Programlamaya Giriş

BİL131 - Bilişim Teknolojileri ve Programlama

Hakan Ezgi Kızılöz

Genel Bilgiler

Ders konuları

1. Programlamaya Giriş
2. Java Programlarının Genel Yapısı
3. Temel Veri Türleri, String, Giriş / Çıkış
4. Akış Kontrolü
5. Metotlar
6. Diziler
7. Dizi Listeleri
8. Nesne Türleri, Nesne, Metotlar

Programlamanın Temelleri

- Problemin veya amacın anlaşılması ve gerçekleştirmek için planlama yapılması gerekir.

- Algoritma tasarımı
  - Algoritma, problemin çözümü için takip edilen adımlar, kurallar kümesi veya süreçtir.
  - Programın kodlanmasından önce problemin çözümü için adımların oluşturulması gerekir.

Programlamanın Temelleri

- Başarılı bir programlama için aşağıdaki adımlar izlenmelidir:
  - Adım 1 - Problemi anlamak: Problemin ne istediğini anlamak, problemin girdilerini ve çıktılarını belirlemek.
  - Adım 2 - Plan yapmak: Problemin nasıl çözülebileceği hakkında düşünmek, eğer problem çözülemiyorsa daha önce çözülmüş benzer bir probleme benzeterek çözmeye çalışmak.
  - Adım 3 - Planı uygulamak: Yapılan planı adım adım uygulamak.
  - Adım 4 - Test etmek: Bulunan sonucu irdelemek ve doğruluğunu kontrol etmek.

Programlama Hataları

* Bir algoritmayı program haline çevirip bilgisayar ortamında çalıştırdığımızda 3 çeşit hata ile karşılaşabiliriz:
  * Sözdizim hataları (Syntax error)
  * Çalıştırma zamanı hataları (Runtime error)
  * Mantık hataları (Logical error)

Programlama Dilleri ve Derleyiciler

* Günümüzde programlama dillerinin çoğu kullanıcıların kolaylıkla anlayıp kullanacağı şekilde tasarlanmıştır. Bu tür dillere **üst-düzey diller** denir.
  * *Örnek: Java, C, C++, Basic, vb..*
* İnsanların aksine, bilgisayarlar üst-düzey dilleri anlayamaz. Bilgisayarların anlayabileceği dillere ise **alt-düzey diller** denir. Alt-düzey dillere genellikle **makine dilleri** veya **assembly** denmektedir.

Programlama Dilleri ve Derleyiciler

* Üst-düzey dilde yazılmış bir programın makine diline çevrilmesi için derleyiciler (compiler) kullanılmalıdır.
* Bir derleyici bütün üst-düzey dilleri makine diline çeviremez, her üst-düzey dil için farklı derleyici gerekmektedir.
* Bunun yanı sıra; aynı üst-düzey dil için bile, her mikroişlemci ve işletim sistemi için ayrı bir derleyici gerekmektedir.
* Java için farklı bir yol izlenmektedir.

Java Byte Kodu

* Java derleyicisi, Java programını makine diline değil **java byte kodu** adı verilen bir programa çevirir.
* Bu program, **Java Virtual Machine (JVM)** denilen hayali bir bilgisayarın makine kodudur.
* Bu kod, **yorumlayıcı** tarafından makine diline kolaylıkla çevrilebilmektedir.
* Java üst-düzey dili ile yazılmış bir program, *javac* komutu ile java byte koda çevrildikten sonra *java* komutu ile çalıştırılmaktadır.

Merhaba dünya! Programı

```
//Merhaba dünya! programı.

public class MerhabaDunya
{
   public static void main (String [] args)
   {
      System.out.println("Merhaba dunya!");
   }
}
```

Java Programının Temel Formatı

- Java programları fonksiyonlar halinde yazılır
  - Tüm fonksiyonların birer adı, veri alma ve veri geri döndürme özellikleri vardır.
  - Fonksiyonlar programın işlevlerini gerçekleştirir.
  - Java programlarının başlangıç noktası **main()** fonksiyonudur.

main() fonksiyonu

- Java programının başlangıç noktasıdır.
- Java programları bir **`main()`** fonksiyonuna sahiptir.
- main() fonksiyonu içerisinde yazılmış olan bütün deyimler, yazılmış oldukları sırada çalıştırılmaktadır.

```
public static void main(String [] args)
{
    ...
}
```

Java Deyimleri

- Java deyimleri çalıştırılacak komutları ifade eder.
- Java'daki çoğu deyim noktalı virgülle sonlandırılır.
- Merhaba dünya! programı bir deyime sahiptir.

```
System.out.println("Merhaba dünya!");
```

- `out` çıkışı yönlendirir.
- `println` ile String konsol ekranına gönderilir ve yeni satıra geçilir.

Java'da Boşluk Karakterleri

- Enter, tab ve space ile oluşturulan karakterlere boşluk karakterleri (whitespaces) denmektedir.
- Boşluk karakterleri programın okunabilirliğini artırır.
- Derleyici boşluk karakterleri göz ardı eder.
- Merhaba dünya! programı aşağıdaki gibi yazılırsa yine çalışır.

```
public class MerhabaDunya { public static void main
   (String [] args) { System.out.println("Merhaba
   dünya!"); }}
```

Açıklamalar

- Açıklama satırları program hakkında bilgiler vermek için kullanılır.
- Java içinde açıklama yazmanın iki yolu vardır:

- Tek satırlık açıklamalar

```
// Açıklamalar bu satıra yazılabilir.
// Derleyici, iki eğik çizgi işaretinden itibaren
// satır sonuna kadar olan her şeyi gözardı eder
```

- Çok satırlık açıklamalar

```
/* Açıklama yazmanın diğer bir yoludur. Derleyici, eğik
çizgi yıldız ile yıldız eğik çizgi arasındaki her şeyi
gözardı eder. */
```

Yazım Dili

- Yazım dili, programın yazım kurallarını ifade eder.
- Derleyici programda yazım hatası bulursa anlamlı bir mesajla programcıya bilgi verir.
- Programda az sayıda hata bulunsa bile, derleyici çok sayıda hata üretebilir.

Belgeleme ve Program Yazma Tarzı

- Yazılan programın kolay okunabilir olması gerekir.
- Tanımlayıcı açıklamaların yapılması gerekir.
- Değişken isimleri anlamlı ve uygun uzunlukta olmalıdır.
- Programdaki bloklar hizalı olmalıdır.
- Başlangıçta okunabilirlik için harcanan zaman, derleyici hatalarının düzeltilmesi veya programın güncellenmesi sırasında çok zaman kazandırır.

Belgeleme ve Program Yazma Tarzı

- Yazılan programın anlaşılabilirliğini artırmak amacıyla açıklamalar yazmak gerekir.
- Aynı satır içine açıklamalar **//** ile yazılır.

```
int a = 5; // a degiskenine deger atandi
// Bu satırın tamamı yorumdur.
// Burada a = 10 yazılsa bile
// java bunu göz ardı edecektir.
// a = 10;
System.out.println(a);
```

Çıktı:
5

Belgeleme ve Program Yazma Tarzı

- Birden çok satırdan oluşan açıklamalar **/*** ile ***/** arasında yazılır.

```
/*
* TOBB Ekonomi ve Teknoloji Üniversitesi
* Bilgisayar Mühendisliği Bölümü
*/
int a = 5;
System.out.println(a);
```

Çıktı:
5

Belgeleme ve Program Yazma Tarzı

* Programın anlaşılabilirliğini artırmak amacıyla blokların aynı hizada yazılması gerekir.

```
public class GuzelProgram
{
    public static void main(String [] args)
    {
        System.out.println();
            .
            .
            .
        System.out.println();
    }
}
```

Programlama Dillerindeki Veri Türleri

- Bilgisayarlar farklı veri türleri üzerinde işlem yapabilmektedir.
- Bir veri türü, programda kullanılacak verinin alabileceği değer aralığını belirler.
- Kullanılacak her değer için bir tür belirlenmelidir.

Veri Türü	İsim	İçerdiği Veri	Örnek
Karakter	char	Bir tek karakter	a
Tam sayı	int	Bir tam sayı değeri	43
Ondalıklı Sayı	double	Bir ondalıklı sayı değeri	3.35364757458
Mantıksal İfade	boolean	Değeri true veya false olan bir mantıksal ifade değeri	true

Java'da Temel Veri Türleri

* Aşağıdaki tabloda Java'da kullanılan veri türleri ve bu türlerin değer aralıkları görülmektedir.

	Veri Türü	Boyutu (byte)	Değer Aralığı
Tam Sayılar	byte	1	-128 ... +127
	short	2	-32768 ... +32767
	int	4	-214748368 ... +214748367
	long	8	-2^{63} ... $+(2^{63}-1)$
Ondalıklı sayılar	float	4	$\pm 3.40 \times 10^{38}$ ... $\pm 1.40 \times 10^{-45}$
	double	8	$\pm 1.76 \times 10^{308}$ ... $\pm 4.94 \times 10^{-324}$
Karakter	char	2	Unicode karakter
Boole	boolean	1 bit	true/false

Değişkenler

- Dışarıdan alınan veya hesaplayarak elde edilen verilerin tamamını o anda kullanmayız, bazı verilerin değerlerini daha sonra kullanabilmek için saklamak isteriz.
- Bunun için değişkenlere ihtiyacımız vardır.
- Değişkenler, türü belli olan ve kendisine atanan değeri saklayarak daha sonra erişmemize izin veren yapılardır.

Değişken Tanımlamaları

- Java'da değişken tanımlamak için

```
veri_türü degisken_adi;
```

yazım formatı kullanılır.

- Örnek değişken tanımlamaları:

```
double balance;
double deposit;
double withdraw;
int transaction_count;
int check_number;
```

Değişken Tanımlama ve Değer Atama

- Değişkenlere değer atanması yapabilmek için öncelikle değişkenin tanımlanması gerekmektedir. Tanımlanmamış değişkene değer ataması yapılamaz.

```
double deger;
deger = 35.29;
```

- Bir deyim içerisinde hem değişken tanımı, hem de değişkenin değer ataması yapılabilir.

```
int num = 5;
```

- Aynı türden farklı değişkenler, bir deyim içerisinde virgül (,) ile ayrılarak tanımlanabilirler.

```
double money = 0.0, speed;
speed = 0.0;
```

İsimlendirme Kuralları

- Değişkenlere isim verilirken uyulması gereken kurallar vardır:
  - İsimler A-Z, a-z, 0-9 veya _ karakterlerinden oluşabilir.
  - İsmin ilk karakteri harf veya _ olmalıdır.
  - İsimlerde ~ ! @ # $ % ^ & * ( ) – “+ = \ | ’, sembolleri ve boşluk karakteri kullanılamaz.
  - Anahtar kelimeler değişken ismi olarak kullanılamaz.
- Java'da isimler büyük/küçük harfe duyarlıdır, yani *double ay;* ile *double AY;* farklı değişkenlerdir.

İsimlendirme Kuralları

- İsimlendirme ile ilgili örnek durumlar aşağıdadır:

Değişken İsmi	Geçerli / Geçersiz	Geçersiz ise Sebebi
balance	geçerli	-
transaction amount	geçersiz	boşluk karakteri içeremez
convert_2_#s	geçersiz	# gibi karakterler kullanılamaz
MyMoney	geçerli	-
case	geçersiz	Java dilindeki anahtar sözcükler kullanılamaz
Case	geçerli	-
4_temperature	geçersiz	rakam ile başlayamaz

Operatörler

- Java da operatörler belirli bir işlemi ifade eder.

```
F_temp = 9.0/5.0 * C_temp + 32.0;
```

- Operatörler eşitliğin sağ tarafında kullanılırlar.

Atama Operatörü

- Atama operatörü (=) sağ taraftaki değeri sol taraftaki değişkene aktarır.

```
miktar = 1534.34;
islemSirasi = 8;
x = y;
```

- Bir deyimle birden fazla değişkene değer atanabilir.

```
a = b = c = 0;
```

Atama Operatörü

- Sol taraftaki değerin sağ tarafa ataması yapılamaz.
- Örneğin

```
double x, y;
5.2 = x;      // hata
3 + 7.1 = y; // hata
```

atamaları hatalı iken

```
double x, y;
x = 5.2;      // dogru kullanim
y = 3 + 7.1;  // dogru kullanim
```

atamaları doğrudur.