi bulunmaktadır. Oluşturulan film klibi animasyon yazılımının kütüphanesine eklenir. Buradan istenildiği kadar çoğaltılıp sahneye eklenebilir.

Film kliplerini ActionScript içerisinde kullanmak için sahnedeki yerini ve film klibinin ismini bilmek yeterlidir. Sahnede onlarca film klibi ve bu klipler kütüphaneden aynı film klibinden kopyalanmış olabilir. Kod yazarken oluşabilecek karmaşayı önlemek için film kliplerinin kütüphane isimlerinin haricinde sahnedeki her nesneye bir örnek ismi (instance name) atamak gereklidir. Özellikler panelinden örnek ismi oluşturulabilir.

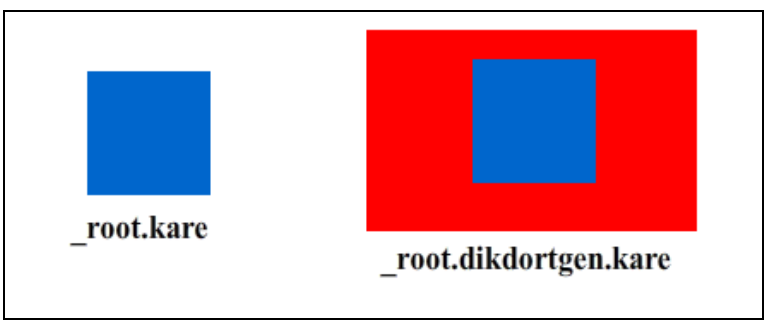
Örnek ismi oluşturmak için:

- Örnek ismi ekleyeceğimiz nesnemiz sahneden seçin.
- Özellikler panelindeki örnek adı bölümüne değişken isimlendirme kurallarına uygun olarak bir örnek ismi verin.




**Resim 1.18: Örnek adı (instance name).**

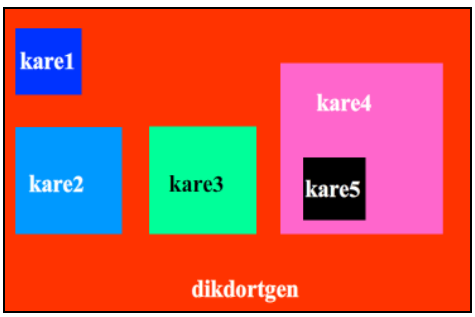
Film klibinin ismi dışında sahnede bulunduğu yer de önemlidir. Örneğin, ana sahne üzerinde bulunan bir film klibiyle başka bir film klibinin içinde bulunan bir film klibi için yazılacak kodlar farklı olacaktır.




**Resim 1.19: Adres tanımlama**

Adresleme kurallarını pekiştirmek için aşağıdaki örneği yapınız.

- Yeni bir çalışma sahnesi açın.
- Sahneye dikdörtgen çizip film klibi haline getirin örnek adını dikdörtgen yapın.
- Dikdörtgen klibine çift tıklayarak içerisine girin ve kare1, kare2, kare3, kare4 örnek isimlerine sahip film kliplerini oluşturun.
- Kare4 film klibinin içerisine girip kare5 örnek ismine sahip şekli oluşturun.
- Sonuç olarak aşağıdaki şekli elde etmelisiniz.




**Resim 1.20:Nesne konumları**

- Ana sahneye dönüp eylemler panelini açın.
  - Dortgen film klibinin içindeki kare4 klibinin içindeki kare5 klibini ana sahne üzerinden 90 derece döndürmek için aşağıdaki komutunu kullanın.

```
1  dortgen.kare4.kare5.rotation=90;
```

  - Bu komut ana çalışma sahnesine yazıldığında geçerli olacaktır. Dortgen isimli film klibinin içerisine yazılmak istenilirse dortgen yerine **this** komutu ile başlamalıdır.

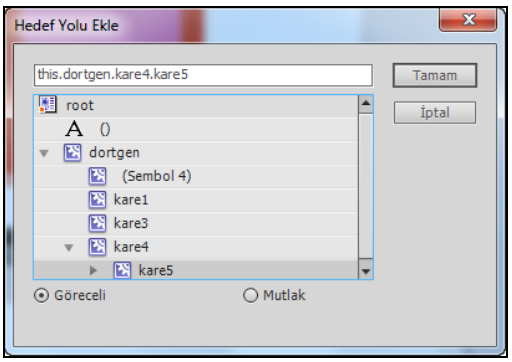
```
1  this.kare4.kare5.rotation=90;
```

  - Aynı işi kare5 film klibi içerisinde sahneye kod yazıp yapmak istersek

```
1  this.rotation=90;
```

şeklinde yazılıp çalıştırılabilir.

- Hedef yol ekle  simgesi ile sahnedeki konumunuza göre nesneleri seçip yolu kolay bir şekilde panel üzerine ekleyebilirsiniz.




**Resim 1.21: Hedef yol ekle.**

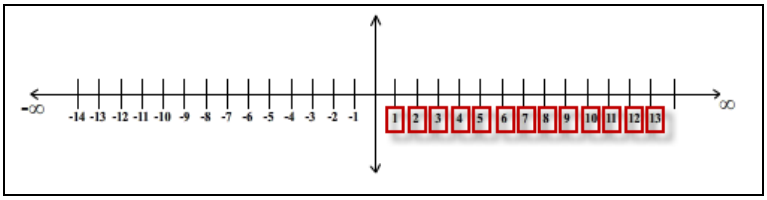
## 1.4.2. Veri Türleri

ActionScript 3.0 içerisinde kullanılan veri türleri basit ve ileri düzey veri türleri olarak ikiye ayrılır. Uygulama içerisinde hangi tür verilerle çalışılacaksa ona uygun veri türü seçilmesi gerekmektedir. Karışık uygulamalarda belleği tasarruflu kullanmak ve uygulamanın performanslı çalışmasını sağlamak için veri türlerini uygun seçmek önemlidir.

### 1.4.2.1. Basit düzey veri türleri

- **Pozitif tam sayılar (Uint)**

0 ile 4.294.967.295 arasındaki tam sayıları tutabilen veri türüdür.




**Resim 1.22: Uint sayı doğrusu**

Veri türünün alabileceği maksimum ve minimum değerleri görmek için:

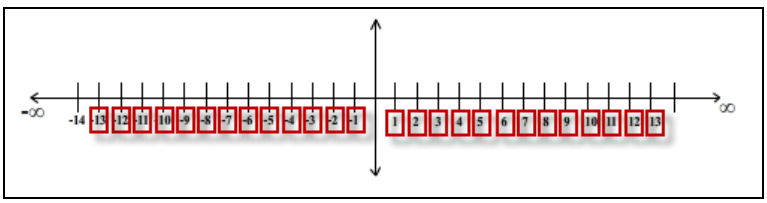
- Yeni bir çalışma sahnesi açınız.
- Eylem paneli açılarak aşağıdaki kodlar ekleyiniz.

```
1  trace(uint.MIN_VALUE);
2  trace(uint.MAX_VALUE);
```

- CTRL+enter tuşlarına basıp output ekranında 0 ve 4.294.967.295 sayılarını görebilirsiniz.

➢ **Tam sayılar (int)**

-2.147.483.648 ile 2.147.483.647 sayıları arasında tam sayı değerleri alabilen veri türüdür. Uintin aksine negatif değerler de alabilir.



**Resim 1.23: İnt sayı doğrusu**

Veri türünün alabileceği maksimum ve minimum değerleri görmek için:

- Yeni bir çalışma sahnesi açınız.
- Eylem paneli açılarak aşağıdaki kodlar ekleyiniz.

```
1  trace(int.MIN_VALUE);
2  trace(int.MAX_VALUE);
```

- CTRL+enter tuşlarına basıp output ekranında -2.147.483.648 - 2.147.483.647 değerlerini görebilirsiniz.

➢ **Tüm sayılar (Number)**

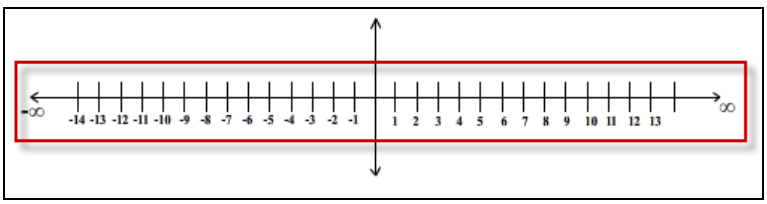
4.9406564584124654e-324 ile 1.79769313486231e+308 sayıları arasındaki tüm sayılar için kullanılan veri türüdür

Veri türünün alabileceği Maksimum ve minimum değerleri görmek için:

- Yeni bir çalışma sahnesi açınız.
- Eylem paneli açılarak aşağıdaki kodlar ekleyiniz.

```
1  trace(Number.MIN_VALUE);
2  trace(Number.MAX_VALUE);
```

- CTRL+enter tuşlarına basıp output ekranında 4.9406564584124654e-324 ile 1.79769313486231e+308 değerlerini görebilirsiniz.



**Resim 1.24: Number sayı doğrusu**

Görüldüğü üzere Number veri türü her iki sayısal veri türünü de kapsamaktadır. Öyleyse neden diğer veri türleri var sorusu akla gelebilir. En başta bahsedildiği gibi performans ve tasarruf sebeplerinden dolayı uint ya da int kullanabilecekken Number kullanmamak akıllıca olacaktır.

Günlük hayattan örnek verecek olunursa küçücük bir odada dev ekran bir televizyon ya da kocaman bir salonda 37 ekran bir televizyon kullanmak ne kadar doğrudur düşünmek gerekir. Yaptıkları iş aynı olsa da bulundukları odaya göre yanlış seçim olacaklardır.

Oluşturulacak 1000-2000 kayıtlık basit bir adres defterinde veri türü önemli olmazken büyük veri grupları ile uğraşan, örneğin banka uygulamaları vb. uygulamalarda veri türü hayati öneme sahip olacaktır.

- **Alfanümerik veri türü (String)**

İçerisinde harf, kelime, sayı, özel karakterler kısacası klavyeden basılan her tuşu barındırabilen veri türüdür. Yazılan her şeyi içerisine veri olarak alabilir. Sayısal değerleri veri olarak aldığında bu değerler ile matematiksel işlemler yapılamaz.

- **Mantıksal veri türü (Boolean)**

Doğru (true) ve yanlış (false) değerler alabilen veri türüdür. Genellikle karşılaştırmalarda kullanılır. Bu veri türünde tanımlanmış bir değişken doğrudan true ya da false değeri verilebileceği gibi karşılaştırma sonucu true ya da false değerini alabilir.

- **Nesne veri türü (object)**

Nesne kavramı ve nesneler hakkında gerekenler sınıflar konusunda ayrıntılı işlenecektir. Temel olarak nesnelere metotlar tanımlanabilir ve veriler yüklenebilir.

- **Void veri tipi**

Fonksiyon için kullanılan veri tipidir. (Fonksiyonlar daha sonra detaylı işlenecektir). Sadece fonksiyonlarda kullanılır. Fonksiyonun dışarıya veri döndürmediği zamanlarda kullanılır. Fonksiyonda return kullanılacaksa void kullanılmaz.

#### 1.4.2.2. İleri Düzey Veri Türleri

İleri düzey olarak tanımlanabilecek veri türleri: XML, XMLList, Array, date, error, function, RegExp şeklindedir.

### 1.4.3. Operatörler

- **Toplama operatörü (+)**

Bu operatör sayısal ifadeleri matematiksel olarak toplar. String ifadeleri yan yana yazdırır.

Aşağıdaki örnekte sayı türündeki iki ifade matematiksel olarak toplanmıştır.

```
1  var  degisken1:uint=5;
2  var  degisken2:uint=10;
3  trace(degisken1+degisken2); //işlemin sonucu 15 olarak görünür
```

Burada ise ifadeler string olduğundan dolayı yan yana yazılmıştır.

```
1  var  degisken1:String="5";
2  var  degisken2:String="10";
3  trace(degisken1+degisken2); //işlemin sonucu 15 olarak görünür
```

- **Fark operatörü (-)**

Sadece matematiksel ifadelerde kullanılır. Çıkarma işlemi yapılır.

```
1  var  degisken1:uint=50;
2  var  degisken2:uint=10;
3  trace(degisken1-degisken2); //işlemin sonucu 40 olarak görünür
```

- **Çarpma operatörü(*)**

Matematiksel çarpma işlemi yapılır.

```
1  var degisken1:uint=2;
2  var degisken2:uint=3;
3  trace(degisken1*degisken2);//işlemin sonucu 6 olarak görülür.
```

- **Bölme operatörü (/)**

Bölme işlemi yapılır.

```
1  var degisken1:uint=6;
2  var degisken2:uint=3;
3  trace(degisken1/degisken2);//işlemin sonucu 2 olarak görülür.
```

- **Mod operatörü (%)**

Bölme işlemindeki kalanı bulmak için kullanılır. Aritmetik işlemleri için önemlidir.

```
1  var  degisken1:uint=51;
2  var  degisken2:uint=10;
3  trace(degisken1 % degisken2);//işlemin sonucu 1 olarak görünür
```

➢ **< (küçük), > (büyük), <= (küçük eşit) , >= (büyük eşit) Karşılaştırma Operatörleri**

İki ifadenin karşılaştırmasını yapar sonuç olarak boolean ifade elde edilir.

```
var  degisken1:uint=5;
var  degisken2:uint=10;
trace(degisken1>degisken2);//işlemin sonucu false olacaktır.
```

➢ **Eşittir operatörü = =**

Atama operatörü ile oldukça karıştırılır. Matematikteki eşitlik, programlamada = = ifadesi ile gösterilir. = = karşılaştırma operatörüdür.

```
var  degisken1:uint=10;
var  degisken2:uint=10;
trace(degisken1==degisken2);//işlemin sonucu true olacaktır
```

➢ **Eşit değildir (!=)**

İfadelerin eşit olmaması durumunda true değerini döndürecektir.

```
var  degisken1:uint=10;
var  degisken2:uint=15;
trace(degisken1!=degisken2);//işlemin sonucu true olacaktır
```

➢ **Atama operatörü (=)**

Sağ taraftaki değeri sol taraftaki değişkene atama işlemi yapar.

```
sayi=5;  //5 değeri sayi değişkenine atanmıştır.
```

➢ **Arttırma operatörü (++)**

Sayının sonuna ya da başına eklenmesi durumunda sayının değerini 1 arttırır.

```
var degisken1:uint=6;
degisken1++;
trace(degisken1); // çıktı ekranında 7 değeri görülür.
```

➢ **Azaltma operatörü (--)**

Sayının sonuna ya da başına eklenmesi durumunda sayının değerini 1 azaltır.

```
var degisken1:uint=6;
degisken1--;
trace(degisken1); // çıktı ekranında 5 değeri görülür.
```

➢ **Ekle ve ata operatörü(+=)**

Yığmalı toplam işlemi yapacaktır. Soldaki değişkenin değerini sağdaki değer kadar arttırıp değişkene yeni değer verir.

"toplam+=i;" ifadesi, "toplam=toplam+i" ifadesinin kısa olarak yazılmasını sağlar.

```
var degisken1:uint=6;
degisken1+=10;
trace(degisken1); // çıktı ekranında 16 değeri görülür.
```

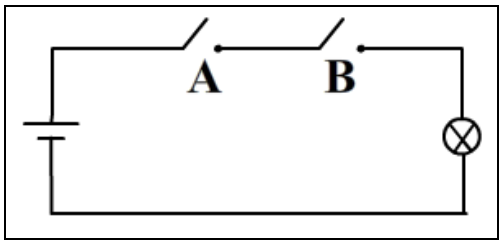
➢ **Mantıksal VE operatörü (&&)**

Koşullardan hepsinin gerçekleşmesi durumunu sınar. Yani şartlardan birisi gerçekleşmezse yanlış (false) değerini döndürecektir.

Aşağıdaki örnekte a'nın 6'dan küçük ve b'nin 4'den büyük olması durumunda ekrana şartlar doğru ifadesi yazdırılacaktır.

```
1  var a:int=5;
2  var b:int=7;
3  if(a<6 && b>4)
4  {   trace("şartlar doğru"); }
```

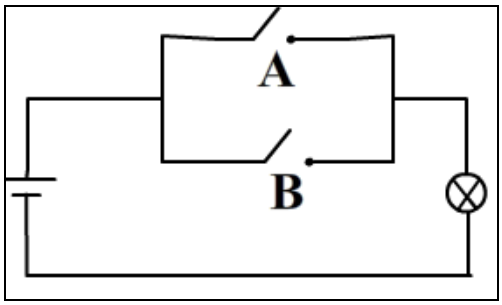
Aşağıdaki elektronik devresinde lambanın yanması için A ve B anahtarının aynı anda kapalı olması gerekir. Aksi halde lamba ışık vermeyecektir.




**Resim 1.25: Ve operatörünün elektronik eşdeğeri.**

➢ **Mantıksal VEYA operatörü ( ||)**

Koşullardan herhangi birisinin gerçekleşmesi durumunu sınar. Şartlardan birisinin gerçekleşmesi doğru (true) değerini döndürecektir.




**Resim 1.26: Veya operatörünün elektronik eşdeğeri.**

## 1.4.4. Değişkenler

Programcılık için değişkenler olmazsa olmazdır. Belirlediğimiz isimlerle oluşmuş bilgisayarın hafızasında yer kaplayan alanlardır. Değişkenleri içerisine bir şeyler koyulabilen defalarca kullanılabilen bardaklar olarak düşünülebilir.

Değişken tanımla yapılmadan program içerisinde kullanılamaz, ActionScript 3.0 ile birlikte değişken tanımlamak zorunlu hale gelmiştir.

➢ Değişken tanımla kuralları aşağıdaki şekilde ifade edilebilir.

- ActionScript 3.0 büyük küçük harf duyarlıdır. Yani MEB, Meb, mEB, meB vb. birbirinden farklı değişkenler olacaktır.
- Sadece İngilizce karakterler kullanınız ı,İ,ş,Ş,ç,Ç,ü,Ü,ö,Ö,ğ,Ğ karakterleri Türkçe karakterlerdir. En çok karıştırılan küçük ı ve büyük İ karakterine dikkat ediniz.
- Özel karakterler kullanmayınız *,?,/ gibi karakterler değişken içersinde kullanılmaz.
- Değişken isimleri rakamla başlayamaz fakat değişken içerisinde rakam kullanılabilir. 1volkan yanlış bir kullanım olacaktır. volkan1, v1olkan, vo1lkan gibi yazılışlar doğrudur.
- Değişken tanımlamada yapılan en büyük hatalardan biri de değişken ismi olarak kısaltma kullanmaktır. Uzun kod satırlarından oluşan uygulamalarda daha sonra bu kısaltmaların ne olduğunu hatırlamak çok zor olabilir. Bu yüzden değişkenlere verilecek isimlerin mantıklı, daha sonra okunduğunda anlaşılacak isimler olması doğru olacaktır. Örneğin: yki yerine yenikullaniciismi yazmak, sonrası için daha kullanışlı olacaktır.
- Kelimeler arasında boşluk bırakılmamalıdır.

➢ **İsimlendirme standartları**

Kod yazarken ya da animasyon programında örnek ismi verirken belirlencek birçok isim olacaktır, bu isimleri belli bir standarta göre yazılması daha sonra ismin kullanıldığı yerlerde acaba "Nasıl yazmıştım?" derdinden kurtaracaktır. Belli standartlar olduğu gibi kullnıcı kendiside bir standart oluşturabilir.

- **Camel standardı**

Yazılan ismin ilk harfinin küçük, birden fazla kelimeden oluşan bir isimse diğer kelimelerin ilk harflerinin büyük olması ile yazılır. ActionScript 3.0 bu standardı kullanmaktadır. Bu standarda göre yazmak zorunlu değildir. Fakat kod yapısına alışılması için bu şekilde bir yazım şeklini benimsemek önemlidir.

**y**eni**K**ullanici**I**smi yazılımı bu standart için bir örnektir.

- **Pascal standardı**

Yazılan değişken isminin kelime ya da kelimelerin ilk harflerinin büyük olması ile yazılır.

**Y**eni**K**ullanici**I**smi yazılımı bu standart için bir örnektir.

- **Underscore standardı**

Yazılan kelimelerin arasına _ işareti koyarak yazım yapılır

Yeni_kullanici_ismi yazılımı bu standart için bir örnektir.

### 1.4.4.1. Değişken Tanımlama

ActionScript içerisinde değişken tanımlamak için *variable* kelimesinin kısaltılmışı olan *var* kelimesini kullanmalıyız.

Şu kalıbı kullanarak değişken tanımlarız

*Var Değişken İsmi: Veri Türü;*

Yukarıdaki kalıp cümlede değişken oluşturulmuş fakat içerisine değer atanmamıştır. Daha önce bahsedilen bardak örneğini düşünüldüğünde; bardak oluşturuldu fakat içerisine bir değer atamamk için atama operatörü kullanılmadır.

Bu işlemi iki şekilde yapılabilir.

- Burada değişken oluşturulmuş ve aynı satırda değeri kendisine atanmıştır.

```
1  var Değişken İsmi:Veri Türü = Değişken Değeri;
2  var sayilar      :uint      = 10;
```

- Birinci satırda değişken oluşturma ikinci satırda değer atama işlemi yapılmıştır.

```
1  var Değişken İsmi:Veri Türü;
2  Değişken İsmi=Değişken Değeri;
3
4  var sayilar      :uint;
5  sayilar      =10;
```

Burada tercih programcıya kalmıştır.

Farklı veri türlerinden değişken tanımlak için:

- Yeni bir çalışma sayfası açınız.
- Eylemler panelini çalıştırınız.
- Pozitif tamsayı ,tamsayı,string, boolean veri türlerinde değişkenler tanımlayıp rastgele değerler veriniz.

```
1  var donguDegiskeni:uint=0;
2  var isim:String="melek";
```

- Burada unutmamanız gereken string ifadeler " " işaretleri arasında yazılmalıdır.

```
1  var sayi:Number=25.456;
2  var dogrumu:Boolean=false;
```

- Eşitliğin sağ tarafında veri türüne göre değeri oluşturacak ifade direk olarak yazılabilir.

```
1  var sayi:uint=19-8;
2  //değişkenin değeri 11 olacaktır.
3  var dogrumu:Boolean=19>80;
4  /*19 sayısı 80 den küçük olduğu için şart yanlış olacak
5  ve doğrumu değişkeni false değeri alacaktır.*/
```

### 1.4.5. Sabitler

Program içerisinde değeri hiç değişmeyen yapılara sabit denir. Örneğin, pi sayısı matematiksel bir sabittir. Değeri asla değişmeyecektir. Ya da sizin uygulamanız için program boyunca sabit kalmasını isteyeceğiniz bir katsayı değeri de sabit olarak tanımlanabilir.

Sabit tanımlamanın değişken tanımlamadan tek farkı, en başta yazılan var yerine *const* yazılmasıdır.

*const pi:Number=3.14;*

Bu ifadenin program içerisinde değeri değiştirilmek istenirse, animasyon yazılımında derleyici hatası alınacaktır.

*const pi:Number=3.14;*
*pi=4.3;*

Bu iki satır yazılıp derlemeye çalışıldığında 1049 (Sabit olarak belirtilmiş bir değişkene kuraldışı atama) hatası verecektir.

### 1.4.6. Trace Cümlecikleri

Kodlama yaparken kodların sonucunu ya da aldığı değerleri görme ihtiyacı oluşur. Bir başka deyişle kodların doğru olup olmadığı tst edilmek istenebilir. Bu gibi durumlarda *trace* büyük fayda sağlayacaktır. İstenilen anda istenilen değişkenlerin anlık değerleri ya da istenilen ifade output ekranına yazdırılabilir.

Kullanımı oldukça basittir. Tek değişken yazdırılabileceği gibi birden fazla değişken ve string ifade de yazdırabilir. Örneğin;

```
1  trace(kullaniciIsmi);
2
3  trace("Kullanıcı ismi:",kullaniciIsmi,"yaşı:",kullaniciYasi);
```

Trace içindeki her ifade birbirinden virgül işareti ile ayrılmış ve string ifadeler çift tırnak içerisinde yazılmıştır.

## UYGULAMA FAALİYETİ

ActionScript panelini açmak, değişken tanımlamak, operatörleri değişkenlerle kullanmak ve çıkış panelini çalıştırmak için aşağıdaki uygulamayı yapınız.

| İşlem Basamakları | Öneriler |
|---|---|
| ➢ Yeni bir animasyon belgesi oluşturunuz. | ➢ Dosya > Yeni (CTRL + N) komutunu kullanabilirsiniz. |
| ➢ Birinci frame sağ tıklayıp eylemleri seçiniz. | ➢ Değiştir > Belge (CTRL + J) komutunu ile Belge ayarları iletişim penceresini kullanabilirsiniz. |
| ➢ Camel lokasyonuna göre ad soyad değerlerini tutacak bir değişken ismi bulunuz. | ➢ İlk kelime küçük, bitişik diğer kelimelerin ilk harflerini büyük olarak yazabilirsiniz. |
| ➢ Veri türü olarak metinsel veri türünü kullanınız. | ➢ Metinsel veri türü olarak String ifadesini kullanabilirsiniz. |
| ➢ Değişkene kendi isminizi değer olarak atayınız. | ➢ Atama yaparken = operatörünü kullanınız. |
| ➢ Yaşınız bilgisini tutacak bir değişken ismi bulunuz. | ➢ Değişken ismi belirlenirken en kısa ve en anlamlı kelimeyi bulmaya çalışınız. |
| ➢ Uygun veri türünü belirleyiniz. | ➢ Pozitif tam sayılar için kullanılan uint veri türünü kullanabilirsiniz. |
| ➢ iki değişkeni birbirine ekleyiniz. | ➢ İçerisinde string ifade bulunan değişkenleri + operatörü ile birbirine ekleyebilirsiniz. |
| ➢ Yaşınızı tutan değişken değerini 1 arttırınız. | ➢ ++ operatörlerini kullanarak ifadelere 1 ekleyebilirsiniz. |
| ➢ Çıkış paneline ad soyad bilgisini tutan değişkeni yazdırınız. | ➢ Trace() komutu ile ifadeleri çıkış paneline yazdırabilirsiniz. |
| ➢ Animasyonu test ederek görüntüleyiniz. | ➢ Kontrol et > Filmi test Et > Test Et (CTRL + Enter) komutunu kullanabilirsiniz. |

## KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadığınız beceriler için **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| **1.** Animasyon yazılımını çalıştırabildiniz mi? | | |
| **2.** Eylemler panelini açabildiniz mi? | | |
| **3.** Camel lokasyonuna göre değişken ismi buldunuz mu? | | |
| **4.** Veri türü olarak string ifadeyi tanımlayabildiniz mi? | | |
| **5.** Atama operatörünü kullanarak isminizi değer olarak verdiniz mi? | | |
| **6.** Değişken isimlendirme kurallarına uygun isim verdiniz mi? | | |
| **7.** Yaşınız bilgisini tutan değişken için veri türünü uint yaptınız mı? | | |
| **8.** İki değişkeni birbirine eklerken = operatörünün neden eklediğini anladınız mı? | | |
| **9.** ++ operatörünün nasıl çalıştığını öğrendiniz mi? | | |
| **10.** Trace() ifadesi ile adınızı soyadınızı yazdırabildiniz mi? | | |
| **11.** Animasyonu test edebildiniz 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.** (  ) ActionScript 3.0 nesne yönelimli bir programlama dilidir..

**2.** (  ) Eylemler panelinde sadece 1 satıra açıklama satırı ekleyebiliriz.

**3.** (  ) Değişken isimleri rakamlarla başlayamaz.

**4.** (  ) Atama ( = ) operatörü iki ifadenin birbirine olan eşitliğini gösterir.

**5.** (  ) Arttırma operatörü (++) değişkenin değerini 1 arttırır.

**6.** (  ) Trace ifadesi sadece değişkenlerin değerini gösterebilir.

**7.** (  ) Negatif tam sayılar için uint veri türünü kullanabiliriz.

**8.** (  ) Eylemler panelinde satır numaralarını göstermek için CTRL+Shift+L kısa yolunu kullanabiliriz.

**9.** (  ) Sahnedeki tüm nesnelerin örnek isimleri aynı olabilir.

**10.** (  ) Tüm şartların doğru olduğunu sınayan operatör veya ( || ) dır.

## 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 faaliyetine geçiniz.

# ÖĞRENME FAALİYETİ–2

**AMAÇ**

Animasyon yazılımını kullanarak fonksiyon işlemlerini gerçekleştirebileceksiniz.

**ARAŞTIRMA**

- Diğer programlama dillerinde fonksiyon tanımlama ve kullanmayı araştırınız.

# 2. FONKSİYONLAR

## 2.1. Fonksiyonlar

Uygulama içerisinde bir komut ya da komut grubu tekrar tekrar kullanmak istenebilir. Bu gibi durumlarda aynı satırlar defalarca yazılmak zorunda kalınabilir. Birkaç yüz satır kodlamada çok büyük problem olmayacaktır. Fakat uygulama büyüdükçe hata kontrolü yapması da zorlaşacaktır. Tekrar eden kodlarda belli bir değer değiştirmek istendiğinde tüm tekrar eden satırların yeniden düzenlenmesi gerekecektir.

Bu tarz durumlarda uygulama geliştiricilerin yardımına fonksiyonlar yetişmektedir. Kısaca fonksiyonlar, tekrar eden kod bloklarının alınıp paketlenmesine yarar. Uygulama içerisinde artık bu paketin ismi kullanılarak istenilen yerde bu kodlar çalıştırılabilir. Bu sayede daha sonra üstüne yapılacak değişiklikler ve hata kontrolü hızlı bir biçimde yapılabilir.

## 2.2. Fonksiyon Tanımlama

Fonksiyon tanımlama aşağıdaki şekilde yapılabilir: Bu şekilde fonksiyon tanımlandığında editör penceresinin istenilen yerinde fonksiyon çalıştırılabilir.

Fonksiyonu tanımlamak için:

```
1  fonksiyonismi ():void
2  {
3  Fonksiyon çalıştığında yapılacak işler
4  };
```

Fonksiyonu çalıştırmak için:

```
1  fonksiyonismi()
2
```

Şeklinde yazarak istenilen yerde fonksiyon çalıştırılır.

Sahnedeki örnek ismi “kutu” olan nesneyi x ekseninde 250, y ekseninde 250 px noktasına getirecek fonksiyonu yazmak için:

- Yeni bir belge oluşturunuz.
- Sahnemize 20* 20 px büyüklüğünde kare oluşturunuz.
- Film klibi haline getirip örnek adı olarak *kutu* ismini veriniz.
- Sahneye tıklayıp Pencere > Eylemler (F9) komutunu seçiniz.
- Fonksiyon ismini *ilerlet* belirleyin. İsterseniz değişken tanımlama kuralları çerçevesinde istediğiniz ismi verebilirsiniz.

```
1  function ilerlet():void
2  {
3  kutu.x=250;
4  kutu.y=250;
5  }
```

- Animasyonu test ediniz.
- Kutu nesnesinin yeri değiştirmeyecektir.
- Fonksiyon tanımlanmış, fakat tetiklenmediği için kodlar çalışmamaktadır.
- Tetiklemek için fonksiyon komutunu eylem paneline eklememiz gereklidir.

```
1  function ilerlet():void
2  {
3  kutu.x=250;
4  kutu.y=250;
5  }
6  ilerlet();
```

## 2.3. Fonksiyonlara Veri Gönderme - Alma

Fonksiyonların daha önce belli işlerin yapılmasını sağlayan paketlenmiş kodlar olduğundan bahsedilmişti. Bu kod paketlerine isteğe göre değer verip, işlenmiş verinin sonucu geri alınabilir. Böylelikle fonksiyonlar anlık olarak verileri işleyip sonucu programa dâhil edebilir.


**Resim 2.1: Fonksiyonlara veri gönderme - alma**

Örneğin, dairenin alanını hesaplayan bir fonksiyonun yazıldığını düşünelim. Fonksiyon her çalıştırıldığında girilen yarıçap verisine göre dairenin alanını hesaplayıp sonucu ekrana yazdırsın.

```
1  function daireninAlani(r:Number):void
2  {
3  // dairenin alanı pi*r2
4
5  var alan:Number=3.14*r*r;
6  trace("dairenin alanı : ",alan);
7  }
```

Fonksiyon yukarıdaki kodlarla *daireninAlani* ismiyle paketlendi. Şimdi fonksiyonu 5 yarıçap değerine göre çalıştırılmak istendiğinde;

```
1  function daireninAlani(r:Number):void
2  {
3  // dairenin alanı pi*r2
4
5  var alan:Number=3.14*r*r;
6  trace("dairenin alanı : ",alan);
7  }
8  daireninAlani(5);
```

Şeklinde fonksiyonu eklenmesi yeterlidir.

Kodlar çalıştırıldığında aşağıdaki şekilde sonucu görülecektir:

Zaman Çizelgesi | Çıktı | Hareket Düzenleyici | Derleyici Hataları

```
dairenin alanı :  78.5
```

Yukarıdaki örnekte fonksiyona tek değer gönderildi. Uygulama içerisinde birden fazla değerle (parametre) fonksiyon çalıştırılması gerekebilir. Örneğin daire, dik prizmanın hacmini hesaplayan bir fonksiyon yapıldığında yarıçap ve yükseklik bilgilerinin fonksiyona gönderilmesi gerekecektir.

Birden fazla değer (parametre) göndermek için parametre tanımlanırken istenilen parametrelerin tanımlanıp aralarının virgülle ayrlması gerekir.

```
1  function prizmaHacmi(r:Number,h:Number):void
2  {
3  // herhangi dik prizmanın hacmi = taban alanı*yüksekliktir.
4  // dairenin alanı pi*r2
5  var alan:Number=3.14*r*r;
6  var hacim:Number=alan*h;
7  trace("Prizmanın Hacmi : ",hacim);
8  }
9  prizmaHacmi(3,6);
```

$P(n)=n^3-3n^2+4n-2$ polinomu için P(2)= ?, P(0) = ?, P(1) = ? değerlerini hesaplayan bir fonksiyon yazmak için:

- Yeni bir çalışma sayfası açıp eylemler panelini çalıştırınız.

- Hesapla isminde bir fonksiyon oluşturun. Fonksiyon polinomdaki matematiksel işlemi yapacaktır.

- Küp ya da kare alma komutları ilerleyen konularda ele alınacağı için temel matematiksel işlemlerini kullanarak fonksiyonu çalıştırınız.

```
1  function hesapla(n:Number):void
2  {
3  var sonuc:Number;
4  sonuc=(n*n*n)-(3*n*n)+(4*n)-2;
5  trace("polinomun sonucu :",sonuc);
6  }
7  hesapla(2);
8  hesapla(0);
9  hesapla(1);
```

- Animasyonu test ederek, ekran çıktısını kontrol ediniz.

```
Zaman Çizelgesi | Çıktı | Hareket Düzenleyici | Derleyici Hataları
polinomun sonucu : 2
polinomun sonucu : -2
polinomun sonucu : 0
```

Fonksiyonlarda parametreden sonra yazılan *void* kelimesi, fonksiyonun geriye değer döndürmediği anlamına gelmektedir. Yani fonksiyon, yaptığı işlem sonucunda uygulamaya herhangi bir veri göndermemektedir.

Fonksiyonlardan veri almak için yukarıdaki alan örneğini tekrar yapınız.

```
1  function daireninAlani(r:Number):Number
2  {
3  // dairenin alanı pi*r2
4  return(3.14*r*r);
5  };
6
7  var gelenSayi:Number=daireninAlani(2);
8  trace(gelenSayi);
```

*"void"* kelimesi yerine fonksiyonun ne tür veri geri döndüreceğini yazılmaktadır. Yukarıdaki örnekte fonksiyon dairenin alanını hesaplayıp return kelimesiyle number türünde bilgiyi fonksiyon dışına aktarmaktadır. "gelenSayi" değişkeninin değeri 12.56 olacaktır.

# UYGULAMA FAALİYETİ

ActionScript 3.0 kullanarak fonksiyon yazmak ve kullanmak için aşağıdaki uygulamayı yapınız.

| İşlem Basamakları | Öneriler |
|---|---|
| ➢ Yeni bir animasyon belgesi oluşturunuz. | ➢ Dosya > Yeni (CTRL + N) komutunu kullanabilirsiniz. |
| ➢ Birinci frame sağ tıklayıp eylemleri seçiniz. | ➢ Değiştir > Belge (CTRL + J) komutunu ile Belge ayarları iletişim penceresini kullanabilirsiniz. |
| ➢ islem isminde bir fonksiyon tanımlayınız. | ➢ Tanımlama yapanilmek için function kullanmalısınız. |
| ➢ Fonksiyonumuz bilgi döndürmesin | ➢ Veri döndürmemesi için void kullanabilirsiniz. |
| ➢ Fonksiyon içerisine gelen a ve b tam sayı değerleri fonksiyon içinde çarpılsın. | ➢ Fonksiyon isminden sonra a ve b değişkenlerini tanımlamayı unutmayınız. |
| ➢ Çarpımın sonucunu çıkış ekranında yazdırınız. | ➢ Trace ifadesi ile yazdırma yaptırabilirsiniz. |
| ➢ 3 ve 4 değerleri için fonksiyonu çalıştırınız. | ➢ Panele fonksinyon isminden sonra değerleri yazarak istediğin yerde çalıştırabilirsin. |
| ➢ Fonksiyonun geriye number türünde değer döndürmesi için kodları yeniden düzenleyiniz. | ➢ Void yerine Number yazabilirsiniz. |
| ➢ Return ifadesini kodlara ekleyiniz. | ➢ Fonksiyon dışarıya hangi ifadeyi döndürecekse onun başına return koymalısınız. |
| ➢ Fonksiyonun dışarıya vereceği değeri başka bir değişkene aktar | ➢ Fonksiyon dışında değişken tanımlamayı ve atama operatörünü kullanmayı unutmayınız. |
| ➢ Çalışmayı test ederek görüntüleyiniz. | ➢ Kontrol et > Filmi test Et > Test Et (CTRL + Enter) komutunu kullanabilirsiniz. |

## KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadığınız beceriler için **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| **1.** Fonksiyon tanımlayıp söz dizimini uygun şekilde yaptınızmı? | | |
| **2.** Geriye değer döndürmemesini sağladınızmı? | | |
| **3.** Tanımladığınız fonksiyonu sahne içerisinde kullanabildinizmi? | | |
| **4.** Fonksiyona veri gönderebilmek için fonksiyona parametre tanımı yapabildinizmi? | | |
| **5.** Gönderdiğinizi verileri fonksiyon içerisinde kullanabildinizmi? | | |
| **6.** Fonksiyon sonucunu sahnede görebildinizmi? | | |
| **7.** Fonksiyonun değer döndürmesini sağladınızmı? | | |
| **8.** Geriye değer döndüren fonksiyonu uygulama içerisinde kullanabildinizmi? | | |
| **9.** Geriye değer döndüren ile döndürmeyen fonksiyonu sahnede kullanabildinizmi? | | |

## 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.** ( ) Bütünü parçalara ayırmak için fonksiyonları kullanabiliriz.

**2.** ( ) Fonksiyon tanımlayarak aynı kod bloğunu hızlı bir şekilde başka bölümlerde kullanabiliriz.

**3.** ( ) Fonksiyon tanımlamak için variable kelimesinin kısaltılmışı olan var kelimesi kullanılır.

**4.** ( ) Dışarıdan veri alırken parametre tanımlamak zorundayız.

**5.** ( ) Void ve return kelimelerini aynı fonksiyon içersinde kullanabiliriz.

**6.** ( ) Geriye değer döndürürken return kelimesi kullanılır.

**7.** ( ) Fonksiyonun ne tür değer döndüreceği fonksiyon tanımlanırken belirtilmelidir.

## 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 faaliyetine geçiniz.

# ÖĞRENME FAALİYETİ–3

**AMAÇ**

Olaylarla ilgili düzenlemeleri yapabilecek ve sahnede olayları kontrol edebileceksiniz.

**ARAŞTIRMA**

- İnternetdeki ActionScript ile yapılmış oyun ve uygulamaları inceleyip klavye ve fareye verdikleri tepkileri inceleyiniz.

# 3. OLAYLAR (EVENT)

## 3.1. ActionScript Olayları

ActionScript 3.0 ile artık sahnede gerçekleşen her şey, bir olay olarak kabul edilmiştir. Örneğin; farenin herhangi bir yere tıklaması, klavyeden bir şeyler yazılması, sahnenin çalışması, zaman çizgisi üzerinde bir frame'in çalışması gibi… Her şey artık olay olarak kabul edilmektedir. Olayların gerçekleşmesi yazılmış fonksiyonların tetiklenmesini sağlar.

Her nesne için ayrı olaylar tanımlanabilir. Tanımlanan bu olaylar olay dinleyicileri ile izlenmektedir. Uygulamayı istendiği gibi geliştirmek için olayların çok iyi bilinmesi gerekir.

ActionScript içerisinde kullanılabilecek birçok olay bulunmaktadır. ActionScript dilinin gücü de buradan gelmektedir. Bunlar fare olayları, klavye olayları, zamana bağlı olaylar olarak ayrılabilir. Burada temel olaylar işlenecektir. Bu olaylar sistemin otomatik tetiklediği ve kullanıcının tetiklediği olaylar olarak ikiye ayrılmaktadır.

"Neden onlarca olay dinleyicisi var?" sorusu akla gelebilir. İnteret üzerinden oyunlar oynanırken bazı durumlarda farenin tıklanması yetmeyecektir. Sürüklenmesi veya belli bir bölgede dolaştırılması ya da klavyeden belli bir tuşa/tuşlara basılması istenebilir. İşte bu yüzden gereken tüm durumlar için olay yöneticisi oluşturulabilir.

## 3.2. EventListener

Olay, dinleyicilerini sahneye eklemek ve tetikleyeceği fonksiyonu tanımlamak şu şekilde yapılır: Öncelikle olay dinleyicisini nesneye eklemek gerekir.

Kare örnek ismine sahip bir nesneye olay dinleyicisi eklemek için:

- Sahneye bir kare çiziniz ve kareyi film klibi haline dönüştürünüz.
- "Örnek Adı" (instance name) bölümüne *kare* ismini verin.
- Kare nesnesinin üzerine fareyle gelindiğinde fare fonksiyonu çalışması ve fare karenin üzerinde çıktısını vermesi isteniyor. Buna göre olay dinleyicisini ekleyiniz.

```
1
2
kare.addEventListener();
```

- Burada nesnemizi hangi olaya karşı tetikleceği belinmektedir. Farenin nesne üzerine gelmesi buradaki olaydır.

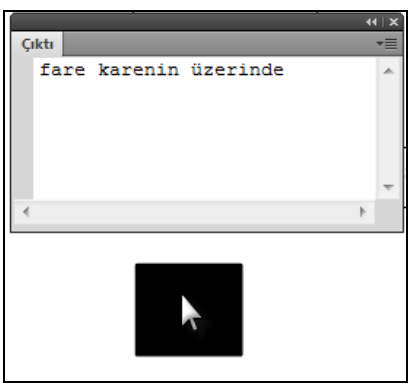
```
1
2
kare.addEventListener(MouseEvent.MOUSE_OVER,fare);
```

- Fare karenin üstüne geldiğinde fare fonksiyonumuz çalışacaktır.

- Fare fonksiyonunu yazın. Fare fonksiyonu, fonksiyonlar konusunda öğrendiğinizin haricinde bir konu içermemektedir. Buradaki fonksiyonun tek farkı, içerisine gönderilecek parametre farklılık göstermektedir. Fonksiyona mouse olayı olduğunu anlatmak için parametre olarak *event:MouseEvent* söz dizimini kullanınız.

```
1  kare.addEventListener(MouseEvent.MOUSE_OVER,fare);
2  function fare(event:MouseEvent):void
3  {
4  trace("fare karenin üzerinde");
5  }
```

- Filmi test ediniz. Fare kare üzerine geldiğinde çıkış ekranında "fare karenin üzerinde" uyarısı görülmelidir.




**Resim 3.1: Olay yöneticisinin çalışması**

Uygulama içerisinde nesneye atanan olay iptal edilmek istenebilir. Örneğin, bir oyunda şart gerçekleşmişse nesne üzerindeki olayı silmek ya da değiştirmek gerekebilir. Bu gibi durumlarda *removeEventListener* komutunu kullanılarak nesne üzerindeki olaylar silinebilir.

Yukarıda yaptığımız örnekteki olay dinleyicisini silmek için:

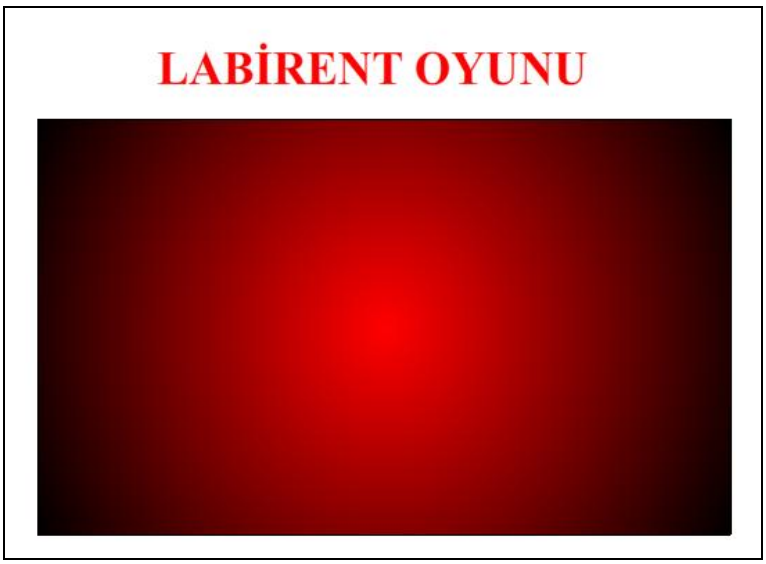
- *kare.removeEventListener(MouseEvent.MOUSE_OVER,fare);* komutunu kullanınız. Bu komutu bir fonksiyon ya da bir karar yapısının içerisinde kullanabilirsiniz.

**Örnek:**

Şimdi öğrendiklerimizle basit bir oyun tasarlayalım. Oyunun amacı fare ile labirentin yolları arasında ilerleyip labirentin orta noktasındaki çıkış kapısına ulaşmaktır. Kapıya ulaşmaya çalışırken duvarlara temas edildiğinde ceza puanı oluşarak üç defa duvara temas edildiğinde "Oyun Bitti" uyarısı alınacaktır ve bu durumdan sonra skor çalışmayacaktır.

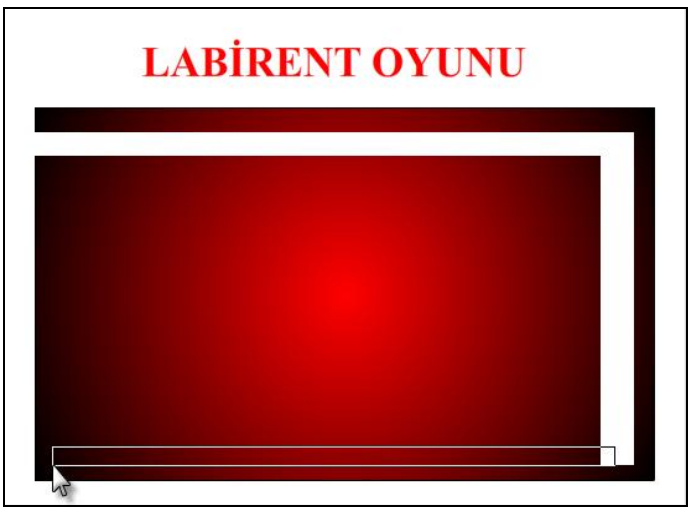
Oyunu hazırlamak için:

- Yeni bir çalışma sahnesi açın ve boyutu 500x300 piksel olan bir dikdörtgen çizin.

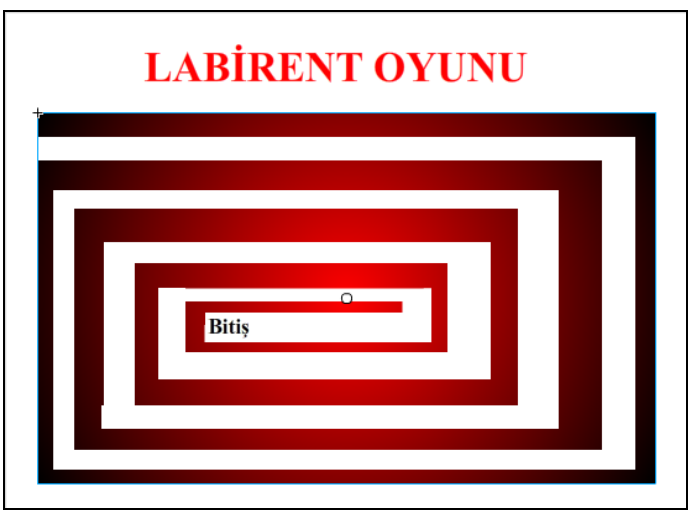



**Resim 3.2: Labirent oyunu ana sahnesi.**

➢ Dikdörtgen içerisine labirent yollarını oluşturmak için seçme aracı (v) ile seçip delete ile dikdörtgen içerisinden çıkartın. Labirentin zorluk derecesini yolları daraltıp genişleterek beliryeyebilirsiniz.

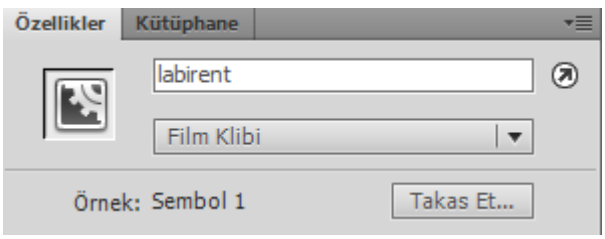



**Resim 3.3: Labirent oyunu ana sahnesi yol hazırlama**




**Resim 3.4: Labirent oyunu sahnenin tamamlanmış hali**

➢ Labirenti seçip film klbine dönüştürün. Örnek ismi olarak *labirent* ismi verin.




**Resim 3.5: Yapılan çizim için örnek ismi oluşturma**

- Öncelikle oyunda hangi olay dinleyicisinin kullanılacağına karar vermeniz gerekmektedir. Buraya kadar sadece farenin bir nesne üzerine gelme olayını knlarda yer aldı. Bu olay bu oyun için yeterli olacaktır.

- Labirent nesnesine olay dinleyicisini ekleyin. Tetikleyeceği fonksiyonun ismini *oyun* olarak ayarlayın.

```
1  labirent.addEventListener(MouseEvent.MOUSE_OVER,oyun);
2
```

- Artık animasyon yazılımı, labirent nesnesi üzerine farenin gelip gelmediğini gözleyecek ve fare labirent üzerinde dolaşırsa oyun fonksiyonunu çalıştıracaktır.

```
1  function oyun(event:MouseEvent):void
2  {
3  Bu bölüme artık oyunun gerçekleşmesi için
4  gerekli kodları yazacağız.
5  }
```

- Farenin labirentin üzerine her geldiğinde skor tutacak bir değişkene ihtiyacımız var. Fonksiyonun dışında başlangıç değeri 0 tam sayı tipinde *say* isminde bir değişken tanımlayın. Bu değişkeni fonksiyon içerisinde tanımlanırsa fonksiyon her çalıştığında *say*'ın değeri 0 olacaktır.

```
1  var say:uint=0;
2
```

- Fonksiyon her çalıştığında değişkenin değeri 1 artmalıdır. *say*++ ya da *say*=*say*+*1* (yığmalı toplam) komutları kullanılabilir.

- Basit bir karar yapısı kullanarak *say* değişkeninin değeri 3 olduğunda yani üç kere kare duvarlarının üzerine gelirse oyunu bitirecek ve çıkış ekranına oyun bitti uyarısını verecek. Bu karar yapısı ilerleyen konularımız içerisinde ayrıntılı olarak işlenecektir.

```
1  if (say==3)
2  {
3  trace("oyun bitti");
4  labirent.removeEventListener(MouseEvent.MOUSE_OVER,oyun);
5  /*labirent üzerindeki olay yöneticimizi silerek
6  artık farenin nerede gezdiği önemsiz hale geliyor.*/
7  }
```

- Oyunun son kodları

```
var say:uint=0;

labirent.addEventListener(MouseEvent.MOUSE_OVER,oyun);

function oyun(event:MouseEvent):void
{
say++;
if (say==3)
{
    trace("oyun bitti");
    labirent.removeEventListener(MouseEvent.MOUSE_OVER,oyun);
    }
}
```

şeklinde olacaktır.

Görüldüğü üzere animasyon yazılımı, basit kodlarla ve hayal gücüyle birlikte güzel oyunlar geliştirmeni sağlamaktadır.

## 3.3. Fare Olayları

### 3.3.1. CLICK

Farenin bir nesneye sol tuş ile tıklanıp bırakılma olayıdır.

mouseEvent.CLICK şeklinde yazılır.

### 3.3.2. DOUBLE_CLICK

Farenin bir nesneye çift tıklama olayıdır.

mouseEvent.DOUBLE_CLICK şeklinde kullanılır

### 3.3.3. MOUSE_DOWN

Farenin sol tuşuna basma olayıdır. Click den farkı sadece basma ile çalmasıdır.

mouseEvent.MOUSE_DOWN şeklinde kullanılır.

### 3.3.4. MOUSE_UP

Farenin sol tuşunu bırakma olayıdır.

mouseEvent.MOUSE_UP şeklinde kullanılır.

### 3.3.5. MOUSE_OUT

Farenin bir nesne üzerinden ayrılma olayıdır.

mouseEvent.MOUSE_OUT şeklinde kullanılır.

### 3.3.6. MOUSE_MOVE

Farenin nesne üzerinde gezme olayıdır.

mouseEvent.MOUSE_MOVE şeklinde kullanılır.

### 3.3.7. MOUSE_OVER

Farenin nesne üzerine gelme olayıdır.

mouseEvent.MOUSE_OVER şeklinde kullanılır.

### 3.3.8. MOUSE_WHEEL

Farenin tekerleğinin döndürülme olayıdır.

mouseEvent.MOUSE_WHEEL şeklinde kullanılır.

### 3.3.9. ROLL_OVER

Farenin sol tuşunun basılı tutularak nesne üzerine gelme olayıdır.

mouseEvent.ROLL_OVER şeklinde kullanılır.

### 3.3.10. ROLL_OUT

Farenin sol tuşunun basılı tutularak nesne üzerinden ayrılma olayıdır.

mouseEvent.ROLL_OUT şeklinde kullanılır.

## 3.4. Klavye Olayları

Klavyeden basılan tuşları algılamayı saplayan olaylardır. İki adet davranış sergiler.

### 3.4.1. KEY_DOWN

Klavyeden herhangi bir tuşa basma durumunda çalışır. Hangi tuşa basıldığını anlamak için keycode özelliği kullanılabilir.

```
1  import flash.events.KeyboardEvent;
2
```

Buradaki kod flash tarafından otomatik eklenecektir. Event sınıfının klavye metotlarını uygulama içerisine dâhil edecektir.

```
1  function klavye(event:KeyboardEvent):void
2  {
3  trace(event.keyCode)
4  }
5  stage.addEventListener(KeyboardEvent.KEY_DOWN,klavye);
```

Yukarıdaki kodları çalıştırıldığında klavyeden basılan "v" tuşu için asci kodu çıkış ekranında görülebilir.


**Resim 3.6: v tuşunun asci kod karşılığı**

### 3.4.2. KEY_UP

Klavyeden basılmış bir tuşun bırakılması olayıdır.

```
1  import flash.events.KeyboardEvent;
2
```

Buradaki kod, flash tarafından otomatik eklenecektir. Event sınıfının klavye metotlarını uygulama içerisine dâhil edecektir.

```
1  function klavye(event:KeyboardEvent):void
2  {
3      trace(event.keyCode)
4  }
5  stage.addEventListener(KeyboardEvent.KEY_UP,klavye);
```

Örneğimizi çalıştırdığımızda klavyeden elinizi çektiğiniz anda, çıkış ekranında basılan tuşun kodunu görebilirsiniz.

## 3.5. Zamana Bağlı Çalışan ve Tekrar Eden Olaylar

ActionScript 3.0 içerisinde ENTER_FRAME, TIMER ve TIMER_COMPLETE olayları en fazla kullanılan olaylar arasındadır. ENTER_FRAME olayı uygulama başladığından itibaren sonuna kadar çalışır. Timer olaylarında çalışma sürelerini belirlenebilir.

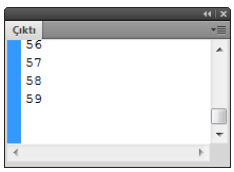
### 3.5.1. ENTER_FRAME

Bu olay sahne çalışma hızına bağlı olarak (Frame Rate) sürekli fonksiyonları çalıştırır. Sahne çalıştırıldığı anda tetiklenmeye başlar ve sahne hızına göre sürekli olarak uygulama kapatılana kadar çalışır.

Sahne çalışma hızı değiştirildiğinde tetiklenen fonksiyonun çalışması da değişecektir. Uygulama süresince tetiklenen fonksiyon çalışacağından işi bittiğinde tetikleyiciyi silmek doğru bir davranış olacaktır.

```
1  var say:uint=0;
2  stage.addEventListener(Event.ENTER_FRAME,calis);
3
4  function calis(event:Event):void
5  {
6  trace(say++);
7  }
```

Yukarıdaki sahneyi izleyen tetikleyici çalıştığı anda *calis* fonksiyonu tetiklenecek ve fonksiyon içindeki *say* değişkeni, sahne hızına göre kendini arttıracaktır. Çıkış ekranında say değişkeninin değerinin arttığını görülebilir. Sahne hızı arttırıldığında *say* değişkenin değeri daha hızlı artacaktır.




**Resim 3.7: Say değişkeninin sahne çalıştığındaki artan değeri**

## 3.6. Kod Parçacıkları (CodeSnipet)

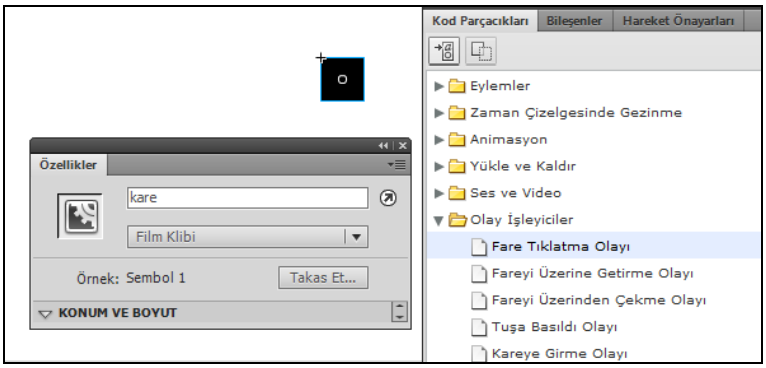
Kod parçacıkları paneli animasyon yazılımının beşinci versiyonunda gelmiş bir yeniliktir. Kod yazmak yerine panelden ilgili kodu seçip çift tıklamak yeterlidir. Panel; eylemler, zaman çizgisi kontrolleri, hazır animasyonlar, dışarıdan dosya yükle ve kaldır, ses ve video, olay yöneticileri şeklinde altı bölümden oluşmaktadır.

Paneli açmak için Pencere > Kod parçacıkları komutu ya da ActionScript editör penceresinin köşesinden 

seçilip açılabilir. Kodların sahneye düzgün eklenip çalışması için sahneye o kodun atanacağı bir nesne olmak zorundadır. Nesnenin örnek isminin (instanse name) tanımlanmış olması gereklidir yoksa panel kendisi o nesneye bir isim verecektir.

Kod parçacıklarını kullanmakiçin:

- Sahnedeki kare nesnesine seçin ve kod parçacıkları bölümünü açın.




**Resim 3.8: Kod parçacıkları paneli**

- İstediğiniz eylemi seçerek çift tıklayın. Kod otomatik olarak eklenecektir.

```
/* Fare Tıklatma Olayı
Belirtilen sembol örneğini tıklattığınızda
kendi özel kodunuzu ekleyebileceğiniz bir işlev yürütülür.
Talimatlar:
1. Aşağıda "// Özel kodunuzu başlatın" yazan satırdan sonra gelen
yeni bir satıra özel kodunuzu ekleyin.Sembol örneği tıklatıldığında
kod yürütülür.
*/
kare.addEventListener(MouseEvent.CLICK, fl_MouseClickHandler_3);

function fl_MouseClickHandler_3(event:MouseEvent):void
{
    // Özel kodunuzu başlatın
    // Bu örnek kod, Çıktı panelinde "Fare Tıklatıldı"
    //sözcüklerini görüntüler.
    trace("Fare tıklatıldı");
    // Özel kodunuzu sonlandırın
}
```

## 3.7. ActionScript İle Animasyon

Daha önceki konularda geçen, zaman çizgisi üzerinde yapılan animasyonlar ActionScript kodu yazarak da yapılarak kontrol edilebilir. Bu işe yarayan animasyon sınıfları nesnelerin şeklini değiştirmek ve onlara hareket kazandırmak üzere ikiye ayrılır.

### 3.7.1. Biçim Animasyonları

10 adet biçim animasyonu vardır.

- **Blinds:** Nesneye Panjur efekti verir.
- **Rotate:** Nesneyi döndürür.
- **Wipe:** Nesne üzerine silme efekti yapar.
- **Zoom:** Yakınlaştırma efekti uygular.
- **Fly:** Uçma efekti yapar.
- **Irıs:** Açılıp kapanan iris efekti yapar.
- **Photo:** Görünür kaybolur efekti yapar.
- **Squeeze:** Sıkma efekti uygular.
- **PixelDissolve:**piksel piksel yok etme ya da oluşturma efekti yapar.
- **Fade:** Nesneyi yok etme ve görünür yapma efekti uygular.

**Örnek:**

Yukarıdaki animasyon, sınıflarının yazılacağı animasyon komutu içerisinde tip olarak belirtilmeldir. Biçim animasyonlarının kullanılabilmesi için *TransitionManager* kullanılması gereklidir.

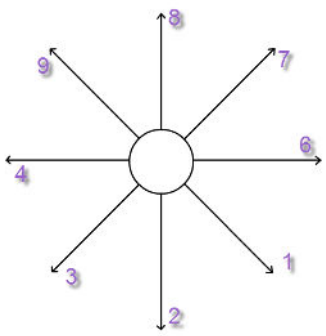
- Animasyona başlamadan önce *transition* ve *transition.easing* sınıf ve metotlarını sahneye *import* edin.
- Sahneye 100 x 100 piksel büyüklüğünde bir kare çiziniz.
- Karenin rengini belirleyiniz.
- Film klibi haline getirip örnek ismini *kare* olarak belirleyiniz.

Nesneye sırayla animasyon türlerini uygulayarak deneyiniz.

```
import fl.transitions.*;
import fl.transitions.easing.*;

TransitionManager.start(kare,//kare nesnesi için yöneticiyi başlattık
{   //Bu bölümde kare nesnesi için parametre veriyoruz.

type:Blinds,            //animasyonun tipi burada belirlendi

direction:Transition.IN,//animasyonun yönünü belirledik
//in ve out olmak üzere 2 tane

duration:3,         //animasyonun gerçekleşmesi için gerekli zaman

easing:Elastic.easeIn,  //animasyonun türünü belirliyoruz

numStrips:5,            //panjur sayısı belirlendi.

dimension:1     //o değeri yatay 1 değeri dikey panjur oluşturdu.
} );
```

Yukarıdaki panjur animasyonunda kullanılan type, direction, duration, easing parametreleri tüm animasyonlar için ortak parametrelerdir. Bazı animasyon sınıflarının kendine özel parametreleri vardır.

- Döndürme (rotate) animasyonu için:
  - *ccw:true* parametresi saat yönünde döndürme
  - *ccw:false* parametresi saat yönünün tersi döndürme
  - **degrees:** 0-9999 arası bir değer verilebilir kaç derece açıyla döneceği belirlenir.
- Silme (wipe) animasyonu için:
  - **startPoint:**1-9 arası bir değer verilebilir. Hareket yönünü buradan belirleyebilirsiniz.




**Resim 3.9: StartPoint yönleri**

- Uçuş (fly) animasyonu için:
  - **startPoint:** 1-9 arası bir değer verilebilir. Hareket yönünü buradan belirleyebilirsiniz.

- İris (iris) animasyonu için:
  - **startPoint:** 1-9 arası bir değer verilebilir. hareket yönünü buradan belirleyebilirsiniz.
  - *Shape: Irıs.CIRCLE* parametresi daire olarak animasyonu gerçekleştirir.
  - *Shape: Irıs.SQUARE* parametresi kare olarak animasyonu gerçekleştirir.
- Sıkma (squeeze) animasyonu için:
  - **Dimension:** o değeri yatay 1 değeri dikey animasyon yönü belirler.
- Piksel piksel oluşturma-yoketme (pixeldissolve) animasyonu için
  - **xSections:** 1-50 arası bir değer alır yatay hücre sayısını belirler.
  - **ySections:** 1-50 arası bir değer alır dikey hücre sayısını belirler.

Buradaki parametrelerdeki değerleri değiştirerek birbirinden güzel animasyonlar yapabilirsiniz.

### 3.7.2. Hareket Animasyonları

Hareket sınıfını kullanmak biçim sınıflarını kullanmak kadar kolaydır. Öncelikle *tween* ve *easing* sınıflarını flash içerisine import etmek gereklidir. Hareket animasyonları nesnelerin özellikleri kullanılarak yapılır.

x, y, width, height, scaleX, scaleY, rotation, alpha bunlara örnektir. Animasyon bu özelliklere göre yönlendirilebilir. Birden fazla animasyon türü kombine edilerek nesneye değişik animasyonlaar uygulanabilir.

Temel kullanım şablonu aşaıdaki gibidir;

```
1  İmport fl.transitions.tween;
2  İmport fl.transitions.easing.*;
3
4  var hareket ismi:tween=new Tween(hedef,"metot",easing türü,
5
6  ilk konum,son konum, animasyon süresi, zaman metodu);
```

- Hareket animasyonlarında başlangıç ve bitiş değeri özelliğe göre değişecektir. Örneğin x ve y metotlarına göre yapılan animasyonda başlangıç ve bitiş değeri, sahnedeki konuma göre olacaktır.
- Alpha için kullanılan değerler ise 0 ve 1 arasında olacaktır.
- Scale metodlarında verilen değerler büyüme katsayısı (2 kat, 3 kat vb.) olarak alınacaktır.
- Yükseklik ve genişlik (width, height) metotlarında değer piksel olarak verilmelidir. rotation da başlangıç ve bitiş değeri (0-360) derece cinsinden olmalıdır.

Sahne üzerinde birden fazla animasyon yapmak için:

- Yeni bir animasyon belgesi açın ve sahneye isminizi yazın.
- Yazdığınız ismi film klibine dönüştürün ve örnek isim olarak isim atayın.
- Sınıfları sahneye import edin.

```
1  import fl.transitions.Tween;
2  import fl.transitions.easing.*;
```

- İsim nesnesine tek tek hareket animasyonlarını tanımlayın. Her kod satırından sonra animasyonu test ederek kontrol edebillirsiniz.

```
1  import fl.transitions.Tween;
2  import fl.transitions.easing.*;
3
4
5  var birinciAnimasyonum:Tween = new Tween(isim,"rotation",Elastic.easeInOut,0,360,5,true);
6  var ikinciAnimasyonum:Tween = new Tween(isim,"alpha",Elastic.easeInOut,1,0.2,5,true);
7  var ucuncuAnimasyonum:Tween = new Tween(isim,"scaleX",Back.easeOut,0,2,5,true);
8  var dorduncuAnimasyonum:Tween = new Tween(isim,"scaleY",Back.easeInOut,0,2,5,true);
9  var besinciAnimasyonum:Tween = new Tween(isim,"witdh",Back.easeInOut,0,2,5,true);
```

## UYGULAMA FAALİYETİ

Animasyon yazılımında olay yöneticileri eklemek fonksiyon tetiklemek ve kod tabanlı animasyonlar yapmak için aşağıdaki uygulamayı yapınız.

| İşlem Basamakları | Öneriler |
|---|---|
| ➢ Yeni bir belge açınız. | ➢ Karşılama ekranını veya Dosya > Yeni (CTRL + N) komutunu kullanabilirsiniz. |
| ➢ Sahneye daire çizip film klibi haline getirin ,örnek ismi olarak daire verin. | ➢ Özellikler panelinden örnek ismini verebilirsiniz. |
| ➢ Daire nesnesine farenin üstüne gelme durumunu gözleyen bir dinleyici tanımlayın. | ➢ addEventListener komutunu kullarak yapabilirsiniz. |
| ➢ Dinleyicimiz durum fonksiyonunu tetiklesin. | ➢ addEventListener komutunda fonksiyonu tetikleyebilirsiniz. |
| ➢ Faremiz dairenin üzerine geldiğinde çıkış ekranında fare üzerinde yazsın | ➢ Çıkış ekranına trace komutu ile yazdırabilirsiniz. |
| ➢ Diğer fare olaylarını için farklı nesnelere olay dinleyicileri ekleyip her birini test edin. | ➢ Sahneye her olay için farklı nesne tanımlamayı unutmayın.<br>➢ Her olay için farklı fonksiyon tanımlayabilirsiniz. |
| ➢ Sahnemize klavyeden b tuşuna basma olayını gözleyen bir dinleyici oluşturun | ➢ B tuşu için keyCode komutunu kullabilirsiniz. |
| ➢ Kod parçacıkları kullanarak sahneye fare olaylarını ekleyin. | ➢ Sahneye her olay için farklı nesne tanımlamayı unutmayın.<br>➢ Her olay için farklı fonksiyon tanımlayabilirsiniz. |
| ➢ Sahnemize her biçim animasyonu için toplamda 10 adet düğme oluşturun. | ➢ Düğme yerine herhangi bir nesne çizip film klibi haline getirebilirsiniz. |
| ➢ Daire nesnemiz için düğmelerimiz bir biçim animasyonunu çalıştırsın. Parametreleri rastgele değerler vererek oluşturunuz. | ➢ Döndürme animasyonu için ccw,degrees<br>➢ Silme,uçuş,iris için startpoint.<br>➢ Sıkma için dimension<br>➢ Piksel için xSections,Ysections parametrelerini kullanmayı unutmayınız. |
| ➢ Hareket animasyonu için bir düğme ekleyerek ease türlerini inceleyin | ➢ Eylem panelinin otomatik kod tamamlama özelliğini kullanıp ease türlerini ortaya çıkartabilirsiniz. |

## KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadığınız beceriler için **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| **1.** Sahnedeki nesneye olay dinleyicisi ekleyebildinizmi? | | |
| **2.** Olay dinleyicisi ile fonksiyon çalıştırabildinizmi? | | |
| **3.** Fare olaylarını kullanabiliyormusunuz? | | |
| **4.** Klavyeden basılan tuşu uygulamada kullanabiliyor musunuz? | | |
| **5.** Kod parçacıkları panelini kullanabildiniz mi? | | |
| **6.** Biçim animasyonlarının parametrelerini biliyor musunuz? | | |
| **7.** Nesneye biçim animasyonu yaptırabilir misiniz? | | |
| **8.** Nesneye hareket animasyonu yaptırabilir misiniz? | | |
| **9.** Hareket animasyonu yaptırırken nesne parametrelerini biliyor musun? | | |
| **10.** Birden fazla animasyonu kombine edip kullanabilir misiniz? | | |

## 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.** (  ) KEY_UP ve KEY_DOWN klavye tuşları ile ilgili olayları gözler.

**2.** (  ) MOUSE_CLICK ve MOUSE_DOWN aynı işlemi yapar

**3.** (  ) ENTER_FRAME olayı sonucu tetiklenen fonksiyonun çalışma sayısı sahne hızına göre belirlenir.

**4.** (  ) Kod parçacıklarını kullanırken nesnelere eklenen kodlar değiştirilemez.

**5.** (  ) Kod yazılarak nesnelere animasyon yaptırabiliriz.

**6.** (  ) Çarpma(Crash) biçim animasyonudur.

**7.** (  ) Hız (Speed) parametresi ile nesneler hızlandırılabilir.

**8.** (  ) Startpoint hareketin yönünü tayin eder.

**9.** (  ) Dimension parametresinin 2 değeri vardır.

**10.** (  ) 10 adet hareket animasyonu aynı anda yapılamaz.

## 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 faaliyetine geçiniz.

# ÖĞRENME FAALİYETİ–4

## AMAÇ

Animasyon yazılımını kullanarak Sınıf (class) işlemlerini gerçekleştirebilirsiniz.

## ARAŞTIRMA

- Nesneye yönelik programlama nedir, araştırınız.

# 4. SINIFLAR (CLASS)

## 4.1. Nesne Tabanlı Programlama

Animasyon yazılımı ActionScript 3.0 ile birlikte çok güçlü bir yazılım geliştirme aracı haline gelmiştir. Yazılım geliştiren kişiler için iki tür yazılım geliştirme sistemi bulunmaktadır. Bunlardan birincisi yordam tabanlı yani kodları sahneye yazıp derleme, ikincisi ise nesne tabanlı programlama yapmaktır.

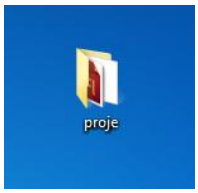
Nesne tabanlı yazılım geliştirme basit olarak animasyon yazılımı dışında kod yazıp uygulamaya dâhil etmek ve kullanmak olarak düşünülebilir. Uygulama için gerekli fonksiyonlar dışarıdaki kod sayfasında toplanıp daha sonra bunlar yazılıma dâhil edilebilir. OOP programlamanın birçok avantajı vardır. Örneğin uygulama daha modüler bir sisteme sahip olacaktır. Kodlar başka sistemlerde rahatlıkla kullanılabilecektir. Bir proje birçok alt alana ayrılarak, birçok kişinin bu alt alanlarda çalışması sağlanabilir.

Nesne tabanlı programlamada öncelikle paket (package) ve sınıf (class) tanımlaması yapılamalıdır. Sınıf tanımlaması yapıldıktan sonra içerisine değişkenler ve fonksiyonlar tanımlanabilir.

## 4.2. Paket Tanımlama

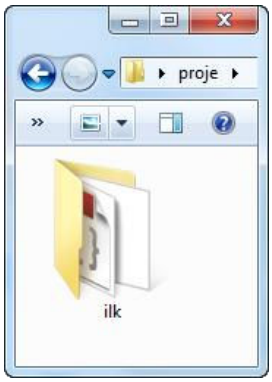
Paket tanımlaması yapmak için:

- Masaüstünüzde, dosyalarınızı düzenli bir şekilde saklamak için proje isimli bir klasör oluşturun.




**Resim 4.1: Masaüstünde oluşturulan proje isimli klasör**

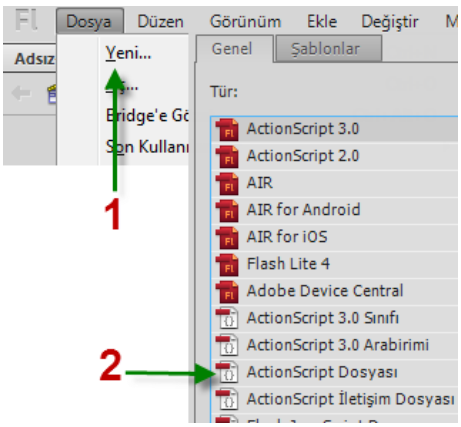
- Bu klasörün içine ilk diye bir klasör daha oluşturun.




**Resim 4.2: Proje klasörünün içindeki ilk isimli klasör**

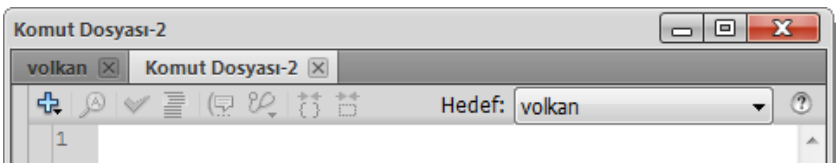
- Animasyon yazılımında yeni bir çalışma sahnesi oluşturun ve proje klasörünün içerisine kendi isminizle kaydedin.

- Dosya > Yeni > ActionScript Dosyası komutnu seçin.



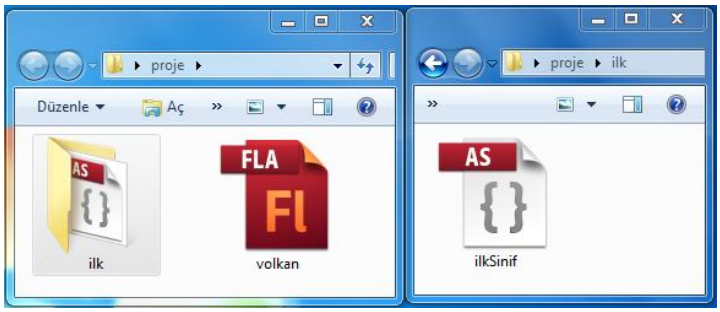

**Resim 4.3: Yeni ActionScript dosyası açma penceresi**

- Animasyon yazılımında 2 adet dosya oluştudunuz ve komut dosyasının hedefinde kendi isminizi görmüş olmamız gerekiyor.




**Resim 4.4: Çalışma dosyalarının program içerisindeki görünümü**

- Çalışma dosyasını kendi isminizle proje klasörüne kaydetmiştiniz. ActionScript dosyasını *ilkSinif.as* şeklinde *ilk* klasörü içerisine kaydedin.

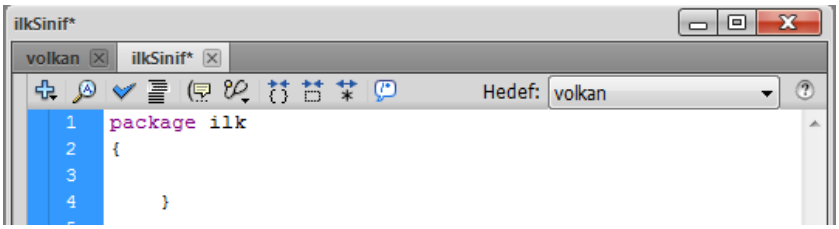



**Resim 4.5: Masaüstündeki klasörlerin yapısı**

- Paket oluşturma aslında sınıfların klasördeki yerini belirten adres deyimleridir. Kaydedilen *.as uzantılı dosyanın fla dosyasına göre nerede olduğunu gösterir.

- Unutmamamız gereken ActionScript dilinin klasörleri nokta ile ayırdığıdır. Yani *ilk* klasörünün içinde *deneme* isimli bir klasör daha olsaydı. Adres yazarken *ilk.deneme* şeklinde yazılması gerekir.

- Artık paket tanımlama yapabiliriz.



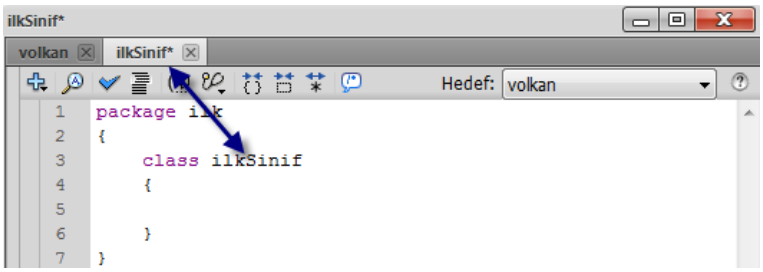

**Resim 4.6: İlk isminde paket tanımlaması**

- Bu işlemle paketleme işlemi tamamlanmış oldu.*.as dosyası *.fla dosyasına göre *ilk* klasörünün içinde olduğu için *ilk* olarak yazılmıştır. Eğer *.fla dosyası ve *.as dosyası aynı yerde olursa *package* komutunun yanına hiçbir şey yazılmaması gerekir.

## 4.3. Sınıf (Class) Tanımlama

Sınıf tanımlamak için:

- *Class* deyiminden sonra class (sınıf) için isim verilmesi gerekir. Sınıf ismi ile oluşturulan *.as dosyasının ismi aynı olmalıdır. Aksi halde sahnede kullanırken bazı problemlerle karşılaşılmaktadır. Bir paket içerisine birden fazla sınıf tanımlanabilir.




**Resim 4.7: ilkSinif class'ının yazılması**

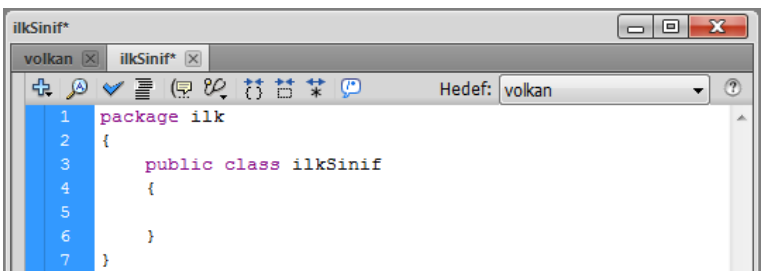
- Sınıf içerisine çeşitli komutlar yazılabilir. Fakat burada yetkilendirme sorunları başlayacaktır. Sınıfın kullanılacağı, nasıl yetkilendirme yapılacağının ayarlanması gerekmektedir.

- Yetkilendirme için public , internal, final, dynamic nitelikleri kullanılmalıdır.

Yetkilendirme yapmak için:

- Yukarıdaki yazım şeklinde yetkilendirme ön tanımlı olarak *internal* olacaktır. Bu nitelik içinde bulunduğu klasörden başka yerde kullanılamayacağı anlamına gelmektedir.

- Class'ın önüne *public* niteliğini getirilirse artık bu sınıf her yerden çağrılabilir ve kullanılabilir anlamına gelmektedir.




**Resim 4.8: Public tanımlamasının yapılması**

Her sınıf için tanımlayıcı bir fonksiyon bulunmak zorundadır. Tanımlayıcı fonksiyon sınıf çalıştığı anda devreye girer. Eğer tanımlayıcı fonksiyonu oluşturulmazsa animasyon yazılımı otomatik olarak boş bir tane tanımlayacaktır. Tanımlayıcı fonksiyon ismi, sınıf ismi ile aynı olmak zorundadır.

```
1  package ilk
2  {
3      public class ilkSinif
4      {
5          public function ilkSinif():void
6          {
7  //tanımlayıcı fonksiyon
8          }
```

Sınıf içerisinde tanımlanan her değişken sınıf nesnesinin **özelliği**, her fonksiyonda sınıf nesnesinin **metodu** olacaktır.

Fonksiyonlar için de yetkilendirmenin yapılması gereklidir. Fonksiyonlar için *public* veya *private* deyimleri kullanılır. *Public* her yerden erişelebilir, *private* sadece aynı sınıfta paylaşılabilmektedir.

Değişken tanımla yapılırken aynı şekilde yetkilendirme yapılmalıdır. Değişkenleri fonksiyonlar içerisinde tanımlanan herhangi bir yetkilendirme yapılmaz. Normal şekilde tanımlanıp kullanılabilir. Burada yetkilendirmeyi *yerel* ve *global* değişkene benzetilebilir.

Hatırlanacak olursa yerel değişkenler tanımlandıkları yerde etkili olup blok parantezlerinini dışında bellekten düşmektedir. *Public* ve *private* durumuda buna benzetilebilir. Fonksiyonlarda olduğu gibi *public* tanımlı değişkenler global değişkenler olurken *private* tanımlı değişkenler yerel değişkenler olmaktadır.

ActionScript dosyamıza geri dönüp tanımlanan paket ve sınıfa, değişken ve fonksiyon ekleyin.

```
1  package ilk
2  {
3      public class ilkSinif
4      {
5          public function ilkSinif():void
6          {
7  //tanımlayıcı fonksiyon
8          }
9          public function selam():void
10         {
11 trace("merhaba");
12 }
13 public var selamla:String="MERHABA";
14     }
15 }
```

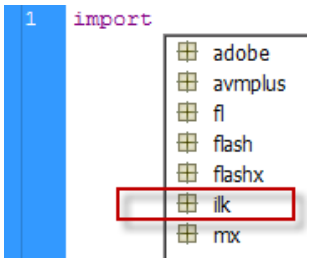
Sınıf içerisinde *ilkSinif* isminde tanımlayıcı fonksiyon, *selam* isminde bir fonksiyon daha ve *selamla* isminde bir string değişken tanımlamıştır.

## 4.4. Sınıfları Sahnede Kullanma

Sınıfı çalışma sahnesinde kullanmak için:

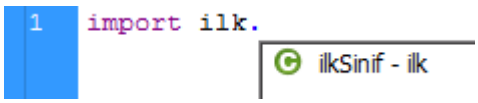
- Kendi isminizle kaydettiğiniz fla dosyamızı açıp , eylemler panelini F9 tuşu ile açın.
- Sınıfı sahneye dahil etmek için *import* komutu kullanılmaktadır. Animasyon yazılımı içerisinde bir çok sınıf bulunmaktadır. Bunların bazıları otomatik olarak çalışma sahnesine dahil olurlar, kendi yazdığınız ve bazı sınıfları ise uygulama geliştirme aşamasında *import* yöntemi ile dahil edilmesi gerekir.

- *import* komutunu yazıp boşluk bırakıldığında ActionScript dosyasının yerini belirten klasör ismi yani *ilk* otomatik olarak çıkacaktır.




**Resim 4.9: Sınıf seçme penceresi**

- *İlk* seçildikten sonra karşınıza *ilk* klasörünün içindeki *ilkSinif* gelecektir. Panel burada kolaylık yapıp sahneye sınıfımızı dahil etmenizi sağlayacaktır.




**Resim 4.10: Sınıf seçme penceresi**

```
1  import ilk.ilkSinif;
```

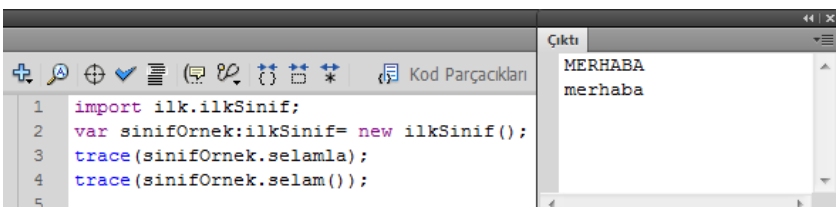
- Örnek oluşturularak sınıflar kullnılabilir. Örnek oluşturma işlemi hazır olan sınıfın bir kopyasını alıp onu çalışma sahnesinde kullanılmasıdır. Orijinal sınıf aynen kalır yani korunur. Tüm işlemler örnek üzerinden yapılır. Örnek çıkartılan nesne orijinali ile aynı özellik ve metotlara sahiptir.

Örnek oluşturma işlemi için:

- *var sinifOrnek:ilkSinif= new ilkSinif();* tanımlaması yapılır.

```
1  import ilk.ilkSinif;
2  var sinifOrnek:ilkSinif= new ilkSinif();
```

- *sinifOrnek* nesnesi artık *ilkSinif* sınıfının tüm özelliklerini almış durumdadır. Sınıfın içerisnde tanımlanan selam metodu ve selamla özelliğini kullanarak yaptığımız işlemlerin doğruluğu kontrol edin.




**Resim 4.11: Nesneyi test eden kod**

## 4.5. Varolan Sınıfları Genişletme

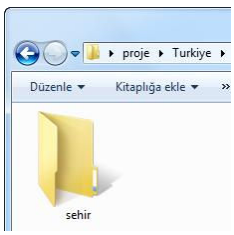
Hazırlanan sınıflar birbiri içerisinde kullanılabilir. Bunun için *extends* metodu kullanılır.

**Örnek:**

Türkiye'nin illerinin özelliklerini anlatan bir proje için ortak olan alanları birbirlerinden alan bir uygulama yapın. Amasya ilindeki kurumlar metodunu Isparta içerisinde kullanılacaktır.

- Proje dosyasının içerisine *Türkiye* adlı bir klasör açın. Bu klasör içersine içerisine *sehir* diye başka bir klasör daha açın.

- Animasyon yazılımı içerisinde açılan yeni sahneyi *Türkiye* klasörü içerisinde *sehirler* ismi ile kaydedin.




**Resim 4.12: Masaüstündeki klasörlerin yapısı**

- İki adet ActionScript dosyası açarak birini *Isparta* diğerini *Amasya* olarak sehir klasörüne kaydedin.

- Amasya dosyası içerisine kurumlar fonksiyonunu oluşturun.

```
1  package sehir
2  {
3  public class Amasya
4  {
5  public function kurumlar():void
6  {
7  trace("Valilik,Belediye,Milli Eğitim Müdürlüğü");
8  } } }
```

- Isparta dosyamızın içerisinde iklim ,bitki, tarim fonksiyonlarını oluşturun.

```
1  package sehir
2  {
3  public class Isparta extends Amasya
4
5  {
6      public function iklimIsparta():void
7  {
8  trace("yazları sıcak ve kurak, kışları az yağışlı ve soğuktur");
9  }
10
11 public function bitkiIsparta():void
12 {
13 trace("orman,fundalık,çayır ve mer'alar dan oluşmaktadır.");
14 }
15
16 public function tarimIsparta():void
17 {
18 trace("ekime elverişli arazilerin çoğunu gül bahçeleri oluşturur.");
19 }
20
21 }
22 }
```

- Yukarıdaki *extends* komutu ile *Amasy*a ilinin kurum isimlerini *Isparta* içerisinde kullanılmıştır.

Çalışma sahnesine *Isparta* sınıfı dâhil edildiğinde *Amasya* sınıfının içerisindeki metot ve sınıflarda Isparta içerisine dâhil olacaktır.

```
1  import sehir.isparta;
2  var goster:isparta=new isparta();
3
4  trace(goster.kurumlar());
5  //çıkış panelinde " Valilik,Belediye,
6  //Milli Eğitim Müdürlüğü " çıktısını verecektir.
```

*Extends* kullanırken sadece iki sınıf birbirine dâhil edilebilir. Daha fazla sınıfı dâhil etmek için *import* kullanılması gerekir.

## UYGULAMA FAALİYETİ

Sınıf nesnesi oluşturmak özellik ve metod tanımlamak başka sınıflara genişletmek için aşağıdaki uygulamayı yapınız.

| İşlem Basamakları | Öneriler |
|---|---|
| ➢ Yeni bir ActionScript dosyası açınız. | ➢ Karşılama ekranını veya Dosya > Yeni > ActionScript komutunu kullanabilirsiniz. |
| ➢ Aynı yerde olan *.fla ve *.as dosyası için paket tanımlaması yapınız. | ➢ Package komutunu kullanabilirsiniz.<br>➢ Aynı klasörde olduğu için yer tanımı yapmaya gerek olmayacaktır. |
| ➢ Merhaba ismi ile sınıfımızı tanımlayınız. | ➢ Sınıf ile as dosyasının ismi aynı olursa problem çıkmaz. |
| ➢ Sınıfın yetkilendirmesini yapınız. | ➢ Public,internal en çok kullanılanlarıdır. |
| ➢ Selam çıktısını veren metod ve özellik tanımlayınız. | ➢ Metodların aslında fonksiyon özellikler ise değişkendir |
| ➢ Tanımlayıcı fonksiyon oluşturunuz. | ➢ Tanımlayıcı fonksiyon sınıfla aynı isimde olmalıdır. |
| ➢ Sınıfı sahnede kullanılması için sahneye dahil ediniz. | ➢ İmport komutunu kullanmalısınız. |
| ➢ Sınıfın örneğini çıkartınız. | ➢ Örneğini çıkarttığımız nesne orijinal ile aynı özellik ve metotlara sahiptir. |
| ➢ İkinci bir sınıf oluşturup ilk sınıfımızdaki metod ve özellikleri ikinciye dâhil ediniz. | ➢ Extends komutunu dâhil etmekte kullanabilirsiniz. |

## KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadığınız beceriler için **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

| Değerlendirme Ölçütleri | Evet | Hayır |
|---|---|---|
| **1.** Paket tanımlama yapabildiniz mi? | | |
| **2.** Sınıf tanımlama yapabildiniz mi? | | |
| **3.** Sınıf içerisinde özellik oluşturabildiniz mi? | | |
| **4.** Sınıfın metodlarını tanımlayabildiniz mi? | | |
| **5.** Sınıfı sahnede kullanabildiniz mi? | | |
| **6.** Sınıfı genişletebildiniz 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.** ( ) Nesne tabanlı programlamada paket ve sınıf tanımlaması yapılması zorunludur.

**2.** ( ) Paket tanımlaması yapılırken ActionScript dosyasının nerede olduğu belirtilmelidir.

**3.** ( ) Paket oluşturulurken package komutu kullanılır.

**4.** ( ) Sınıf oluşturduğumuzda yetkilendirme yapmazsak otomatik olarak public olacaktır.

**5.** ( ) Sınıf içerisinde tanımlanan her fonksiyon sınıfın özelliğidir.

**6.** ( ) Sınıfları sahnede kullanmak için export komutu kullanılmallıdır.

**7.** ( ) Örnek oluşturma işlemi sınıfın aynısından bir tane daha oluşturmak demektir.

**8.** ( ) Örneği oluşturulan nesnenin orjinali silinir.

**9.** ( ) Sınıfları genişletmek için import kullanılır.

**10.** ( ) Extend komutu ile 5 sınıfa kadar genişletilebilir.

## 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 faaliyetine geçiniz.

# ÖĞRENME FAALİYETİ–5

## AMAÇ

Animasyon yazılımını kullanarak karar yapılarını kullanabileceksiniz.

## ARAŞTIRMA

- Günlük hayatta herhangi bir eylemi gerçekleştirmeye nasıl karar veririz. 5 eylem için nedenleri ile birlikte yazınız.

# 5. KARAR YAPILARI

## 5.1. Karar İfadeleri

Karar ifadeleri, programın akışını denetlemeye yarar. Belli bir şart sonucu programın istediğimiz şekilde çalışmasını sağlayan sistemlerdir.

Hemen hemen tüm dillerde aynı yapılar bulunmaktadır. ActionScript 3.0 programlama dilinde if, if-else, else if, switch olmak üzere 4 tanedir.

### 5.1.1. if ifadesi

Şartın doğru olup olmadığını denetler şart doğruysa parantezler arasındaki kodları çalıştırır.

```
1  if(koşul)
2  {
3  //şart doğruysa yapılacak işler
4  }
```

Normal parantezler arasındaki ifade true değerini döndürüyorsa, yani koşul ifadesi doğru ise süslü parantezler arasındaki kodlar çalışacaktır.

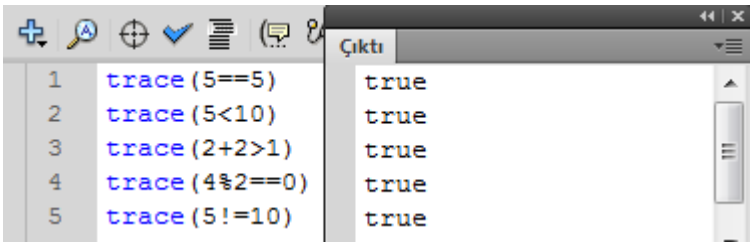
```
1  if(true)
2  {
3  trace("if kodu çalıştı");
4  }
```

Çıktı

```
if kodu çalıştı
```

**Resim 5.1: İf koşul ifadesi**

Yukarıdaki örnekte parantezler arasında direk olarak true yazıp komutların çalıştığı görülmektedir. Önemli olan oraya yazılan ifadenin doğru (true) değerini vermesidir.




**Resim 5.2: Koşullar**

Resim 5.2'deki ifadelerinin her biri true değer döndürecektir.

### 5.1.2. if-else ifadesi

if else ifadesi if gibi çalışır. Tek farkı şartın doğru olmaması durumunda çalıştırılacak başka komutlar eklenebilir.

Kullanımı şu şekildedir.

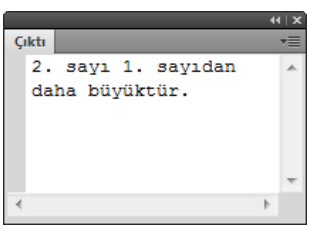
```
1  if(koşul)
2  {
3  //Şart doğruysa çalıştırılacak komutlar.
4  }
5  else
6  {
7  //Şart yanlışsa çalıştırılacak komutlar.
8  }
```

İki sayıdan büyük olanını bulan programı yapmak için:

- Yeni bir çalışma sahnesi açın. F9 fonksiyon tuşuna basarak aşağıdaki kodları yazınız.

```
1  var sayi1:uint=5;
2  var sayi2:uint=10;
3  if(sayi1>sayi2)
4  {
5  trace("1. sayı 2. sayıdan daha büyüktür.")
6  }else
7  {
8  trace("2. sayı 1. sayıdan daha büyüktür.")
9  }
```

- CTRL + Enter tuşları ile uygulamayı test ettiğimizde çıktı ekranında şart sağlanmadığı için 2. Sayı 1. Sayıdan daha büyüktür ifadesi yazacaktır.




**Resim 5.3: Program çıktısı**

### 5.1.3. if-else-if ifadesi

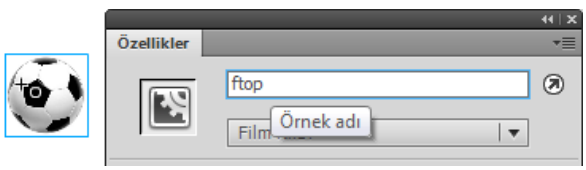
İf-else ifadesinin çoklu kullanımı gibidir. Birden fazla şart kontrol edileceğinde kullanılır.

Kullanımı şu şekildedir.

```
if(koşul 1)
{
//Şart doğruysa çalıştırılacak komutlar.
}
else if(koşul 2)
{
//Şart doğruysa çalıştırılacak komutlar.
}
else if(koşul 3)
{
//Şart doğruysa çalıştırılacak komutlar.
}
else
{
//şartlar yanlışsa çalışacak komut
}
```

**Örnek:** Birden fazla şartın kullanıldığı fareden kaçan futbol topu animasyonu yapınız.

- Yeni bir dosya açın. Sahne büyüklüğünü 600x600 px yapınız.
- Dosya > İçe Aktar kullanarak futbol topu resmini sahneye aktarınız.
- Sahneye eklediğimiz resmi film klibi haline getirin. örnek ismi olarak *ftop* değerini veriniz.




**Resim 5.4: Futbol topu örnek ismi**

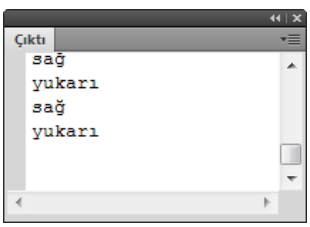
- Sahneye aşağıdaki kodları ekleyiniz.

```
stage.addEventListener(Event.ENTER_FRAME,gez);
function gez(event:Event):void
{
    if (mouseX>300)
    {ftop.x-=2;}
    if (mouseX<300)
    {ftop.x+=2;}
    if (mouseY<300)
    {ftop.y+=2}
    if (mouseY>300)
    {ftop.y-=2}
}
```

- 3. Satırda sahnemizi gözleyen ve gez fonksiyonunu tetikleyen bir olay dinleyicisi eklenmiştir.
- Gez fonksiyonu içerisinde 4 tane şart oluşturuldu.
- mouseX ve mouseY metotları sahne üzerinde farenin yerini koordinat olarak tesbit edip if kontrol deyimi ile farenin sahnenin sağ, sol, yukarı ya da aşağı tarafta olması şartını kontrol etmektedir.
- Fonksiyon çalıştığında 4 karar yapısıda tekrar tekrar çalıştırılmakta. Ve şartlara uyan if komutları altındaki çalıştırılacak kodlar aynı anda çalıştırılmaktadır.
- Farenin durumuna göre hangi if komutunun çalıştığını gözlemlemek için trace komutunu kullanabiliriz. Şartların altına trace komutunu ekliyoruz.

```
1  stage.addEventListener(Event.ENTER_FRAME,gez);
2  function gez(event:Event):void
3  {
4      if (mouseX>300)
5      {ftop.x-=2;
6      trace("sağ");
7      }
8      if (mouseX<300)
9      {ftop.x+=2;
10     trace("sol");
11     }
12     if (mouseY<300)
13     {ftop.y+=2
14     trace("yukarı");}
15     if (mouseY>300)
16     {ftop.y-=2
17     trace("aşağı");
18     }
19 }
```

- CTRL+Enter yapılıp kodları çalıştırıldığında çıkış ekranında hangi kodlar çalışıyor gözlemleyebilirsiniz.




**Resim 5.5: Animasyonun çıkış ekranı**

- ftop.x ve ftop.y futbol topunun sahnedeki koordinatlarını değiştirerek şartlara göre nesnenin fare hareketinin tersi yönünde hareketlenmesini sağlamaktadır.

### 5.1.4. Switch- Case

Birden fazla koşul için gerekli kontrolü yapar. Fakat şartın boolean bir ifade döndürmesine bakmaz, değer kısmına yazılan ile kontrol ifadesinin aynı olup olmaması kontrolünü sağlar. Kullanımı ve yapısı çok basittir. İstenilen kadar kontrol ifadesi eklenebilir.

```
switch (değer)
{
case kontrol ifadesi -1:
değer ifadeye eşitse çalıştırılacak komutlar
break;
case kontrol ifadesi -2:
değer ifadeye eşitse çalıştırılacak komutlar
break;
case kontrol ifadesi -3:
değer ifadeye eşitse çalıştırılacak komutlar
break;
.
.
.
default:
değer hiçbir ifadeye eşit değilse çalıştırılacak
kodlar.
break;
}
```

**Örnek:**

Klavyeden basılan 1-5 arası sayıları yazı ile çıkış ekranına yazdıran uygulama yapmak için:

- Sahnemize klavye olaylarını gözleyen bir olay dinleyicisi ekliyoruz. Olay dinleyicimiz sayilar fonksiyonunu tetiklesin.

```
stage.addEventListener(KeyboardEvent.KEY_DOWN,sayilar)
```

- Klavyeden basılan tuşların değerini, deger isimli bir değişkene aktaran fonksiyonumuzu yazınız.

```
stage.addEventListener(KeyboardEvent.KEY_DOWN,sayilar)
function sayilar(event:KeyboardEvent):void
{
var deger:uint=event.keyCode;
trace(deger);
}
```

- Switch deyimi ile, ekrana istediğimiz ifadeyi yazdırmanızı sağlayan kodlar aşağıdaki gibidir.

```
stage.addEventListener(KeyboardEvent.KEY_DOWN,sayilar)
function sayilar(event:KeyboardEvent):void
{
var deger:uint=event.keyCode;
switch (deger){
case 49:
trace("bir")
break;
case 50:
trace("iki")
break;
case 51:
trace("üç")
break;
case 52:
trace("dört")
break;
case 53:
trace("beş")
break;
default:
trace("1-5 arası sayılar için çalışır")
break;
}}
```

- Klavyeden basılan tuşu, deger değişkenine aktarıp switch yapısı ile 1 den 5 e kadar herhangi bir sayı tuşuna basıldığında, yazıyla karşılığı ekrana yazdırılacaktır.

- Event.keyCode ifadesi basılan tuşun asci karşığını aldığı için sayıların asci sistemdeki karşılıklarını karşılaştırdık.

- 1-5 dışında bir tuşa basıldığında default yapısı çalışıp 1-5 arası sayılar için çalışır ifadesini yazdıracaktır.

## 5.2. Döngüler

Bir işlemi, istediğimiz sayıda bilgisayara yaptırmak için kullandığımız kodlara döngü denir. Aynı işlemi defalarca tekrar yazmak yerine, döngü kurup uygulamanın otomatik olarak işlemi tekrar etmesi sağlanabilir. Durumlara uygun döngüler kurmak gerekir.

### 5.2.1. For döngüsü

Tekrar sayısını bilinen durumlarda for döngüsünü kullanılır. Örneğin bir işlemi 100 kere tekrar ettirmek için for döngüsünü kullanmak mantıklı olacaktır. Döngü değişkeninin başlangıç değerinden itibaren değişken artış miktarı kadar değerini değiştirir. Koşul doğru olduğu sürece döngü çalışır.

Kodların kullanımı şu şekildedir:

```
1  for(başlangıç değeri;koşul;artış miktarı)
2  {
3  Çalıştırılacak kodlar;
4  }
```

Örneğin, çıkış ekranına 100 kere Isparta yazdıran bir uygulama için for döngüsü kullanılabilir.

```
1  var i:uint;
2  for (i=0;i<100;i++)
3  {
4  trace("ısparta");
5  }
```

Döngüdeki i değişkeni 0'dan başlayacak ve parantezlerin içindeki kod ya da kodların her çalışması sonucunda i değeri 1 artacaktır.

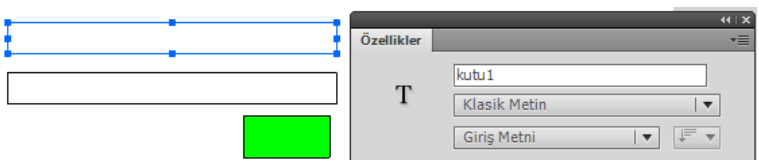
Şart her döngü sonunda kontrol edilecek, i değerinin 100 sayısından küçük olduğu sürece döngüm çalışacaktır.

**Örnek:**

Ekrandan girilen ifadeyi yine ekran üzerinden girilen sayı kadar çıkış ekranında yazdıran uygulama yapmak için:

- Sahnemize 2 adet metin kutusu yerleştirin. Metin kutularını özelliklerinden giriş metni olarak belirleyin. Metin kutularından birisi yazdırılacak ifadeyi diğeri ise kaç defa yazdırılacağını belirleyecektir.

- Son olarak bir adet düğme ekleyip örnek isimlerini kutu1, kutu2, dugme olarak belirleyiniz.




**Resim 5.6: Uygulama ekranı**

- Aşağıdaki kodları yazınız.

```
1  dugme.addEventListener(MouseEvent.CLICK, calistir);
2
3  function calistir(event:MouseEvent):void
4  {
5  var say:uint=Number(kutu1.text);
6  var ifade:String=kutu2.text;
7
8  for(var i:uint=0;i<say;i++)
9  {
10 trace(ifade);
11 }
12 }
```

- 5 Satırdaki Number( ) fonksiyonu sayı türünde olmayan veri tiplerini sayıya dönüştürmeyi sağlar metin kutularından giriş yapılan ifadeler uygulama içerisine string olarak aktarılır. Bu yüzden ilk ifadeyi döngümüzde kullanacağımız için sayısal ifadeye dönüştürür.

### 5.2.2. While döngüsü

Döngünün çalışma sayısını bilmediğimiz durumlarda kullanırız. if ve for döngüsünün birleşimi gibidir. Parantez içerisindeki şart doğru olduğu sürece döngü çalışacaktır.

```
1  var i:uint=4;
2  while (i<10)
3  {
4  trace(i);
5  }
```

Yukarıdaki kodlarda şartın değeri hep 4 olacaktır. Dolayısıyla şart hiçbir zaman gerçekleşmeyecek ve döngü sonsuz defa çalışmak isteyecektir. Böyle durumlarda animasyon yazılımı 15 saniye bekler ve döngüyü sonlandırır. For döngüsü gibi döngü değişkenini arttıran bir komut olmadığından döngü değişkeni hep aynı kalacaktır. Bu durumdan kurtulmak için döngü değişkeni artırılmalıdır.

*i++;* ya da *i=i+1;* şeklinde eklenen komutla artış miktarını 1 olarak belirler ve döngünün 6 kere çalışmasını sağlarız. Artış miktarını istediğiniz gibi değiştirebilirsiniz.

```
1  var i:uint=4;
2  while (i<10)
3  {
4  trace(i);
5  i++;
6  }
```

## UYGULAMA FAALİYETİ

Karar yapılarının kullanıldığı sayı tahmin oyunu hazırlamak için aşağıdaki uygulamayı yapınız.

| İşlem Basamakları | Öneriler |
|---|---|
| ➢ Yeni bir belge açınız. | ➢ Karşılama ekranını veya Dosya > Yeni (CTRL + N) komutunu kullanabilirsiniz. |
| ➢ Sahneye bir adet metin kutusu ekleyip özelliğini giriş metni olarak ayalarlayın. | ➢ Giriş metin kutularına border koyabilirsiniz. |
| ➢ Bir adet kare çizimi yapın ve film klibine çevirin. | ➢ Kareyi seçip F8 tuşunu kullanabilirsiniz. |
| ➢ Metin kutusunun örnek ismi giriş, karenin örnek ismi düğme olsun. | ➢ Örnek ismini belirlemeden nesneleri seçmelisiniz. |
| ➢ Sahneye tıklayıp eylemler panelini seçin | ➢ F9 fonksiyon tuşunu kullanabilirsiniz. |
| ➢ Oyundaki bulunması gereken sayıyı belirtmek için bir değişken tanımlayın. Değişken ismi sayi veri türü uint değeri 40 olsun. | ➢ Değişken tanımlamak için var komutunu kullanmalısın. |
| ➢ Oyuncumuzun 5 defa tahmin yapabildiği hak değişkeni tanımlayalım başlangıç değeri 0 olsun. | ➢ Atama operatörü ile başlangıç değerini 0 yapabilirsin. |
| ➢ Düğmemize olay dinleyicisi tanımlayalım fare tıklanması olayını gözlesin ve oyun fonksiyonunu tetiklesin. | ➢ MOUSE_CLICK olayını kullanabilirsiniz. |
| ➢ Oyun fonksiyonda 3 adet if-else if yapısı kullanalım. | ➢ Sayının girilen sayıdan büyük olma durumunu sonra küçük olma durumunu en son olarak eşit olma durumlarını karşılaştırmalısın.<br>➢ İf(Number(giris.text)>sayi) komutunu örnek alabilirsin. |
| ➢ Hak değişkeni fonksiyon her çalıştığında 1 artsın. | ➢ ++ operatörü ile 1 arttırma yapabilirsin. |
| ➢ Hak değişkeni 5 değerine ulaştığında sahnedeki metin kutusunda hakkın bitti yazsın. | ➢ İf komutu kullanarak işlemi yapabilirsin<br>➢ Sahnedeki metin kutusunu değiştirmek için<br>➢ Giriş.text=”hakkın bitti”; komutunu kullanabilirsin. |
| ➢ Çizimlerin koordinatlarını değiştirin. | ➢ Çizimi seçerek özellikler panelinde konum ve boyut alanını kullanabilirsiniz. |
| ➢ Uygulamayı test ederek görüntüleyin. | ➢ Kontrol et > Filmi test Et > Test Et (CTRL + Enter) komutunu kullanabilirsiniz. |

# ÖLÇME VE DEĞERLENDİRME

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

**1.** (  ) if deyimi ile tek şart kontrol edilebilir.

**2.** (  ) if deyiminde şart ifadesi karşılaştırma olmak zorundadur.

**3.** (  ) Şartın doğru olmadığı durumlarda başka kodları çalıştırmak için else komutu kullanılır.

**4.** (  ) birden fazla if deyimi aynı kod bloğu içerisinde çalıştırılabilir.

**5.** (  ) switch-case kontrol ifadesinde değer ile kontrol ifadesi doğru ise şart sağlanmış olur.

**6.** (  ) Program yazarken defalarca tekrar ettiğimiz kodları döngüler ile hızlıca yazdırabiliriz.

**7.** (  ) For ve while deyimleri uygulama içerisinde farklı  işlemi yapar.

**8.** (  ) For döngüsünde başlangıç değeri ,koşul ve artış miktarı olmalıdır.

**9.** (  ) Döngülerde şart ifadeleri mantıklı yazılmazsa program sonsuz döngüye girebilir.

**10.** (  ) Animasyon yazılımı sonsuz döngüye girdiğinde bilgisayarı yeniden başlatmak gerekir.

## 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

**Aşağıda boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.**

**1.(  )** Açıklama ifadeleri kodların akışını değiştirebilir.

**2.(  )** Nesnelerin yol tanımlarını yapabilmek ya da sınıfların özellik ve metodlarına ulaşmak için nokta(.) ifadesi kullanılır.

**3.(  )** Mtegm ve MTEGM aynı değişkenlerdir.

**4.(  )** Veya operatörü koşullardan herhangi birinin gerçekleşmesi durumunu sınar.

**Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.**

**5.** Fonksiyon veri döndürmüyorsa tanımlama sonuna aşağıdakilerden hangisi eklenmelidir?
- **A)** Void
- **B)** Number
- **C)** Boolean
- **D)** String

**6.** Sahnemizde kare isimli nesneyi gözlemek için aşağıdakilerden hangisi kullanılır ?
- **A)** kare.addEvent
- **B)** eventListener_kare();
- **C)** kare.addEventListener();
- **D)** addEventListener.kare();

**7.** Fare'nin bir nesneyi sol tuş ile tıklayıp bırakma olayı aşağıdakilerden hangisidir?
- **A)** Double_ Click
- **B)** Click
- **C)** Mouse_down
- **D)** Mouse_Over

**8.** Klavyeden basılmış bir tuşun bırakılma olayı aşağıdakilerden hangisidir?
- **A)** Key_press
- **B)** Key_down
- **C)** Key_Up
- **D)** Key_Code

**9.** Aşağıdakilerden hangisi biçim animasyonu **değildir**?
- **A)** Zoom
- **B)** Photo
- **C)** Fade
- **D)** Screen

**10.** Aşağıdakilerden hangisi döndürme animasyonunun parametresidir?
- **A)** StartPoint
- **B)** Degrees
- **C)** xSections
- **D)** Shape

**11.** Sınıf dosyalarının uzantısı aşağıdakilerden hangisidir?
- **A)** *.cs
- **B)** *.as
- **C)** *.fla
- **D)** *.swf

**12.** Sınıfları genişletmek için hangi komut kullanılır?
- **A)** ClassLarge
- **B)** Extends
- **C)** BigClass
- **D)** Package

**Aşağıda verilen cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız**

**13.** Bir işlemi istediğimiz sayıda tekrar ettirmek için kullandığımız kodlara…….. denir.

**14.** Nesne tabanlı programlamada ilk olarak ……..ve …….. tanımlaması yapılmalıdır.

**15.** …….. ifadesi ile istediğimiz anda istediğimiz değişkenlerin anlık değerlerini ya da istediğimiz ifadeyi …….. ekranına yazdırabiliriz.

## 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 öğretmeninize başvurunuz.

# CEVAP ANAHTARLARI

## ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

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

## ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

| | |
|---|---|
| **1** | **Doğru** |
| **2** | **Doğru** |
| **3** | **Yanlış** |
| **4** | **Doğru** |
| **5** | **Yanlış** |
| **6** | **Doğru** |
| **7** | **Doğru** |

## ÖĞRENME FAALİYETİ-3'ÜN CEVAP ANAHTARI

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

## ÖĞRENME FAALİYETİ-4'ÜN CEVAP ANAHTARI

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

## ÖĞRENME FAALİYETİ-5'İN CEVAP ANAHTARI

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

## MODÜL DEĞERLENDİRMENİN CEVAP ANAHTARI

| | |
|---|---|
| **1** | **Yanlış** |
| **2** | **Doğru** |
| **3** | **Yanlış** |
| **4** | **Doğru** |
| **5** | **A** |
| **6** | **C** |
| **7** | **B** |
| **8** | **C** |
| **9** | **D** |
| **10** | **B** |
| **11** | **B** |
| **12** | **B** |
| **13** | **DÖNGÜ** |
| **14** | **PAKET,SINIF** |
| **15** | **TRACE,OUTPUT** |

# KAYNAKÇA


- http://help.adobe.com/tr_TR/ActionScript/3.0_ProgrammingAS3/flash_as3_programming.pdf (14.03.2012/20.00)