

# Java Programlamaya Giriş

2. Hafta

Dr. Öğr. Üyesi BÜŞRA ÖZDENİZCİ KÖSE

İşletme Bölümü

İşletme Fakültesi



# q2ivm3u

---

ENF101 Programlamaya Giriş

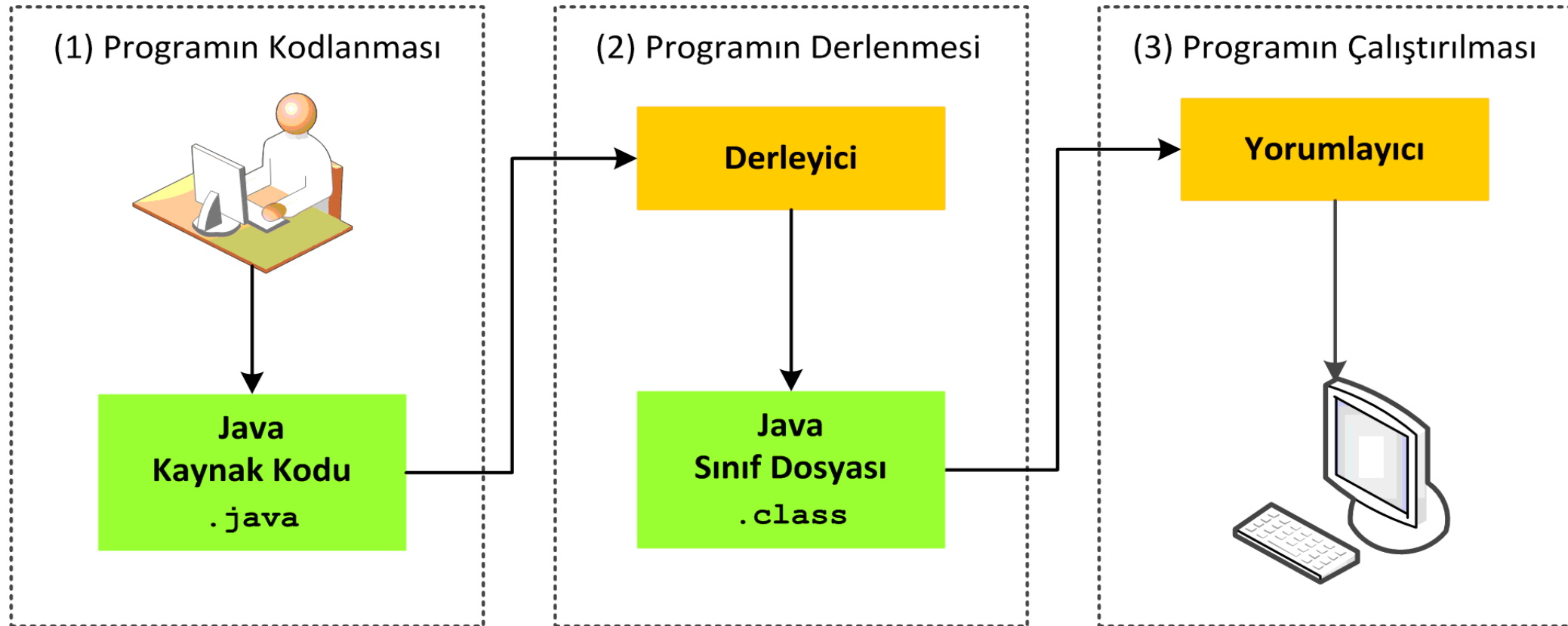


# Dersimizin Amacı

- Enformasyon teknolojisi problemlerini tanımlamak ve analiz edebilmek
- Algoritmalar tasarlayabilmek ve geliştirebilmek
- Tanımlanmış problemleri çözmek için bilgisayar programı yazabilmek



# Java Programı Nasıl Çalışır?



# Basit Bir Java Programı

- Ekrana *Merhaba Dünya!* yazısını görüntüleyecek veya yazdıracak basit bir Java programı ile başlayalım

01	public class Merhaba	Başlık
02	{	Gövde
03	public static void main( String[] args )	
04	{	
05	System.out.println("Merhaba Dünya!");	
06	}	
07	}	
01	Merhaba Dünya!	

# Başlık

- İlk satır Sınıfı (class) tanımlamaktadır. Her Java programının **en az bir sınıfı olmalıdır** ve her sınıfın bir ismi vardır.
- Her sınıf ismi **büyük harf** ile başlar. Bu örnekte sınıf ismi Merhaba.

01	<b>public class Merhaba</b>	Başlık
02	{	Gövde
03	public static void main( String[] args )	
04	{	
05	System.out.println("Merhaba Dünya!");	
06	}	
07	}	

# Gövde

- Üçüncü satırda **main** metodu tanımlanmıştır. Bir sınıf bir veya daha fazla metod içerebilir. Bir Java programı *HER ZAMAN* öncelikle **main** metotundan çalıştırılır. **main** metodu programın çalışmaya başladığı başlangıç noktasıdır.
- Metod, komutlar içeren yapılardır. Bu programda **main** metodu **System.out.println** komutunu içermektedir.

01	public class Merhaba	Başlık
02	{	Gövde
03	<b>public static void main( String[] args )</b>	
04	{	
05	<b>System.out.println("Merhaba Dünya!");</b>	
06	}	
07	}	

# Gövde (cont.)

01	public class Merhaba	Başlık
02	{	Gövde
03	<b>public static void main( String[] args )</b>	
04	{	
05	<b>System.out.println("Merhaba Dünya!");</b>	
06	}	
07	}	



Java kaynak kodları büyük/küçük harfe duyarlıdır.

Örneğin, main metodunu Main olarak yazmak yanlış olacaktır.



# Gövde (cont.)

- Küme parantezleri (*curly braces*) blok oluşturmayı sağlar, programın bileşenlerini gruplar. Java'da her blok, açık küme parantezi ({) ile açılır ve kapalı küme parantezi (}) ile sonlanır. Her sınıfın metotlarını, verilerini gruplamak için sınıf bloğu (*class block*) kullanılır. Her metodun komutlarını gruplamak için metod bloğu (*method block*) kullanılır. İç içe bloklar kullanılabilir.

01	public class Merhaba		Başlık
02	{		Gövde
03	public static void main( String[] args )		
04	{		
05	System.out.println("Merhaba Dünya!");		
06	}		
07	}		

# Gövde (cont.)

01	public class Merhaba		Başlık
02	{	<b>Sınıf bloğu</b>	Gövde
03	public static void main( String[] args )		
04	{		
05	System.out.println("Merhaba Dünya!");		
06	}	<b>Metot bloğu</b>	
07	}		



Açık küme parantezinin mutlaka kapalı küme parantezi ile sonlandırılması gerekir. Hata oluşmasını engellemek için açıldığı anda küme parantezinin kapatılması gerekmektedir.

Java IDE ortamında programlama yaparken, otomatik olarak açılan küme parantezleri kapatılmaktadır 😊

# Gövde (cont.)

- Her komut *noktalı virgül (;)* ile sonlandırılır; komut sonlandırıcı (*statement terminator*) olarak bilinir.

01	public class Merhaba	Başlık
02	{	Gövde
03	public static void main( String[] args )	
04	{	
05	System.out.println("Merhaba Dünya!");	
06	}	
07	}	

**Komut Sonlandırıcı** ←

# Gövde (cont.)

- String, bir sıra karakterlerden oluşan metinleri ifade eder. Bir String mutlaka **çift tırnak** (**double quotation marks**) içerisinde ifade edilir.

01	public class Merhaba	Başlık
02	{	Gövde
03	public static void main( String[] args )	
04	{	
05	System.out.println("Merhaba Dünya!");	
06	}	
07	}	

**Çift Tırnak**

# Gövde (cont.)

- **Parantezler** metotlar için kullanılır:
  - main ()
  - System.out.println ()

01	public class Merhaba	Başlık
02	{	Gövde
03	public static void main( String[] args )	
04	{	
05	System.out.println("Merhaba Dünya!");	
06	}	
07	}	

**Metotlar için Parantez Kullanımı**

# Özel Anlamalı Kelimeler (Reserved Words)

- public, static, void, class kelimeleri **ÖZEL ANLAMLI KELİMELER**dir.
- Java programlarında her biri belirli bir anlamı ifade etmek üzere önceden tanımlanmış –rezerve edilmiş– bazı sözcükler mevcuttur. Bu kelimeler program içinde başka bir amaç için kullanılamaz.

01	<b>public class</b> Merhaba	Başlık
02	{	Gövde
03	<b>public static void</b> main( String[] args )	
04	{	
05	System.out.println("Merhaba Dünya!");	
06	}	
07	}	

# Özel Anlamalı Kelimeler (cont.)

- Örneğin int, bu türde bir değişkeni tanımlarken kullanılabilir fakat bir değişken ismi olamaz.
- Örneğin, Java Compiler programı derlerken class kelimesini gördüğünde, -class kelimesinden sonra sınıfın adının belirtildiğini anlayacaktır.

01	<b>public class</b> Merhaba	Başlık
02	{	Gövde
03	<b>public static void</b> main( String[] args )	
04	{	
05	System.out.println("Merhaba Dünya!");	
06	}	
07	}	

# Java'da Özel Anlamalı Kelimeler

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	<b>public</b>	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	<b>static</b>	<b>void</b>
<b>class</b>	finally	long	strictfp	volatile
const	float	native	super	while

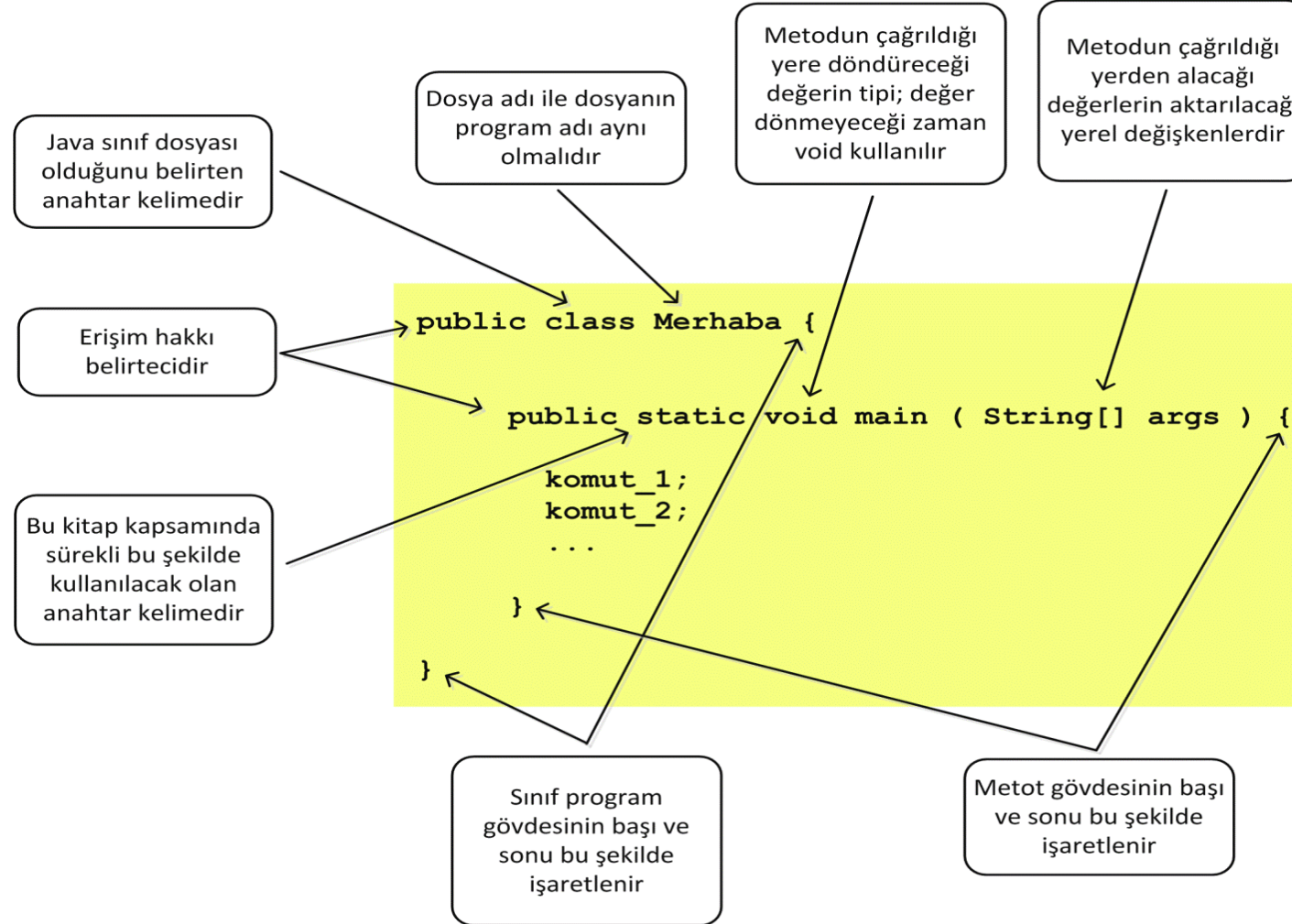


# Açıklamalar (Comments)

- Açıklamaları (**comment**), programlama yaparken yardımcı olması amacıyla kullanırız. Java dosyası içine iki farklı yöntem ile açıklama ekleyebiliriz.
- Açıklama bilgisi bir satıra sığıyor ise **//** karakterleri kullanılabilir. Bu karakterlerden başlayarak satır sonuna kadar yer alan kısımlar Java derleyicisi tarafından dikkate alınmaz.
- Eğer yazılacak olan not bilgisinin birden çok satıra yayılması söz konusu ise bu durumda yorumun öncesine **/\*** simgesi ve sonrasına da **\*/** simgesi konur.

```
01  /*  
02  Programcı: Busra OZDENIZCI KOSE  
03  */  
04  public class Merhaba  
05  {  
06      // main programımız burada!!!  
07      public static void main( String[] args )  
08      {  
09          System.out.println( "Merhaba Dünya!" ); //yazdır  
10      }  
11  }
```

# Kısaca, Java Program Yapısı



# Çeşitli Programlar: Hangisi En İyi ?

```
01 public class Merhaba
02 {
03     public static void main( String[] args )
04     {
05     }
06 }
```

```
01 public class Merhaba
02 {
03     public static void main( String[] args ) { }
04 }
```

```
01 public class Merhaba { public static void main
02 (
03     String[] args){}
04 }
```

# Programlama Biçimi ve Dokümantasyon

- Uygun Açıklamalar
- Anlamlı İsimlendirmeler
- Girintiler (Indentation) ve Satır Aralıkları
- Blok Yapıları

# Programlama Biçimi ve Dokümantasyon

- **Uygun Açıklamalar**
  - Programınızın en başında programın amacını, ne yaptığını, temel özelliklerini, veri yapıları, programcının ismini, tarihini, kısa açıklamasını vb. unsurlardan bahsetmeniz iyi olacaktır.
- Anlamlı İsimlendirmeler
- Girintiler ve Satır Aralıkları
- Blok Yapıları

# Programlama Biçimi ve Dokümantasyon

- Uygun Açıklamalar
- **Anlamlı İsimlendirmeler**
  - Anlamlı ve açıklayıcı isimler kullanınız
  - Sınıf isimleri: İlk harfi mutlaka büyüktür
  - Örneğin **Merhaba** veya **MerhabaDunya** sınıf isimleri olarak kullanılabilir.
- Girintiler ve Satır Aralıkları
- Blok Yapıları

# Programlama Biçimi ve Dokümantasyon

- Uygun Açıklamalar
- Anlamlı İsimlendirmeler
- **Girintiler ve Satır Aralıkları**
  - İki boşluklu girintiler mutlaka kullanınız
  - Kodu parçalara bölmek amacıyla satır boşlukları kullanınız
- Blok Yapıları

# Programlama Biçimi ve Dokümantasyon

- Uygun Açıklamalar
- Anlamlı İsimlendirmeler
- Girintiler ve Satır Aralıkları
- **Blok Yapıları**
  - Küme parantezlerini doğru kullanınız
  - Küme parantezini açarken *End-of-line* biçimini tercih ediniz



*Next-line  
style*

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Block Styles");
    }
}
```

*End-of-line  
style*

```
public class Test {
    public static void main(String[] args) {
        System.out.println("Block Styles");
    }
}
```

# Programlama Hataları

- Syntax Hataları
  - Derleyici sözdizimi hatalarını tespit eder
- Runtime (Program Çalışma) Hataları
  - Programın çalışmasını durdurur
- Mantıksal (Logic) Hatalar
  - Programın hatalı sonuç verir

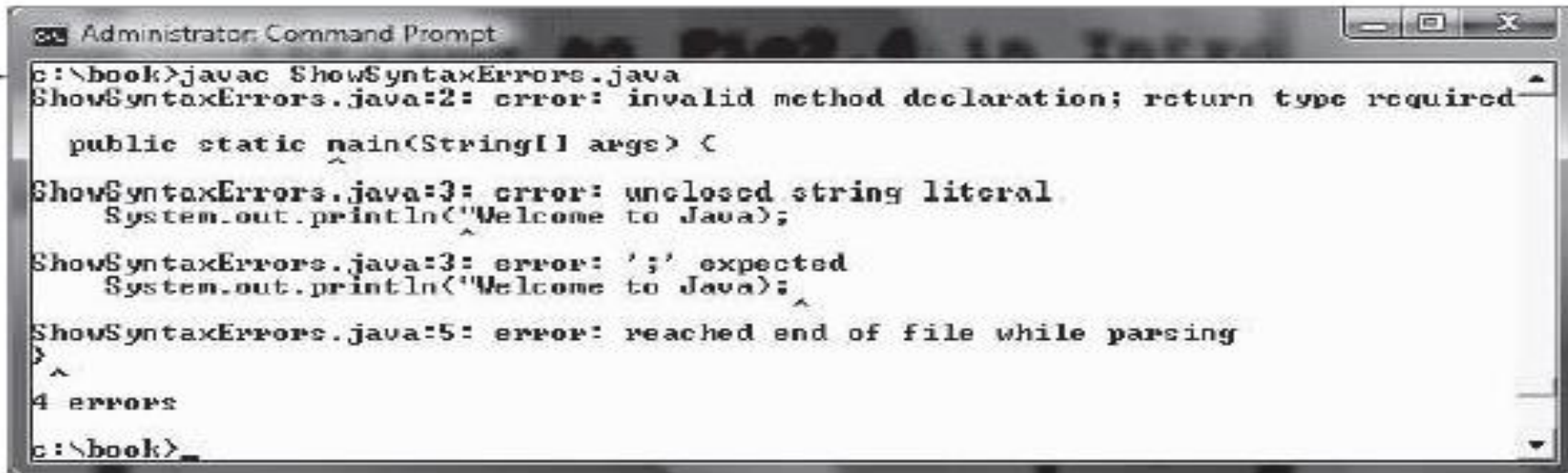
# Syntax Hataları (Syntax Errors)

- En çok karşılaştığımız hatalar: **syntax errors**
- Her programlama dilinde olduğu gibi, Java'nın da kendine ait sözdizimi kuralları (syntax rules) vardır
- Eğer programınız bir syntax hatası gerçekleştirirse -örneğin, noktalı virgül unutursanız, küme parantezini kapatmamışsanız, çift tırnak unutursanız vb.- **Java compiler** yani derleyici syntax hatalarını raporlar

# Syntax Hata Örneği

```
public class ShowSyntaxErrors {  
    public static main(String[] args) {  
        System.out.println("Welcome to Java);  
    }  
}
```

Compile →



```
Administrator: Command Prompt  
c:\book>javac ShowSyntaxErrors.java  
ShowSyntaxErrors.java:2: error: invalid method declaration; return type required  
    public static main(String[] args) {  
                   ^  
ShowSyntaxErrors.java:3: error: unclosed string literal  
        System.out.println("Welcome to Java);  
                           ^  
ShowSyntaxErrors.java:3: error: ';' expected  
        System.out.println("Welcome to Java);  
                           ^  
ShowSyntaxErrors.java:5: error: reached end of file while parsing  
    }  
    ^  
4 errors  
c:\book>
```

# Runtime Hata Örneği

```
public class ShowRuntimeErrors {  
    public static void main(String[] args) {  
        System.out.println(1 / 0);  
    }  
}
```

Run →



```
Administrator: Command Prompt  
c:\book>java ShowRuntimeErrors  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
    at ShowRuntimeErrors.main<ShowRuntimeErrors.java:4>  
c:\book>_
```

# Mantıksal Hata Örneği

- *Mantıksal hatalar*, programın istenilen sonucu vermemesi ile oluşur, programın mantıksal olarak yanlış tasarlanmasından kaynaklanır.
  - Örneğin, yukarıdaki örnekte programın sonucu Fahrenheit 67 derece çıkacaktır, yanlış bir sonuçtur. Sonucun aslında 95.0 derece olması gerekir. Gerekli hesaplamanın yanlış tanımlanmasından dolayı bu hata oluşmuştur.

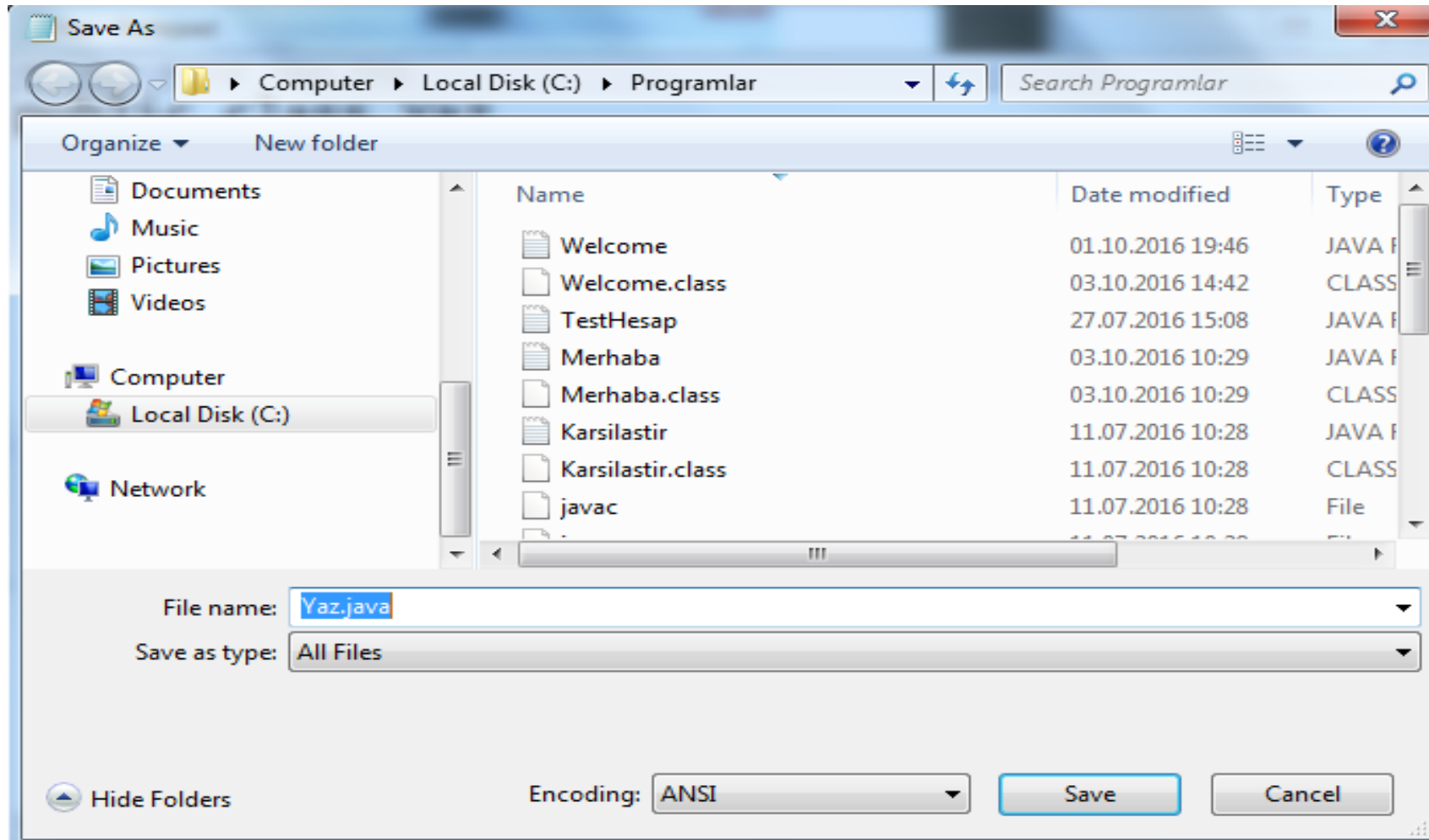
```
public class ShowLogicErrors {  
    public static void main(String[] args) {  
        System.out.println("Celsius 35 is Fahrenheit degree ");  
        System.out.println((9 / 5) * 35 + 32);  
    }  
}
```



# Notepad ile Programlama mümkün (1/3)

01	<code>public class Yaz</code>	Yaz isimli programın başlığı
02	<code>{</code>	Sınıfın başlangıcını belirten simge
03	<code>public static void main( String[] args )</code>	main() metodunu tanımlama
04	<code>{</code>	main() metodunun başlangıcını belirten simge
05 06 07	<code>System.out.println( "Merhaba Java :)" );</code> <code>System.out.println( "Merhaba Dünya :)" );</code> <code>System.out.println( "Java Öğreniyorum!" );</code>	Ekrana kullanıcının göreceği metni yazma
08	<code>}</code>	main() metodunun sonunu belirten simge
09	<code>}</code>	Sınıfın sonunu belirten simge

# Yaz.java ve All Files Type (2/3)





# CMD üzerinde Derlenir ve Çalıştırılır (3/3)



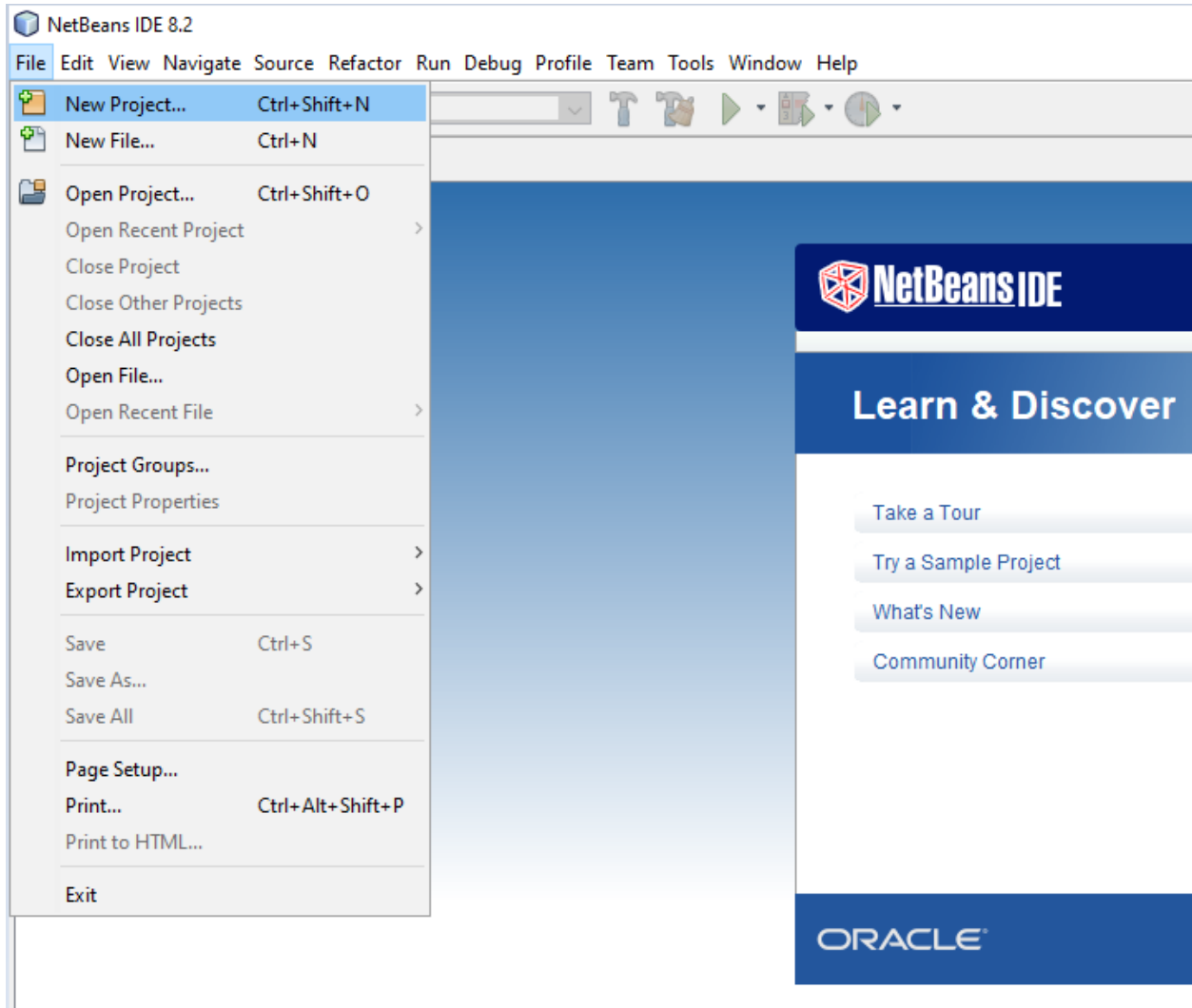
```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Busra>cd ../..Programlar
C:\Programlar>javac Yaz.java
C:\Programlar>java Yaz
Merhaba Java :)
Merhaba Dünya :)
Java öğreniyorum!
C:\Programlar>
```

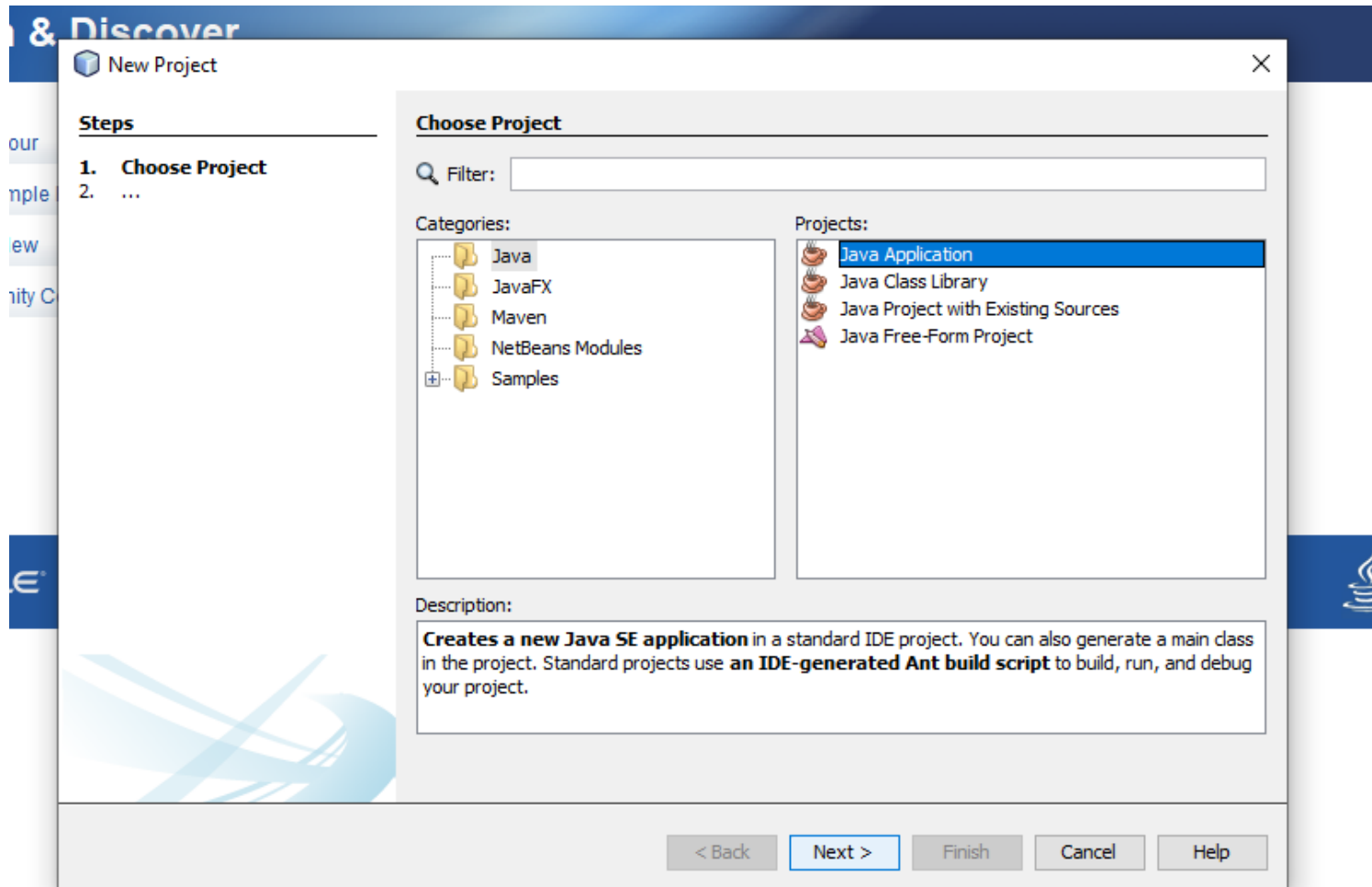
Java compiler → Byte Code (class file) is created

JVM (interpreter) → Byte Code is executed

# Netbeans IDE



# Netbeans IDE (cont.)



# Netbeans IDE (cont.)



**New Java Application**

**Steps**

1. Choose Project
2. **Name and Location**

**Name and Location**

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

< Back Next > **Finish** Cancel Help

Projemizin  
kaydolduğu path

Yaz.java

Projects × Files Services —

Yaz

- Source Packages
  - yaz
    - Yaz.java
- Libraries

Start Page × Yaz.java ×

Source History

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package yaz;
7
8  /**
9   *
10  * @author user
11  */
12  public class Yaz {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19      }
20
21  }
22
```

Navigator ×

Members <empty>

- Yaz
  - main(String[] args)



The screenshot displays an IDE interface with the following components:

- Projects Panel (Top Left):** Shows a project named 'Yaz' containing 'Source Packages' (with a sub-package 'yaz' and file 'Yaz.java'), 'Test Packages', 'Libraries', and 'Test Libraries'.
- Editors (Top):** The 'Yaz.java' file is open, showing the following code:

```
1  /*
2  * To change this license header, choose License Headers in Project Pr
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package yaz;
7
8  /**
9   *
10  * @author user
11  */
12  public class Yaz {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19
20          System.out.println("Merhaba Java :)");
21          System.out.println("Merhaba Dünya :)");
22          System.out.println("Java Öğreniyorum!");
23
24      }
25
26  }
```
- Navigator (Bottom Left):** Shows the 'Yaz' project with the 'main' method selected.
- Sticky Note (Top Right):** A yellow note with the text 'DO IT NOW!' pinned to it.

# Syntax Hatalarını Kontrol Etme



Syntax  
Hataları

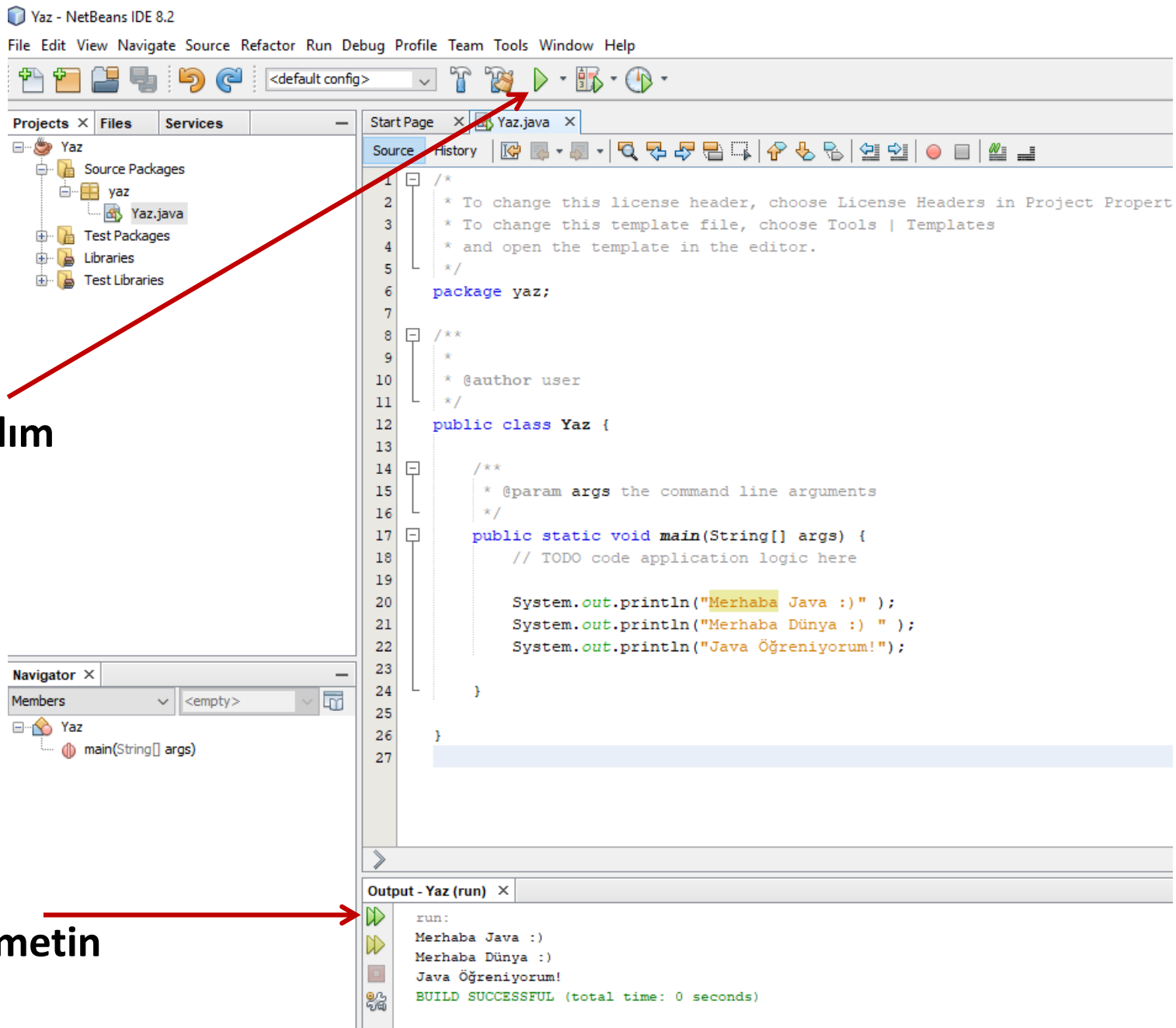
```
17 public static void main(String[] args) {  
18     // TODO code application logic here  
19  
20     System.out.println(Merhaba Java :) " );  
21     System.out.println("Merhaba Dünya :) " );  
22     System.out.println("Java Öğreniyorum!")  
23 }
```

Hata Örneği: komut satırı  
sonunda ; eklenmesi  
gerekmektedir

```
17 public static void main(String[] args) {  
18     // TODO code application logic here  
19  
20     System.out.println(Merhaba Java :) " );  
21     System.out.println("Merhaba Dünya :) " );  
22     System.out.println("Java Öğreniyorum!")  
23  
24 }  
25  
26 }
```

Şimdi, çalıştıralım  
Run Project





Çıktı Ekranımız;  
Programımızın metin  
çıktıları burada  
gösterilecektir



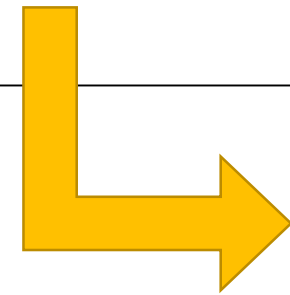
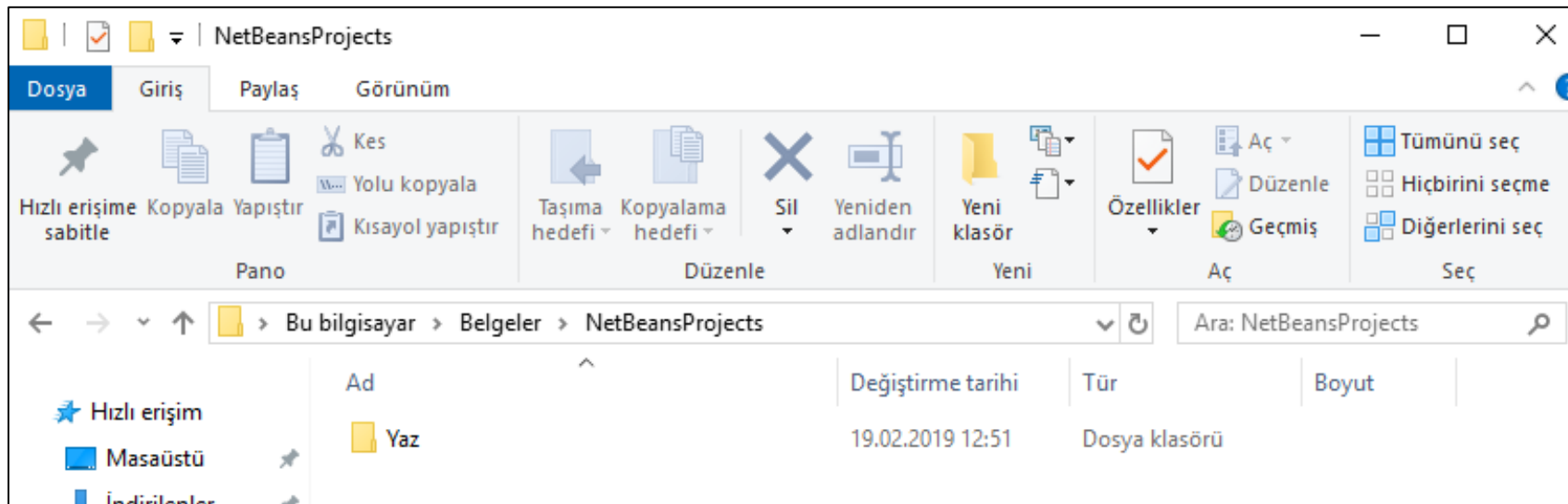


yaz.Yaz >

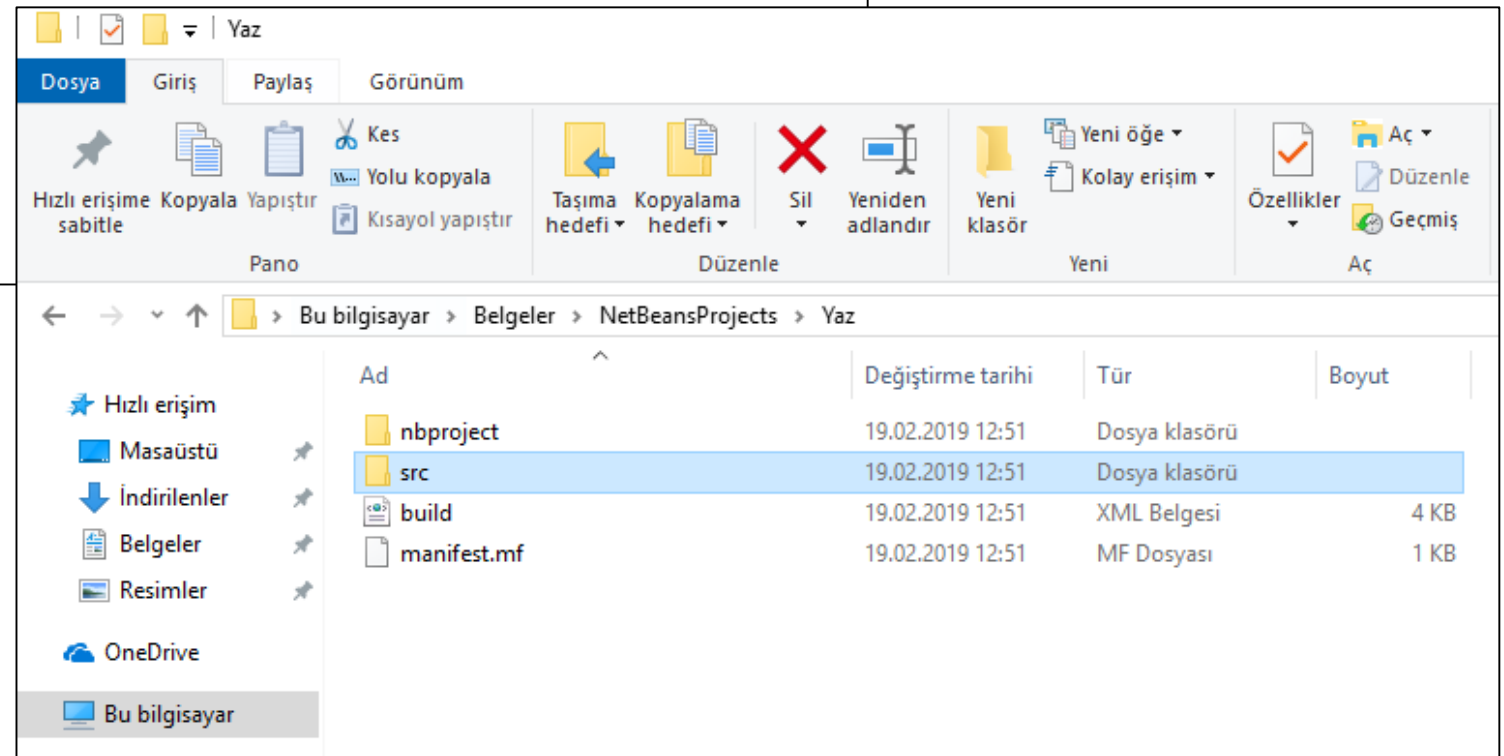
Output - Yaz (run) X

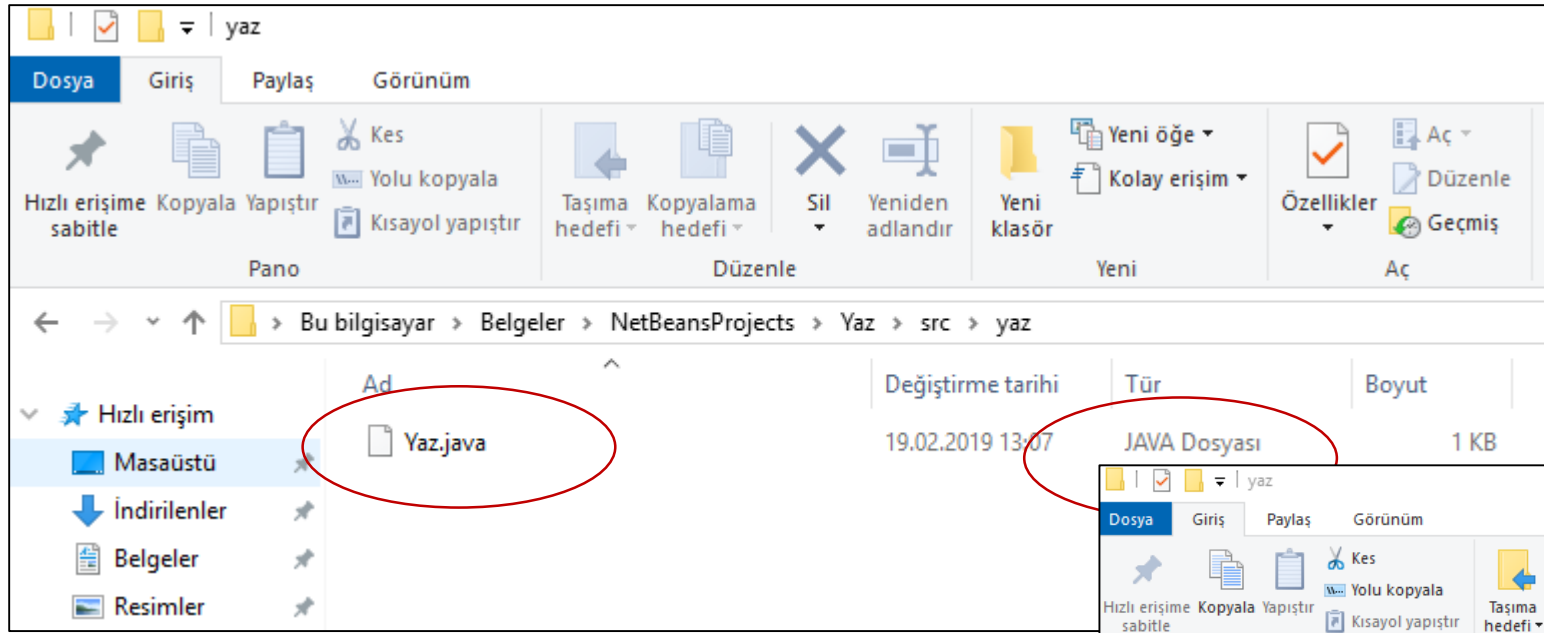
  
  
  


```
run:
Merhaba Java :)
Merhaba Dünya :)
Java Öğreniyorum!
BUILD SUCCESSFUL (total time: 0 seconds)
```



*src (source files)*

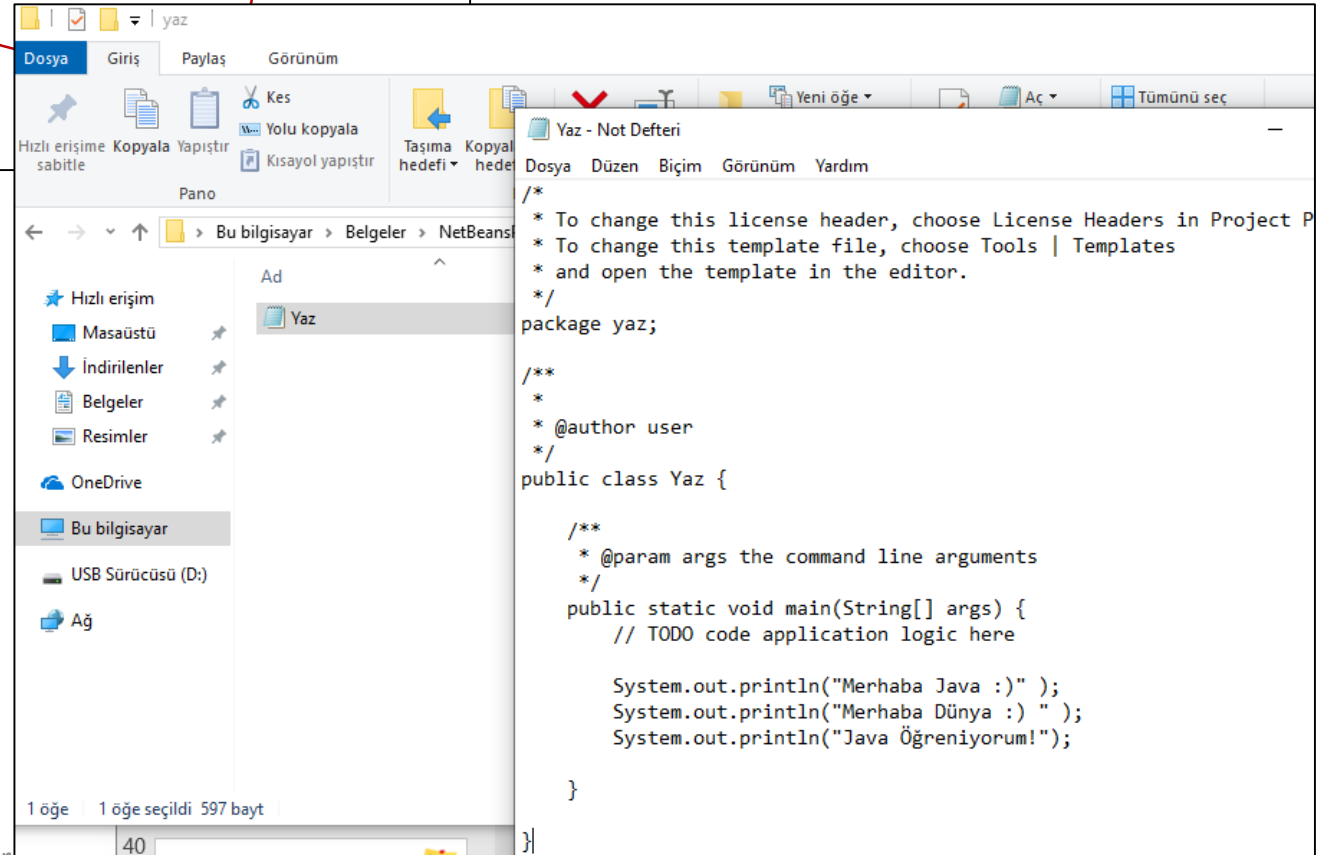




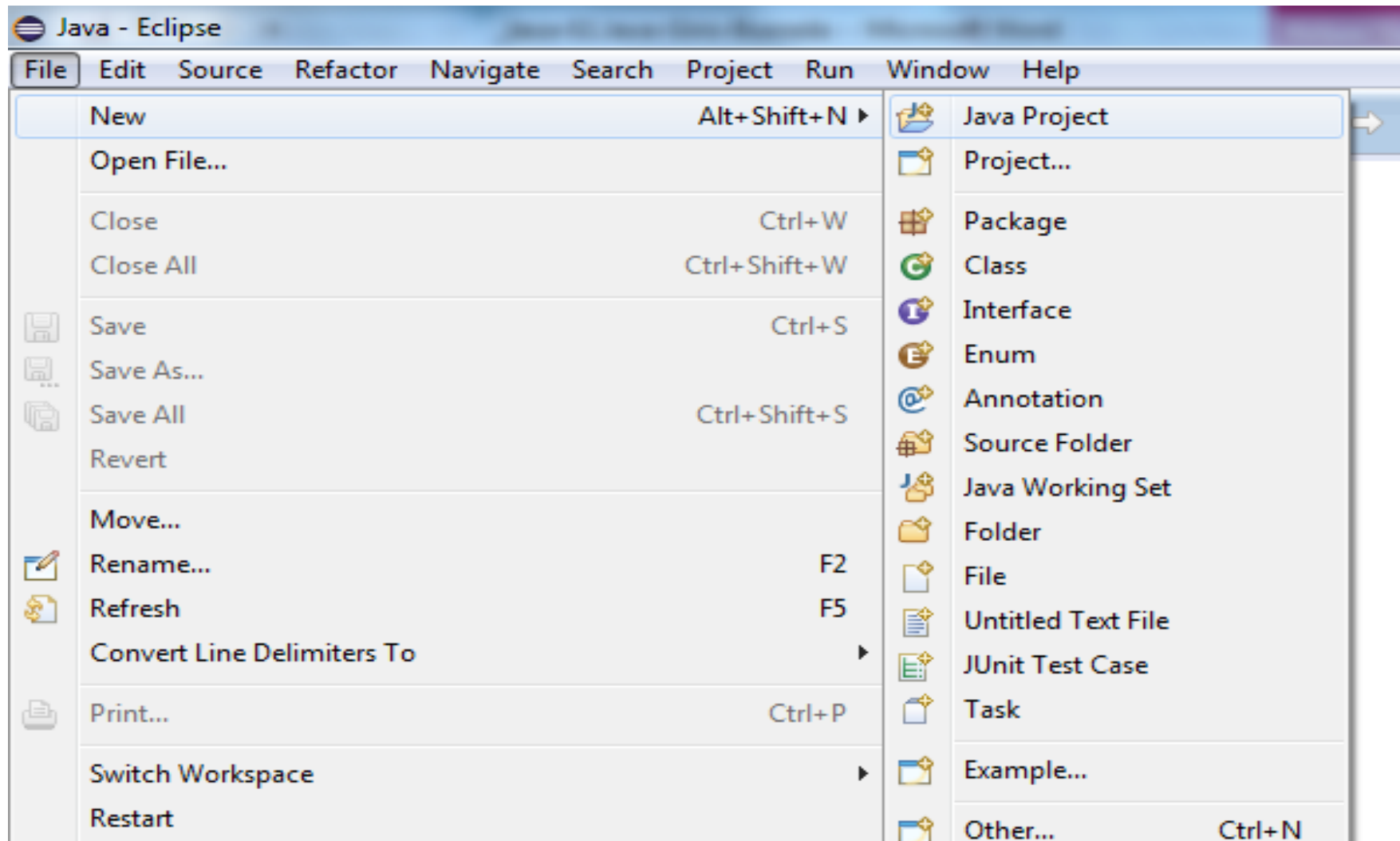
*Oluşturduğunuz .java dosyaları ilgili projenin src klasöründe bulunmaktadır.*

*Her yeni program çalışması için, yeni proje ve yeni Java application oluşturmamız gerekir.*

*Yaz.java JAVA dosyanızı Notepad ile birlikte açarak görüntülemeniz mümkün.*



# Eclipse IDE



Java - Merhaba/src/Yaz.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access Resource Java

\*Yaz.java

```
public class Yaz
{
    public static void main( String[] args )
    {
        System.out.println("Merhaba Java :) ");
        System.out.println("Merhaba Dünya :) ");
        System.out.println("Java Öğreniyorum ! ");
    }
}
```

Problems Javadoc Declaration Console

<terminated> Yaz [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (15 Kas 2015 02:16:20)

Merhaba Java :)  
Merhaba Dünya :)  
Java Öğreniyorum !

Writable Smart Insert 13 : 1



# println( ) ve print( )

- **print** metodu **println** metodu ile aynı işlemi gerçekleştirir, tek farkı:
  - **println** metni görüntüledikten sonra, gösterge bir alt yeni satıra geçer.
  - **print** metni görüntüledikten sonra, gösterge aynı satırda bekler.

Komut	Ekran	Açıklama
<code>System.out.print( "A" )</code>	A_	Ekrana yazma işlemi bittikten sonra, göstergenin ekrandaki pozisyonu _ ile işaretlenmiştir.
<code>System.out.println( "A" );</code>	A —	

# Değişken (Variable) Tanımlama

- Programlarımızda işlemler gerçekleştirmek için, **değişken** tanımlamaları yapılır.
  - Tanımlanan değişken, bilgisayarın belleğinde saklanır.
  - Değişkenler için açıklayıcı isimler kullanınız: örneğin yarıçap için yarıcap ve alan için alan gibi.
  - Değişkenler tanımlanırken, veri tipide belirlenir; integer , floating (noktalı) sayı, karakter, mantıksal (boolean) değer vb.

**veri\_tipi degerAdi;**

01	<code>int i;</code>	i integer değişken tanımlaması
02	<code>int a, b;</code>	a ve b integer değişkenleri tanımlanması
03	<code>double j;</code>	j double değişkeninin tanımlanması
04	<code>char m, k;</code>	m ve k karakter değişkenlerinin tanımlanması

# Değişkenlere Değer Atama

- Değer atama (**assignment statement**) komutu, bir değer tanımlanması işlemidir.

01	<code>int i;</code>	i değişkenin tanımlanması
02	<code>i = 1;</code>	Sabit bir değer (1), i değişkenine atanması
03	<code>int m = 5;</code>	m değişkenin tanımlanması ve sabit bir değerin (5) atanması
04	<code>int x = 4, y = 6;</code>	x değişkenin tanımlanması ve sabit bir değerin (4) atanması, y değişkenin tanımlanması ve sabit bir değerin (6) atanması
05	<code>int k = i + j + m;</code>	k değişkenin tanımlanması ve (i+j+m) işlem (expression) sonucunun atanması
06	<code>int a;</code>	a değişkenin tanımlanması
07	<code>a = 5 * ( 3 + 2 );</code>	İşlem sonucunun a değişkenine atanması
08	<code>int b = 5 * 3 + 2;</code>	b değişkenin tanımlanması ve işlem (expression) sonucunun atanması



# Genel Sayısal İşlemler

Operatör	Açıklama	Örnek	Sonuç
+	Toplama	$13 + 5$	18
	Pozitif sayıları belirtme	+5	+5
-	Çıkarma	$13 - 5$	-2
	Negatif sayıları belirtme	-5	-5
*	Çarpma	$13 * 5$	65
/	Kalansız bölme	$13 / 5$	2
%	Tamsayı bölmeden kalan	$13 \bmod 5$	3

# Algoritmalar

- Program yazmak veya kodlamak, algoritma tasarımı yapmayı ve algoritmaların koda veya program komutlarına dönüştürülmesini gerektirir.
- *Algoritma*, bir problemin nasıl çözümleneceğine dair yapılması gereken adımları ifade eder. Aynı zamanda programcıya, programı bir programlama dilinde yazmadan önce planlaması konusunda yardımcı olur.
- Algoritmalar, doğal bir dil veya *pseudocode* (yalancı kod -biraz programlama komutu içeren doğal bir dil) kullanarak oluşturulabilir.

# AlanHesapla.java

- Yarıçapı 20 olan bir dairenin alanını hesaplayan bir program yazalım!
- Algoritma Tasarımı:
  - (1) Dairenin yarıçapını belirle: bir değişken tanımla ve değer ata
  - (2) Daire alanı hesaplamadan önce: bir değişken tanımla
  - (3) Şimdi, dairenin alanını hesapla ve sonucunu değer olarak değişkene ata  
$$alan = yarıcap * yarıcap * 3.14$$
  - (4) Sonucu ekrana yazdır; sonucu görüntüle
- Bir problemi çözümlerken öncelikle algoritma tasarımı yapmanız faydalı olacaktır 😊

# Algoritma Tasarımı

```
01 public class AlanHesapla {
02     public static void main( String[] args ) {
03
04         // yaricap double değişkeni tanımla ve 20 değerini ata
05         // alan double değişkeni tanımla
06
07         // alan hesapla ve sonucu değer olarak ata
08
09         // Sonucu yazdır veya görüntüle (alan)
10
11     }
12 }
```

Özel anlamlı bir kelime olan double veri tipi, yaricap ve alan değerleri için noktalı, ondalıklı sayı veri tipinde bilgisayarın belleğinde değer saklanmasını sağlar.

```
01 public class AlanHesapla {
02     public static void main( String[] args ) {
03
04         double yaricap;
05         yaricap= 20;
06
07         double alan;
08
09         alan = yaricap * yaricap * 3.14;
10
11         System.out.println("Dairenin Alanı " + alan);
12         System.out.println(yaricap + " yarıçapındaki dairenin alanı " + alan);
13     }
14 }
```

```
01 Dairenin alanı 1256
02 20 yarıçapındaki dairenin alanı 1256
```

Projects

Files

Services

DaireAlan

Source Packages

dairealan

DaireAlan.java

Libraries

Yaz

Source Packages

yaz

Yaz.java

Test Packages

Libraries

Test Libraries

Start Page

Yaz.java

DaireAlan.java

Source

History

1

6

7

8

12

13

14

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

...5 lines

package dairealan;

/\*\*...4 lines \*/

public class DaireAlan {

/\*\*...3 lines \*/

public static void main(String[] args) {

double yaricap;

yaricap= 20;

double alan;

alan = yaricap \* yaricap \* 3.14;

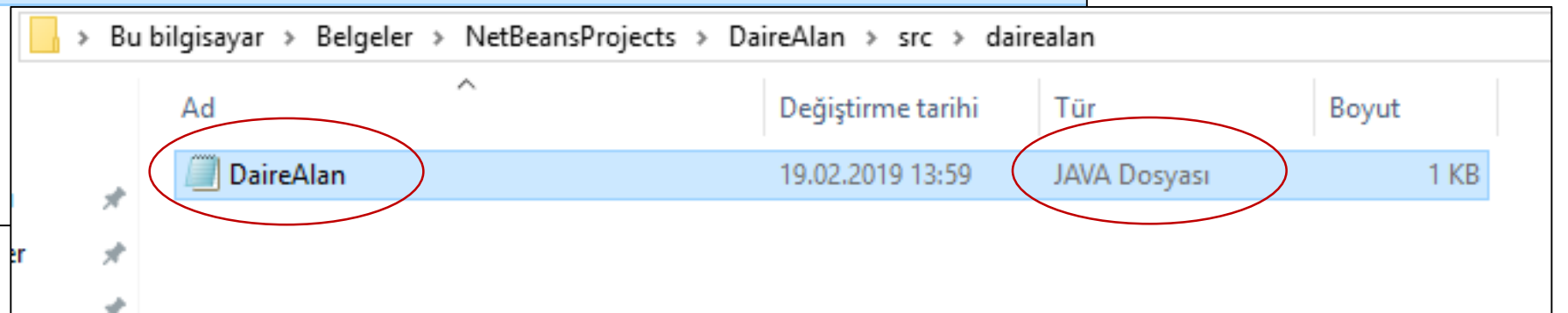
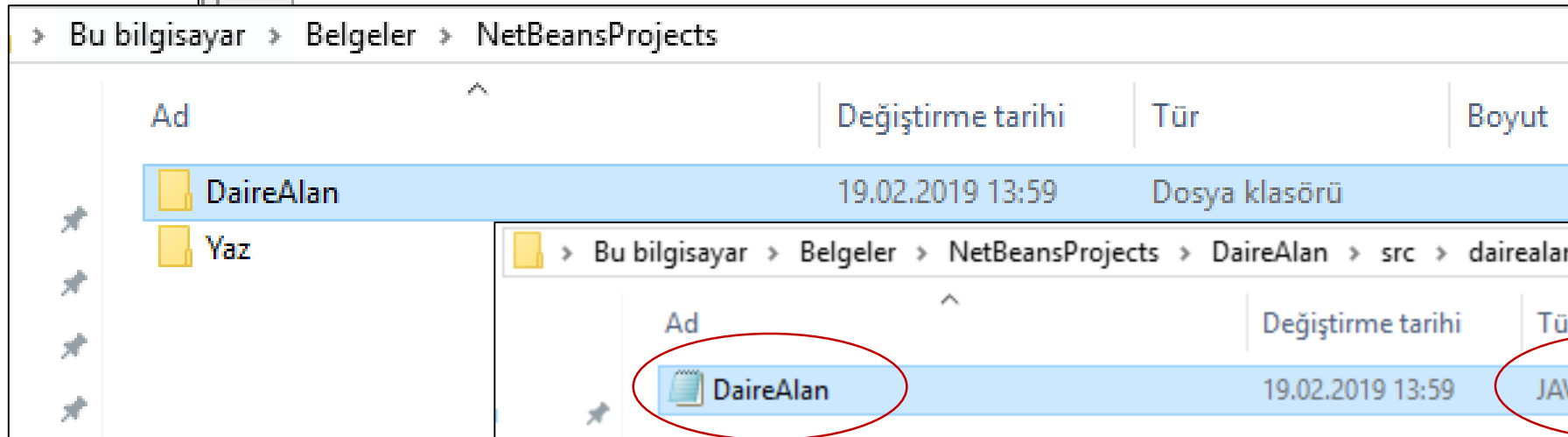
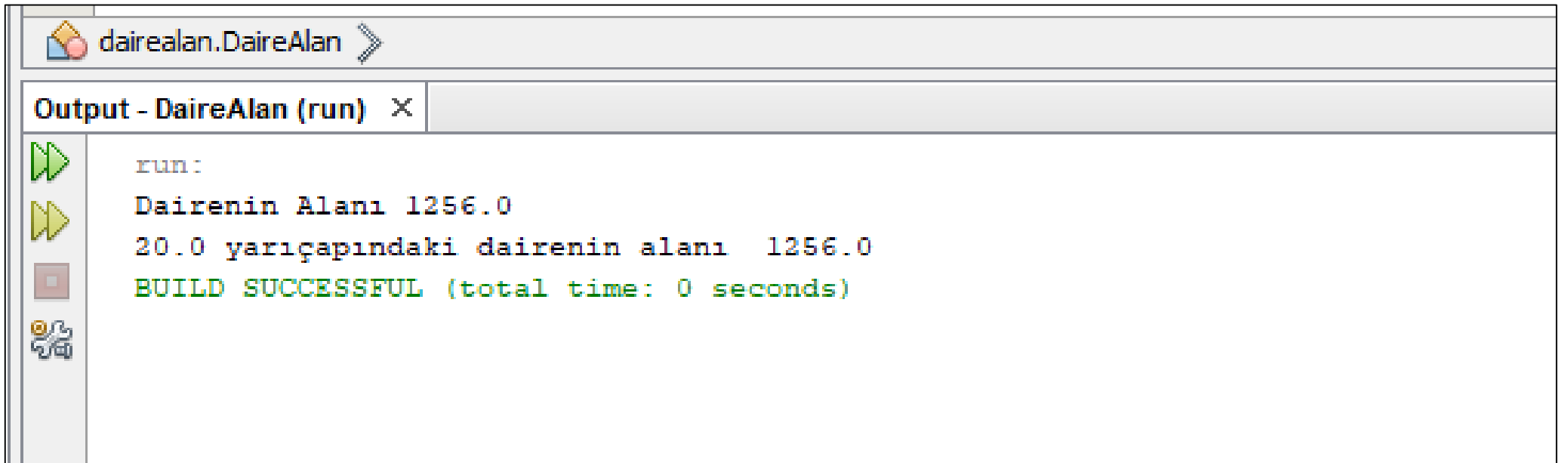
System.out.println("Dairenin Alanı " + alan);

System.out.println(yaricap + " yarıçapındaki dairenin alanı " + alan);

}

}

Navigatör



# String Bağlama Operatörü ve Toplama Sembolü

- Artı işaretinin (+) Java'da iki önemli anlamı vardır: Biri toplama işlemi gerçekleştirmek ve diğeri de Stringleri birbirine bağlamak (*string concatenation operator*) amaçlı kullanılabilir.

```
System.out.println( "Introduction to Java Programming, by Y. Daniel Liang" );
```

```
System.out.println( "Introduction to Java Programming," + " by Y. Daniel Liang" );
```

```
System.out.println( "Sayılar= " + sayi1 );
```

```
System.out.println( "Sayılar= " + sayi1 + " " + sayi2 + " " + sayi3 );
```



# Java'da Özel Anlamalı Kelimeler

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	<b>double</b>	implements	protected	throw
byte	else	import	<b>public</b>	throws
case	enum	instanceof	return	transient
catch	extends	<b>int</b>	short	try
char	final	interface	<b>static</b>	<b>void</b>
<b>class</b>	finally	long	strictfp	volatile
const	float	native	super	while

# Çalışma Soruları

- Carpim.java – İki integer (tamsayı) değişkenin çarpımını hesaplayan bir program oluşturunuz; değişkenlerin değerlerini 20 ve 40 olarak atayabilirsiniz. *Algoritma tasarımı yapmayı unutmayınız.*

01	20 ile 40 değerlerinin çarpım sonucu 800
----	--

- Ortalama.java – Üç integer (tamsayı) değişkenin ortalamasını hesaplayan bir program oluşturunuz; değişkenlerin değerlerini 10, 20, 30 olarak atayabilirsiniz. *Algoritma tasarımı yapmayı unutmayınız.*

01	10, 20 ve 30 değerlerinin ortalaması 20.0
----	---

*Any Questions?*