NESNE TABANLI PROGRAMLAMA

MARDİN ARTUKLU ÜNİVERSİTESİ

HAZIRLAYAN: Öğr.Gör. Hüseyin AHMETOĞLU

4.HAFTA

Koşul İfadeleri

if() Deyimi

Koşul İfadeleri

- Diğer programlama dillerinde olduğu gibi Java'da da kontrol yapıları dilin en önemli parçasını oluşturur.
- Başka bir deyişle, kontrol yapıları olmasaydı bilgisayar programları, bilgi giriş-çıkışı ve bazı hesaplamalar dışında bir işlemi gerçekleştiremezlerdi.
- Bu bölüme kadar verilen programlar bir bakıma düz hat programları olarak görülebilir. Yani şu ana kadar verilen programlarda bir komuttan sonra aradaki komutları atlayarak başka bir noktaya sıçrama söz konusu değildi.
- Bir bilgisayar programında komutlar normal olarak yazılış sırasına göre çalıştırılır.

Koşul İfadeleri

- Örneğin, herhangi bir programlama dilinde yazılmış olan,
  - **Komut1**
  - **Komut2**
  - **Komut3**
  - **……..**
  - **Komutn1**
  - **Komutn**
- şeklindeki bir programda, önce **Komut1** sonra **Komut2** daha sonra **Komut3 ... Komutn1** ve en son da **Komutn** çalıştırılır.
- Programlama problemlerinden birçoğu bu kadar basit değildir.

Koşul İfadeleri

- Programlama dillerinin gücü, aynı işi tekrar tekrar yapabilmelerinden ya da farklı parametre değerleri için değişik işler yapabilmelerinden gelir.
- Kontrol komutları, doğrusal akış sırasını değiştirebilen komutlardır.
- Örneğin **Komut2** bir kontrol komutu olsaydı, bu komut bir koşulu kontrol ederek koşulun sonucuna göre, **Komut3** ,**Komut4** ve **Komut5**'i atlayarak programın icrasını doğrudan **Komut6**'ya gönderebilirdi.
- Bir bilgisayar programında programın kontrolünün aradaki komutlar atlanarak bir komuttan diğerine sıçramasına ***dallanma (branching)*** diyoruz.
- Bu anlamda kontrol komutları dallanmayı gerçekleştiren komutlardır.

Koşul İfadeleri

- Diğer programlama dillerinde olduğu gibi, Java dilinde de esas olarak iki farklı tür kontrol yapısı ya da komutu mevcuttur:
- **Seçme (selection) işlemini gerçekleştiren kontrol komutları:** Bu tip kontrol komutlarında bir ifade kontrol edilerek ifadenin değerine göre çeşitli seçeneklerden bir tanesinde dallanma işlemi gerçekleştirilir: ***if, if else, switch case*** gibi yapılar bu tür komutları oluşturur.
- **Tekrarlama (loop, repetition):** Bu tip kontrol yapılarında bir işlemler grubu bir koşula bağlı olarak belirli sayıda (10 kez, 50 kez vb.) tekrarlı olarak çalıştırılabilir.

if() Deyimi

- **if** deyimi *"ifadeyi denetleyerek, ifadenin değerine göre uygun yollardan bir tanesiyle dallanma işlemini gerçekleştiren"* bir kontrol deyimidir.
- Bu sayede **şartlı dallanma (conditional branching)** adı verilen işlem gerçekleştirilir.
- Şartlı dallanma, bir programlama dili için en temel kontrol yapısıdır.
- Şartlı dallanma işlemiyle bir program, gidişatını daha önceden verdiğiniz yollardan biriyle devam ettirmeye karar verebilir hale gelir. Bir şarta göre bir komut dizisini çalıştırır veya es geçer.

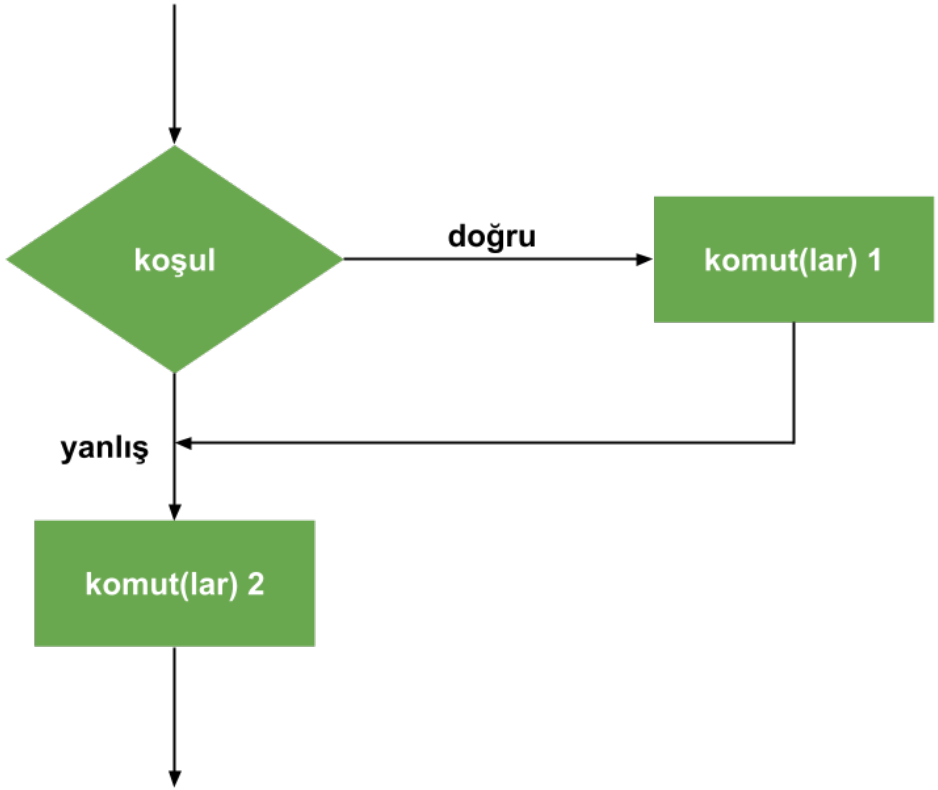
if() Deyimi

- Java dilinde şartlı dallanma ***if*** ve ***else*** anahtar sözcükleriyle gerçekleştirilir.
- *if* deyiminin basit gösterimi şöyledir:

```
if (şart) komut 1
komut 2
```

- Burada şartın değeri **doğru (true)** olarak sağlanıyorsa ancak **komut 1** çalıştırılır. Daha sonra program normal bir akışla **komut 2**'yi çalıştırmaya geçer.
- Şartın değeri **yanlış (false)** ise, bu durumda program doğrudan **komut 2**'ye geçer ve onunla devam eder.
- Kısacası şartın değeri **yanlış** ise **komut 1** hiçbir zaman çalışmayacaktır.

if() Deyimi



if() Deyimi

- Bu noktada, Java dilinde doğru ve yanlışın ne anlama geldiğini belirtelim: Şartın **doğru** olması, o ifadenin matematiksel olarak **1** yani **true** (doğru) olan bir değer üretmesi demektir. **Yanlış** olması ise, matematiksel olarak değerin **0** yani **false** (yanlış) olan bir değer üretmesi anlamına gelir. Programlama dillerinde sadece bu iki değerden birini alabilen değişken türleri *boolean* olarak anılır.

```
if (fiyat==1) {

 //fiyat 1 ise çalıştırılacak komutlar

}
```

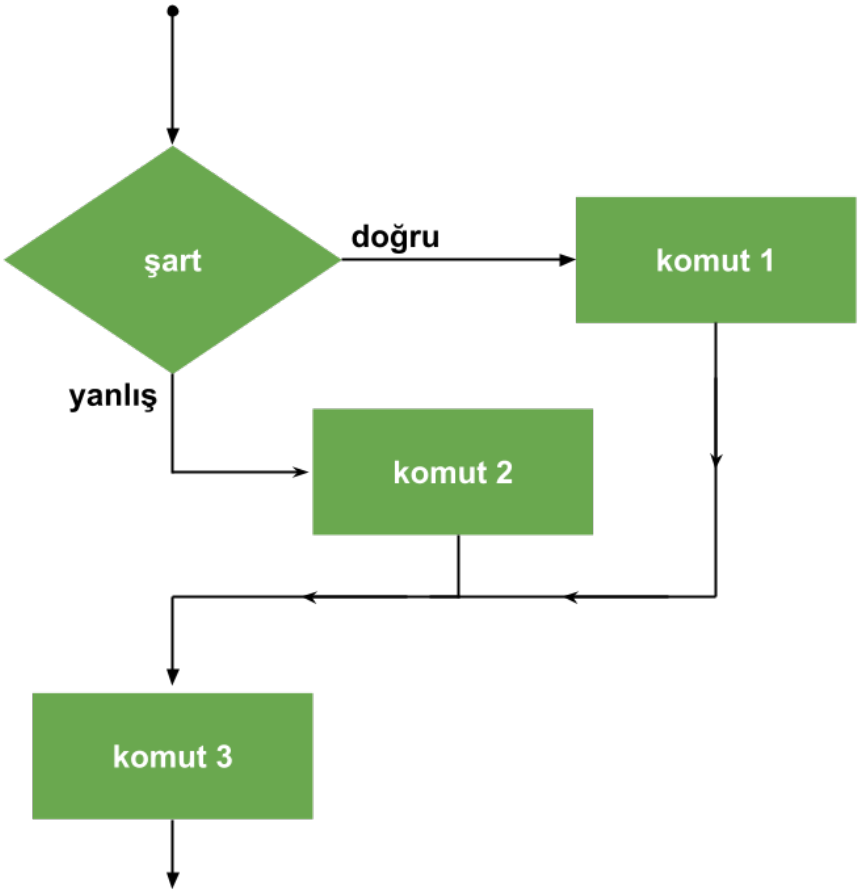
else Deyimi

- "**else**" deyimi sadece if ile birlikte kullanılan bir şartlı dallanma deyimidir. İkisinin beraber kullanımı şöyle bir yapıda gerçekleşir:

```
if (şart)  komut 1
else   komut 2
komut 3
```

- Bu durumda, şart **doğru** ise **komut 1** çalıştırılır ve sonra **komut 3**'e geçilir. Şartın değeri **yanlış** ise, bu durumda da **komut 1** atlanarak doğrudan **else**'i takip eden **komut 2**'ye geçilir. Bu çalıştırıldıktan sonra program yine **komut 3** ile çalışmaya devam eder.

else Deyimi



else Deyimi

- Bu örnekte programın içinde tanımlanmış integer tipli 'parola' değişkeninin değerinin 2013 olup olmadığı şartını denetliyoruz ve eğer şart doğru olarak ifade edilebiliyorsa ekrana '*Parolalar eşleşiyor.'* yazdırıyoruz.

```
public class SartliDallanmaOrnek1 {
    public static void main(String[] args) {
        int parola=2013;

        if (parola==2013) {
            System.out.println("Parolalar eşleşiyor.");
        }

    }
}
```

Örnek 4.1


```
import java.io.IOException;
import java.util.Scanner;

public class SartliDallanmaOrnek2 {
    public static void main(String[] args) throws IOException {

        Scanner giris=new Scanner(System.in);

         //String kullanicininYazdigi;
        int istanbulPlakaKodu=34;
        int girilenPlakaKodu;

         //soruyu ekrana basalım
        System.out.println("Istanbul'un plaka kodu nedir?");

         //klavyeyle girilen değeri integer olarak alalım
        girilenPlakaKodu=giris.nextInt();

        if(girilenPlakaKodu==istanbulPlakaKodu) {
            System.out.println("Evet! Dogru yanit.");
        } else {
            System.out.println("Hayir, Istanbul'un plaka kodu 34'tur.");
        }

    }
}
```

Karşılaştırma İşlemleri

- Şartlı bir ifadede iki değer arasında karşılaştırma işlemi gereklidir. **İlişkisel operatörler (relational operators)** olarak da anılan 6 karşılaştırma operatörü vardır.

KARŞILAŞTIRMA OPERATÖRÜ	İŞLEVİ
<	den daha küçük
>	den daha büyük
<=	küçük ya da eşit
>=	büyük ya da eşit
==	-e eşit
!=	eşit değil

Karşılaştırma İşlemleri

- Java dilinde de karşılaştırma sonucu doğru ise **Boolean** tipinde **true (doğru)** değeri elde edilirken karşılaştırma sonucu yanlış ise de **Boolean** tipinde **false (yanlış)** değeri elde edilir.

İFADE	DEĞER
-12 < 0	1
0 > 23	0
1 == 1	1
3 != 7	1
1 >= -2	1
6 > 8	0

Örnek 4.2

- **f(x)** ve **g(x)** fonksiyonları,
- **x > 0 ise f(x)= 1 / (1 + x) ve g(x)= 1 / (x)**
  x≤0 ise f(x) = 1 / (1+ x^2) ve g(x)=1 / (1 + x + x^2)
- şeklinde tanımlanıyor. x değeri klavyeden girildiğinde, f(x) ve g(x)'i hesaplatan ve yazdıran bir Java programını yazın.

Örnek 4.2


```
import java.io.IOException;
import java.util.Scanner;

public class If1 {
    public static void main(String args[]) throws IOException {
        Scanner giris = new Scanner(System.in);
        double x, f, g;
        System.out.println("Bir sayi giriniz: ");
        x = giris.nextDouble();
        if (x > 0.0) {
            f = 1.0 / (1.0 + Math.log(x));
            g = 1.0 / (x + Math.log(x));
        } else {
            f = 1.0 / (1.0 + x * x);
            g = 1.0 / (1.0 + x + x * x);
        }
        System.out.println("\nf(x)=" + f);
        System.out.println("\ng(x)=" + g);
    }
}
```

```
Bir sayi giriniz:
2

f(x)=0.5906161091496412

g(x)=0.37131279241563214
```

```
Bir sayi giriniz:
-1

f(x)=0.5

g(x)=1.0
```

Örnek 4.3

- Bir satış elemanının sattığı ürün miktarına göre alacağı günlük ücret aşağıdaki gibi belirleniyor:
- Günlük satış miktarı **50 adetten az ise** 15 TL tutarındaki sabit ücrete, satılan ürün başına 1 TL değerinde prim eklenerek günlük ücret belirlenir.
- Günlük satış miktarı **50 adet ya da daha fazla** ise, bu durumda günlük sabit ücret 15 TL alınarak, satılan ürün başına da ilk 50 adet ürün için 2 TL, 50 adedi aşan kısım için de 3 TL prim verilerek günlük ücret belirlenir.
- Bir satıcının günlük satış miktarı bilgisayara girildiğinde **satıcının alacağı günlük ücreti hesaplayan** bir Java programı yazınız.

Örnek 4.3


```
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.Scanner;

class Satis {
    public static void main(String args[]) throws IOException {
        Scanner giris = new Scanner(System.in);
        double satis, ucret; //DecimalFormat tipiyle özel biçimlerde sayılar gösterebilirsiniz.
        DecimalFormat nf = new DecimalFormat( pattern: "###,###.00");
        System.out.println("Gunluk kac tane urun satiyorsunuz? ");
        satis = giris.nextDouble();
        if (satis < 50) {
            ucret = 15.0 + satis * 1.0;
        } else {
            ucret = 15.0 + 50 * 2.0 + (satis - 50) * 3.0;
        }
        System.out.println("Buna gore gunluk ucretiniz: " + nf.format(ucret) + " TL");
    }
}
```

Örnek 4.4


```
import java.util.Scanner;

public class KareKok {
    public static void main(String args[]) {
        Scanner giris = new Scanner(System.in);
        double sayi;
        System.out.println("Karekokunu bulmak icin bir sayi giriniz: ");
        sayi = giris.nextDouble();
        if (sayi < 0) System.out.println("Olmadi, pozitif sayi girmeliydiniz");
        else System.out.println(sayi + "'nin karekoku: " + Math.sqrt(sayi));
    }
}
```

İç içe if Deyimleri

- Tek bir *if* deyimi, programınızın iki seçenekten birini seçmesine olanak sağlar ve tüm dallanma bu kadardır.
- Öte yandan pratikte daha fazla dallanmaya ihtiyaç duyan uygulamalar yapmanız gerekir.
- Birinci karardan sonra ikinci, ikinciden sonra üçüncü kararın (dallanmanın) alınması gerekebilir.
- Bu şekilde giden bir program akışı için **iç içe if deyimleri (nested if statements)** kullanmalısınız.

Örnek 4.5


```
public class IcIceIfDeyimi1 {
    public static void main(String[] args) {
        int sinav1 = 45;
        int sinav2 = 50;
        int sinav3 = 66;
        //ortalama'yı float olarak alıyoruz.
        float ortalama = (float) (sinav1 + sinav2 + sinav3) / 3;
        String sonuc;
        if (ortalama < 45) {
            //0-44 arası ortalama zayıf
            sonuc = "Zayif";
        } else if (ortalama < 55) {
            //45-54 arası ortalama geçer
            sonuc = "Gecer";
        } else if (ortalama < 70) {
            //55-69 arası ortalamaya orta
            sonuc = "Orta";
        } else if (ortalama < 85) {
            //70-84 arası ortalamaya iyi
            sonuc = "Iyi";
        } else {
            //başka bir sonuç kalmadığı için tekrar
            //if ile şart aramıyoruz
            sonuc = "Pekiyi";
        }
        System.out.println("Ortalama: " + ortalama);
        System.out.println("Buna gore sonuc: " + sonuc);
    }
}
```

Örnek 4.6


```
import java.io.IOException;
import java.util.Scanner;
public class IcIceIfDeyimi2 {
    public static void main(String[] args) throws IOException {
        Scanner giris = new Scanner(System.in);
        int a, b, c; //sayılar, bu değişkenlere
        int enKucukSayi; //en küçüğü bir yerde tutalım
        System.out.println("Birinci sayiyi giriniz: ");
        a = giris.nextInt();
        System.out.println("Ikinci sayiyi giriniz: ");
        b = giris.nextInt();
        System.out.println("Ucuncu sayiyi giriniz: ");
        c = giris.nextInt();
        if (a < b) {
            if (a < c) {
                enKucukSayi = a;
            } else {
                enKucukSayi = c;
            }
        } else if (b < c) {
            enKucukSayi = b;
        } else {
            enKucukSayi = c;
        }
        System.out.println("En kucuk sayi: " + enKucukSayi);
    }
}
```

Örnek 4.7

- Gelir vergisinin aşağıdaki kurallara göre belirlendiğini varsayalım:
- Gelir ≤ 150,000 ise vergi oranı %25
  Gelir ≤ 300,000 ise vergi oranı %30
  Gelir ≤ 600,000 ise vergi oranı %35
  Gelir ≤ 1,200,000 ise vergi oranı %40
  Gelir > 1,200,000 ise vergi oranı %50
- Gelir bilgisi klavyeden girilecek.
- Gelir bilgisi, yukardaki vergi hesaplama kurallarına göre kontrol edilerek vergi hesaplanacak.
- Hesaplanan vergi ekrana yazdırılacak.

Örnek 4.7


```java
import java.util.Scanner;

public class Vergi {
    public static void main(String args[]){
        Scanner giris = new Scanner(System.in);
        double gelir, v, v1, v2, v3, v4;
        System.out.println("Gelir miktarınız (TL): ");
        gelir = giris.nextDouble();
        v1 = 150000 * 0.25;
        v2 = 150000 * 0.30;
        v3 = 300000 * 0.35;
        v4 = 600000 * 0.40;
        if (gelir <= 150000)
            v = gelir * 0.25;
        else if (gelir <= 300000)
            v = v1 + (gelir - 150000) * 0.3;
        else if (gelir <= 600000)
            v = v1 + v2 + (gelir - 300000) * 0.35;
        else if (gelir <= 1200000)
            v = v1 + v2 + v3 + (gelir - 600000) * 0.4;
        else
            v = v1 + v2 + v3 + v4 + (gelir - 1200000) * 0.5;
        System.out.println("Odemeniz gereken vergi: " + v);
    }
}
```

İkinci Dereceden Bir Denklemin Köklerinin Bulunması

- $ax^2 + bx + c = 0$
- şeklinde ifade edilir. Denklemin kökleri hakkında bilgi sahibi olabilmek için, denklemin diskriminantı adı verilen,
- $\Delta=b^2-4ac$
- ifadesini hesaplamamız gerekir. Diskriminantın değerlerine göre, eğer $\Delta<0$ ise denklemin gerçel sayılarla ifade edilebilecek bir kökü yoktur.
- $\Delta=0$ ise denklemin, değerleri birbirine eşit olan iki kökü vardır ve bunlar,
- $x_1=x_2= -b/(2a)$
- şeklinde hesaplanırlar. Bunlara *iki kat kök* ya da *çakışık kök* adı verilir.
- $\Delta>0$ ise de denklemin birbirinden farklı iki gerçel kökü vardır ve bunlar,
- $x_1=(-b+\sqrt{\Delta})/(2a)$ ve $x_2=(-b-\sqrt{\Delta})/(2a)$
- şeklinde hesaplanırlar.

İkinci Dereceden Bir Denklemin Köklerinin Bulunması


```
import java.util.Scanner;

public class IkinciDerece{
    public static void main(String args[]){
        Scanner giris = new Scanner(System.in);
        double a, b, c, x1, x2, delta;
        System.out.println("a: ");
        a=giris.nextDouble();
        System.out.println("b: ");
        b=giris.nextDouble();
        System.out.println("c: ");
        c=giris.nextDouble();
        delta = b * b - 4 * a * c;
        if (delta < 0) {
            System.out.println("Gercek kok yoktur. \n");
        }
        else if (delta == 0) {
            x1 = -b / (2 * a);
            System.out.println("x1=x2=" + x1);
        }
        else {
            x1 = (-b + Math.sqrt(delta)) / (2 * a);
            x2 = (-b - Math.sqrt(delta)) / (2 * a);

            System.out.println("x1=" + x1);
            System.out.println("x2=" + x2);
        }
    }
}
```

Soru İşareti Operatörü

- *if / else deyimi* yerine kullanılabilecek bir seçenek de **?** üçlü operatörüdür. *Üçlü (ternary) operatör* denmesinin nedeni doğal olarak üç tane operand ile işlem görmesidir.
- ***?*** operatörü ile kontrol yapısının yazılış biçimi aşağıdaki gibidir:
- **İfade1 ? İfade2:İfade3;**
- Çalışma biçimi şu şekildedir:
- **İfade1** hesaplanır. **İfade1**'in değeri **doğru (true)** ise bu durumda **ifade2** hesaplanır ve bir sonraki deyime geçilir. **İfade1**'in değeri **yanlış (false)** ise, **ifade3** hesaplanır ve bir sonraki deyime geçilir.
- **?** operatörü ile oluşturulan yukardaki kalıp,
- **if(ifade1) ifade2**
  else ifade3;

yapısına denktir.

- **?** operatörü, **ifade1**'in doğru ya da yanlış olması durumunda sadece bir deyimin çalıştırılacağı durumlar için *if / else* yapısı yerine kullanılabilir. **İfade2** ve **ifade3** yerinde fonksiyonlar da bulunabilir.

Soru İşareti Operatörü


```
import java.io.IOException;
import java.util.Scanner;

public class Soru {
    public static void main(String args[]) throws IOException {
        Scanner giris = new Scanner(System.in);
        String sonuc;
        double not;
        System.out.println("Sinav notunu (0-100) giriniz: ");
        not = giris.nextDouble();
        sonuc = not >= 50 ? "Basarili" : "Basarisiz";
        System.out.println("Sonuc: " + sonuc);
    }
}
```

switch / case Yapısı

- Bir programda çok sayıda koşul kontrolü ve bunların sonucuna göre gerçekleştirilmesi gereken işlemler varsa, *if-else* yapıları ile akışın izlenmesi zorlaşabilir. Böyle durumlar genellikle **switch** deyiminin kullanılmasının gerekli olacağı durumlardır. *switch deyimi*, tek bir ifadenin değerine göre sınırsız sayıda çalıştırma yolu belirlemeyi sağlayan bir komuttur.
- **switch** sözcüğünden hemen sonra gelen ifade parantez içinde yer almalı ve bir tamsayı ifade olmalıdır.
- **case** anahtar sözcüklerini izleyen ifadeler tamsayı sabit türünde ifadeler olmalıdır, yani **değişken içermemelidir**.

switch / case Yapısı

- *switch* deyiminin çalışma prensibi basittir. *switch* ifadesi hangi *case'i* izleyen sabitle çakışıyorsa, programın kontrolü o *case'i* izleyen kısma geçer. Bu *case*'den sonraki deyimler de kontrol edilmeden çalıştırılır.
- *case* sabitlerinden hiçbiri ifade ile uyuşmuyorsa, programın akışı *default*'u izleyen kısıma geçer. (Bu kısım mevcutsa) *default*'un en sonda olması şart değildir. Fakat en sona koymak iyi bir programlama stilidir.
- İki *case* sabiti aynı değeri alamaz.
- *Switch/case* yapısında, programın kontrolünün ifadenin değerine göre *case* seçeneklerinden sadece birini çalıştırıp bundan sonra *switch/case* yapısını terketmesini isiyorsak bu durumda her *case* seçeneğinden sonra **break** komutu yerleştirmeliyiz.

Örnek 4.8


```
import java.util.Scanner;

public class Cases{
    public static void main(String args[]) {
        Scanner giris = new Scanner(System.in);
        int secim;
        System.out.println("Secim yapiniz (1 - 2 - 3)");
        secim=giris.nextInt();
        switch (secim) {
            case 1 :
                System.out.println("A sinifi dergilerin listesi");
                break;
            case 2 :
                System.out.println("B sinifi dergilerin listesi");
                break;
            case 3 :
                System.out.println("C sinifi dergilerin listesi");
                break;
            default :
                System.out.println("Hatali secim! 1, 2 ya da 3'e basiniz.");
                break;
        }
    }
}
```

Örnek 4.9

- Tur seçenekleri ekrana yazdırılır.
- Kullanıcıdan bu tur seçeneklerinden bir tanesini seçmesi istenir.
- Kullanıcının seçimine uygun olan tur hakkındaki bilgiler ekrana yazdırılır.

Örnek 4.9


```java
import java.io.IOException;
import java.util.Scanner;

public class Cases2{
    public static void main(String args[]) throws IOException{
        Scanner giris = new Scanner(System.in);
        int i;
        System.out.println("Portakal Turizm A.S. Sunar");
        System.out.println("(1) Orta Avrupa Turu");
        System.out.println("(2) Amerika Turu");
        System.out.println("(3) Uzak Dogu Turu");
        System.out.println("Seciminiz?");
        i = giris.nextInt();
        switch (i) {
            case 1 :
                System.out.println("Ucak ile Viyana, Budapeste, Prag");
                System.out.println("4 yildizli otellerde sok! 1700$\n");
                break;
            case 2 :
                System.out.println("Ozel ucak ile New York, Boston, Los Angeles");
                System.out.println("5 yildizli otellerde sok! 3700$\n");
                break;
            case 3 :
                System.out.println("Ozel ucak ile Bang Kong, Hong Kong");
                System.out.println("Tokyo, Pekin");
                System.out.println("5 yildizli otellerde sok! 4000$\n");
                break;
            default :
                System.out.println("Hatali secim! 1, 2 veya 3 girilmeli");
                break;
        }
    }
}
```

KAYNAKLAR

https://gelecegiyazanlar.turkcell.com.tr/