



# Eclipse

## Bilgisayar Programlama

# Eclipse nedir?

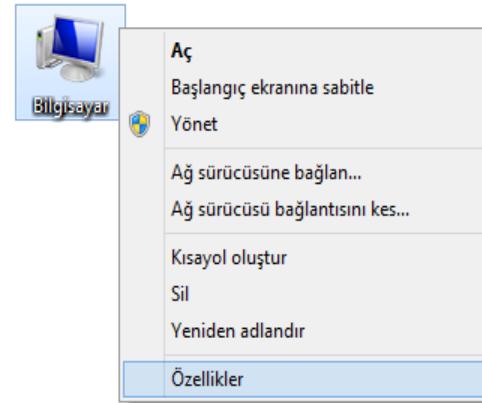


- Eclipse, açık kaynak kodlu bir yazılım geliştirme ortamıdır.
- Hızlı arayüzü, sık görünümü ve çok kuvvetli özellikleriyle kısa zamanda Java geliştiricileri arasında en popüler geliştirme ortamı olmuştur.
- En son kararlı versiyonu **ECLIPSE IDE 2019-12**'dir.



# Eclipse Programını İndirmek

- <http://www.eclipse.org/downloads/> adresine girin.
- İşletim sisteminizin türüne (32 bit/ 64 bit) göre programı indirin.
- İşletim sisteminizin türüne "Bilgisayar"a sağ tıklayıp "Özellikler"den bakabilirsiniz.



## Sistem

Üretici:	Samsung Electronics
Derecelendirme:	<b>5,9</b> Windows Deneyimi Dizini
İşlemci:	Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz 2.50 GHz
Yüklenen bellek (RAM):	6.00 GB (kullanılabilir miktar: 5.89 GB)
Sistem türü:	<b>64 bit İşletim Sistemi, x64 tabanlı işlemci</b>
Kalem ve Dokunma:	Bu Görüntü Biriminde Kalem Girdisi veya Dokunarak Giriş yok



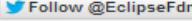
# Eclipse Programını İndirmek

Eclipse Downloads

www.eclipse.org/downloads/

Visit other Eclipse Sites

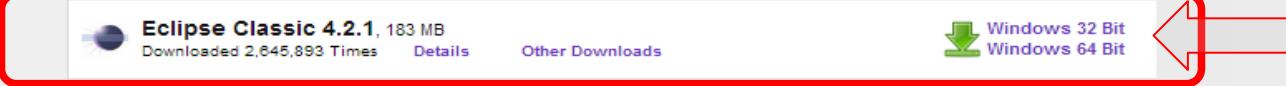
Home Downloads Users Members Committers Resources Projects About Us Google Custom Search Search

Packages Developer Builds Projects   38b

Eclipse Juno (4.2) SR1 Packages for Windows

 **Eclipse IDE for Java EE Developers**, 221 MB  
Downloaded 4,012,069 Times

 Windows 32 Bit  Windows 64 Bit

 **Eclipse Classic 4.2.1**, 183 MB  
Downloaded 2,645,893 Times Other Downloads  Windows 32 Bit  Windows 64 Bit 

 **Eclipse IDE for Java Developers**, 150 MB  
Downloaded 1,939,727 Times  Windows 32 Bit  Windows 64 Bit

 **Eucalyptus Private and Hybrid Cloud Software**  Download

 **Eclipse IDE for C/C++ Developers**, 129 MB  
Downloaded 936,399 Times  Windows 32 Bit  Windows 64 Bit

 **Eclipse for Mobile Developers**, 144 MB  
Downloaded 634,220 Times  Windows 32 Bit  Windows 64 Bit

 **Eclipse IDE for Java and DSL Developers**, 260 MB  
Downloaded 546,047 Times  Windows 32 Bit  Windows 64 Bit

 **Eclipse Modeling Tools**, 274 MB  
Downloaded 327,434 Times  Windows 32 Bit  Windows 64 Bit



# Eclipse Programını İndirmek

## Eclipse downloads - mirror selection

All downloads are provided under the terms and conditions of the [Eclipse Foundation Software User Agreement](#) unless otherwise specified.

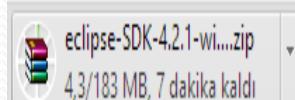
Download [eclipse-SDK-4.2.1-win32-x86\\_64.zip](#) from:



[Turkey] Linux Kullanıcıları Derneği (ftp)

Checksums: [\[MD5\]](#) [\[SHA1\]](#) BitTorrent

...or pick a mirror site below.

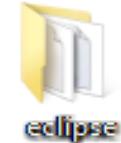
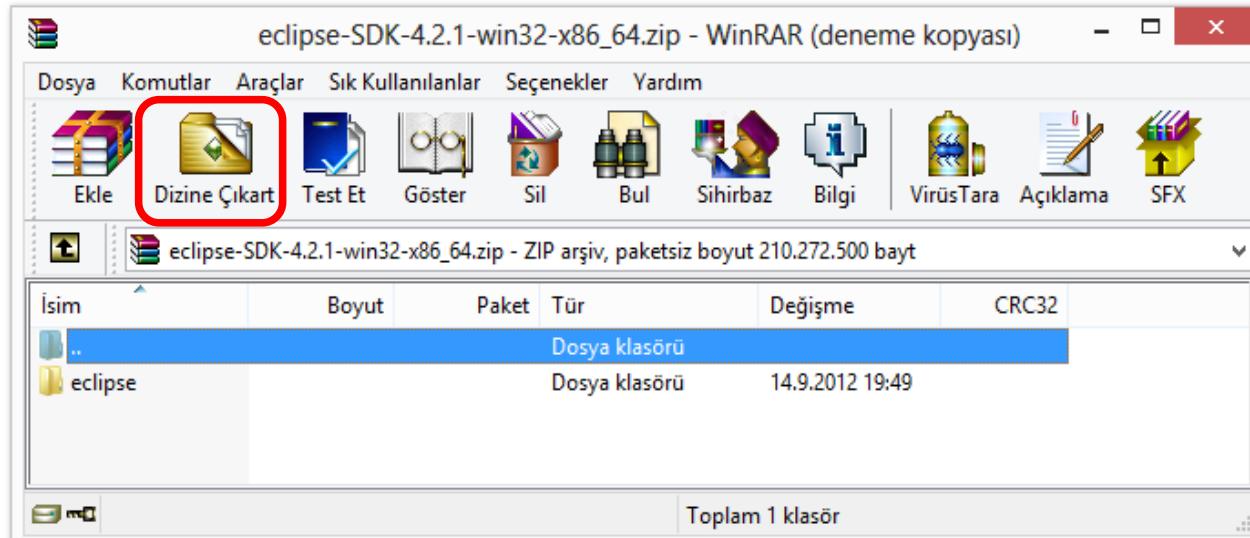


[Tüm indirilenleri göster...](#) X



# Eclipse Programını Kurmak

- Eclipse için kurulum gereklidir. Sıkıştırılmış halde indirilen dosyaları açmak yeterlidir.





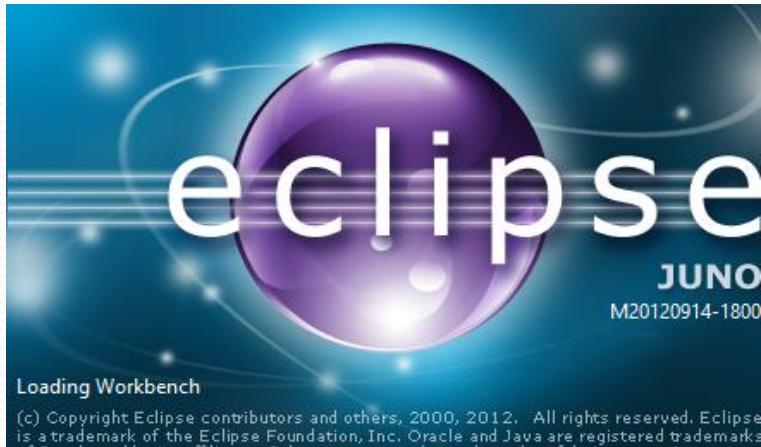
# Eclipse Programını Açımak

- Çıkarılan klasörün içindeki dosyalar aşağıdaki gibidir. Uygulama dosyasına tıklanarak eclipse çalıştırılabilir.

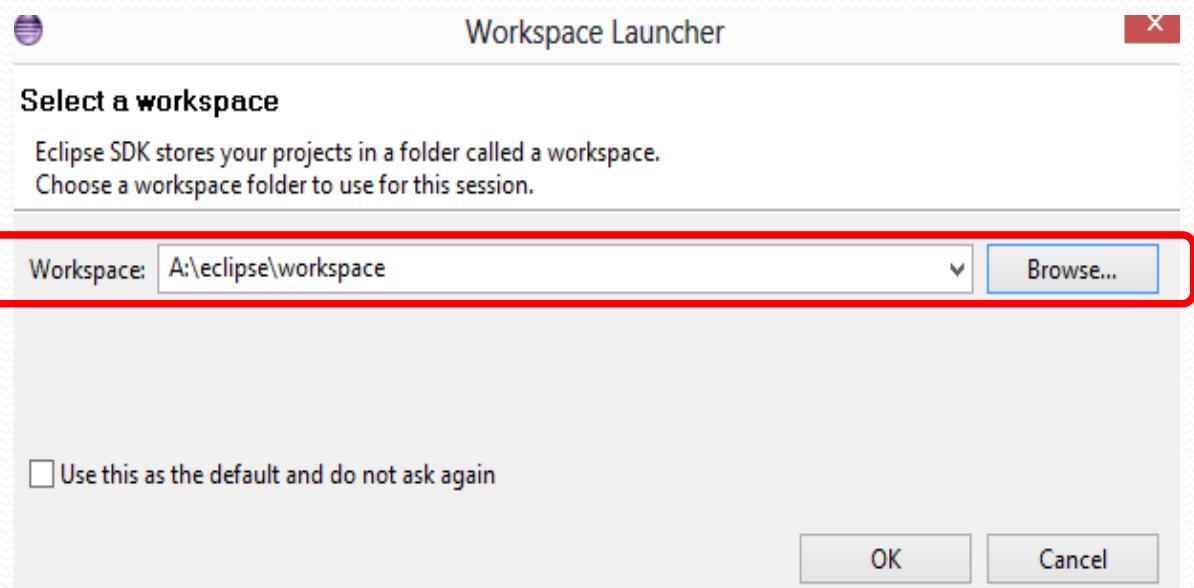
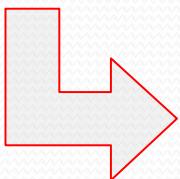
Ad	Değiştirme tarihi	Tür	Boyut
configuration	14.9.2012 20:49	Dosya klasörü	
dropins	14.9.2012 20:49	Dosya klasörü	
features	14.9.2012 20:49	Dosya klasörü	
p2	25.2.2013 18:00	Dosya klasörü	
plugins	14.9.2012 20:49	Dosya klasörü	
readme	14.9.2012 20:49	Dosya klasörü	
.eclipseproduct	14.9.2012 19:13	ECLIPSEPRODUCT...	1 KB
artifacts	14.9.2012 20:49	XML Dosyası	110 KB
eclipse	14.9.2012 19:50	Uygulama	305 KB
eclipse	14.9.2012 20:49	Yapilandırma ayar...	1 KB
eclipsec	14.9.2012 19:50	Uygulama	18 KB
epl-v10	14.9.2012 19:13	Chrome HTML Do...	17 KB
notice	14.9.2012 19:13	Chrome HTML Do...	9 KB



# Eclipse Programını Açımak



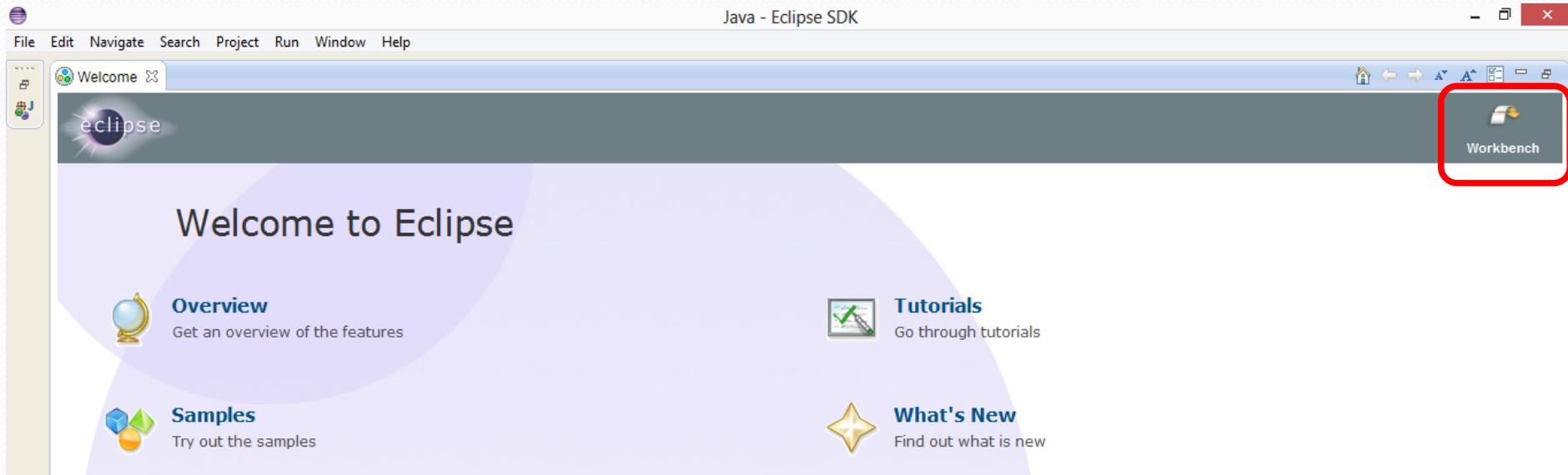
Eclipse tıklandıktan sonra açılan pencerede, çalışma dosyalarımızın kaydedileceği klasörü seçiyoruz. Bu klasör genellikle “workspace” şeklinde isimlendirilir.





# Eclipse Ortamı

- Eclipse'in açılış ekranı aşağıda görüldüğü gibidir. Sağ üst taraftaki "Workbench" butonuna tıklayarak çalışmaya başlayabiliriz.



# Eclipse Ortamı



The screenshot shows the Eclipse Java IDE interface. The title bar reads "Java - Eclipse SDK". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The left sidebar has a "Package Explorer" view showing a single project named "Java". The main workspace area is empty, with the text "Kod yazılacak kısım" (Code writing section) displayed in red. The bottom-left corner of the workspace has the text "Proje, paket ve dosyalar" (Project, package and files) in red. The bottom right corner of the workspace has the text "Hatalar ve kodun çıktısı" (Errors and code output) in red. The bottom status bar shows "0 items" in the Problems view. The bottom navigation bar includes tabs for Problems, Javadoc, and Declaration, and buttons for Back, Forward, Home, and Help.

Kod yazılacak  
kısım

Proje, paket  
ve dosyalar

Hatalar ve kodun çıktısı

# Proje Oluşturmak

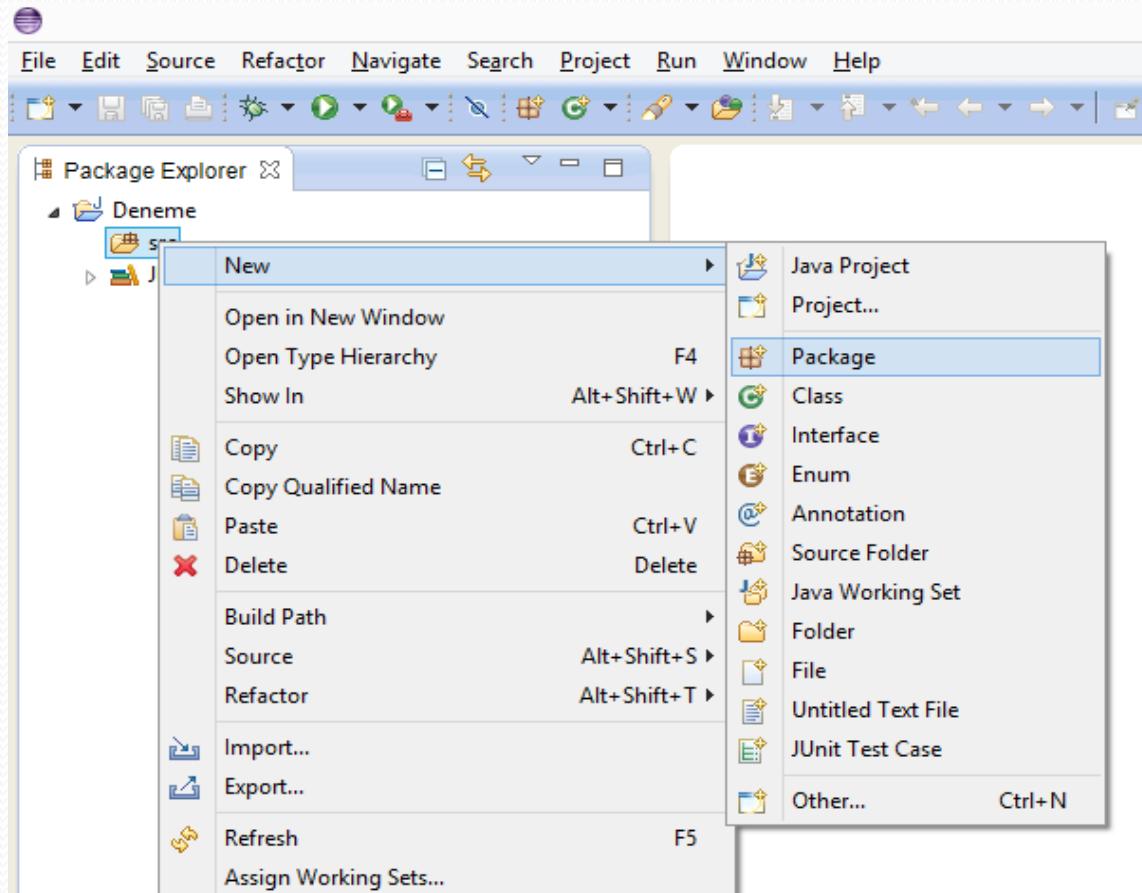


The screenshot shows the Eclipse IDE interface. On the left, the 'New Resource' dialog is open, with the 'Java Project' option selected and highlighted by a red box. A large red arrow points from this dialog to the 'New Java Project' dialog on the right. The 'New Java Project' dialog has a title bar 'New Java Project' and a sub-header 'Create a Java Project'. It contains a text field 'Project name:' with the value 'Deneme', which is also highlighted by a red box. Below it are several configuration options: a checked checkbox 'Use default location' with a location path 'A:\eclipse\workspace\Deneme' and a 'Browse...' button; a 'JRE' section with three radio button options: 'Use an execution environment JRE:' (selected, pointing to 'JavaSE-1.7'), 'Use a project specific JRE:' (pointing to 'jre7'), and 'Use default JRE (currently 'jre7')'; a 'Project layout' section with two radio button options: 'Use project folder as root for sources and class files' (unchecked) and 'Create separate folders for sources and class files' (selected); a 'Working sets' section with an unchecked checkbox 'Add project to working sets' and a 'Select...' button; and a footer with buttons '?', '< Back / Next >', 'Finish' (which is highlighted by a red box), and 'Cancel'.

Paketler ve onların altında da dosyalar oluşturabilmek için öncelikle bir proje oluşturmamız gerekiyor.



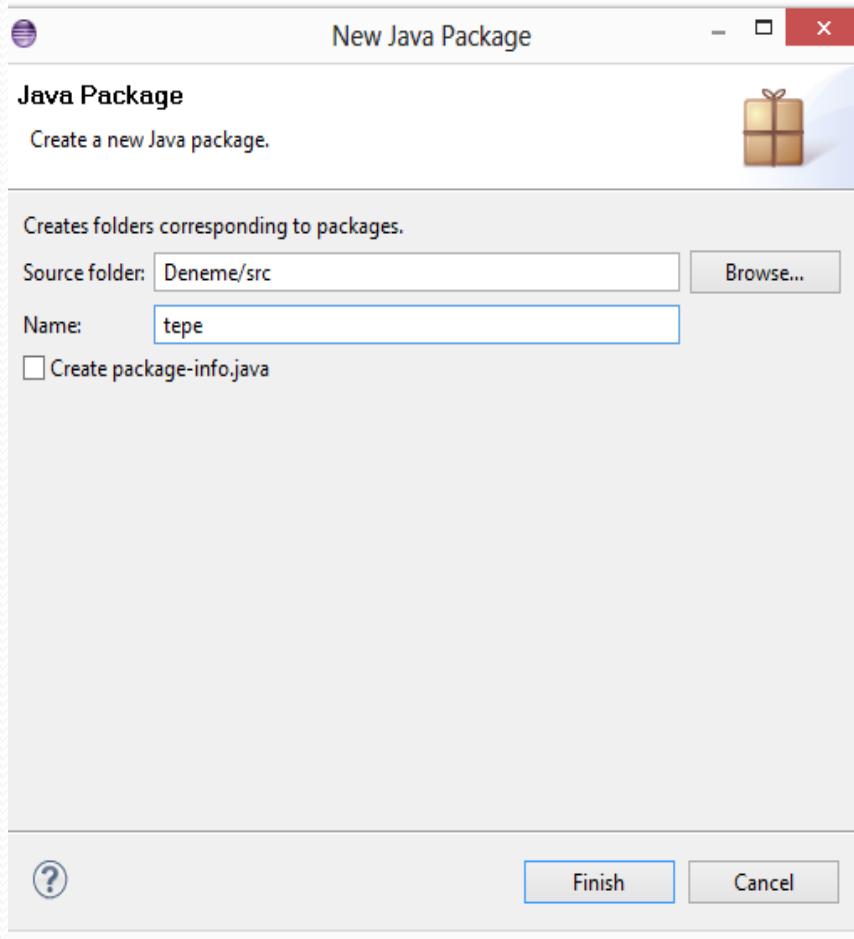
# Proje Oluşturmak



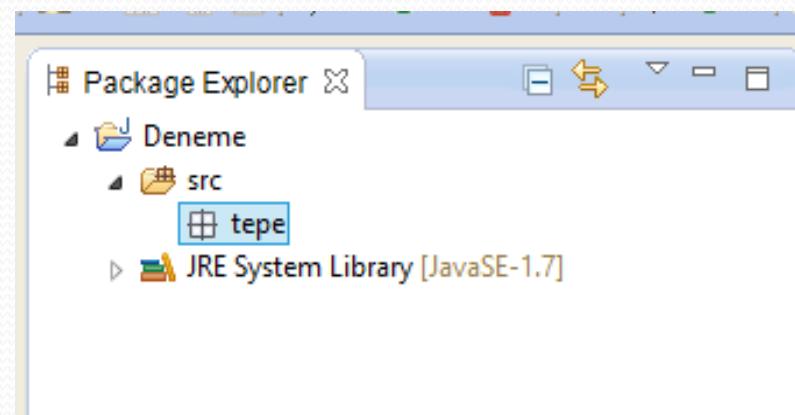
Proje oluşturuktan sonra altında yanda görüldüğü gibi **src (source)** isimli bir klasör belirecektir. Paketlerimizi ve java dosyalarımızı bu klasörün altında oluşturacağız.



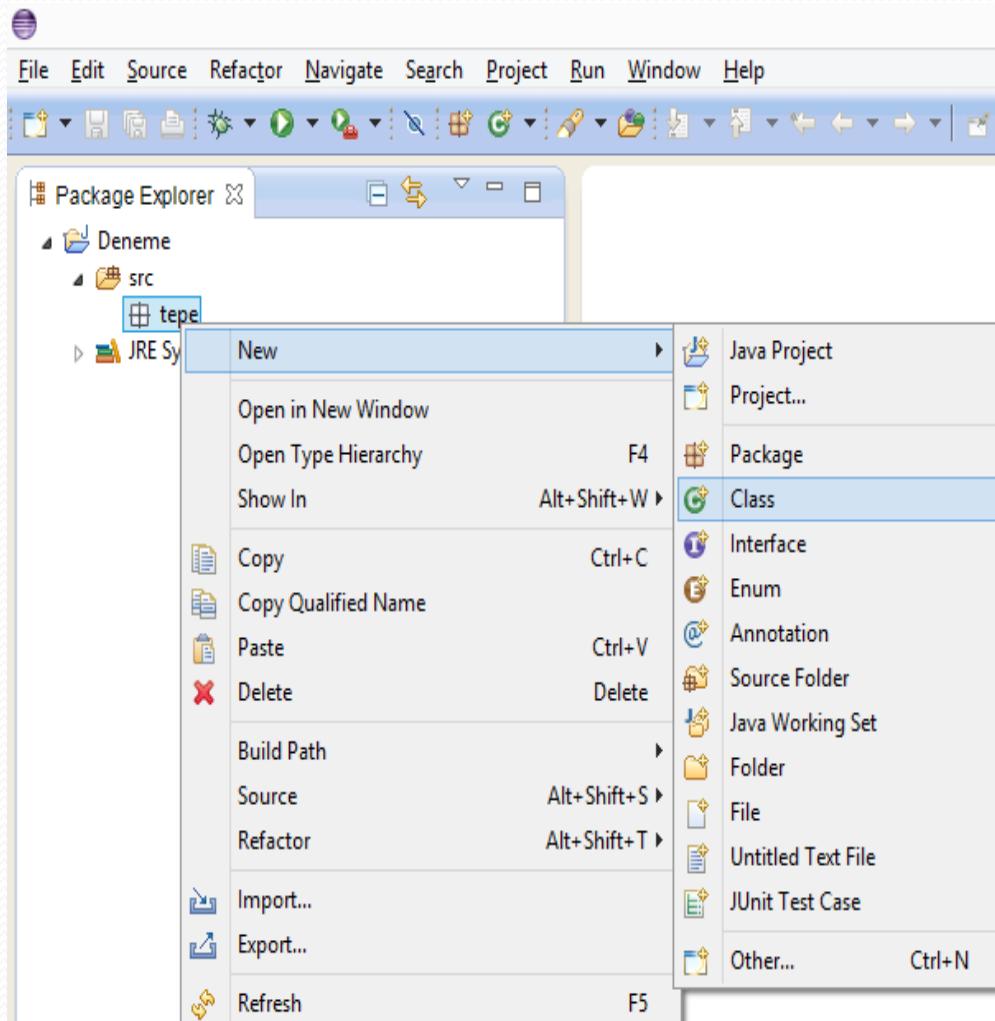
# Proje Oluşturmak



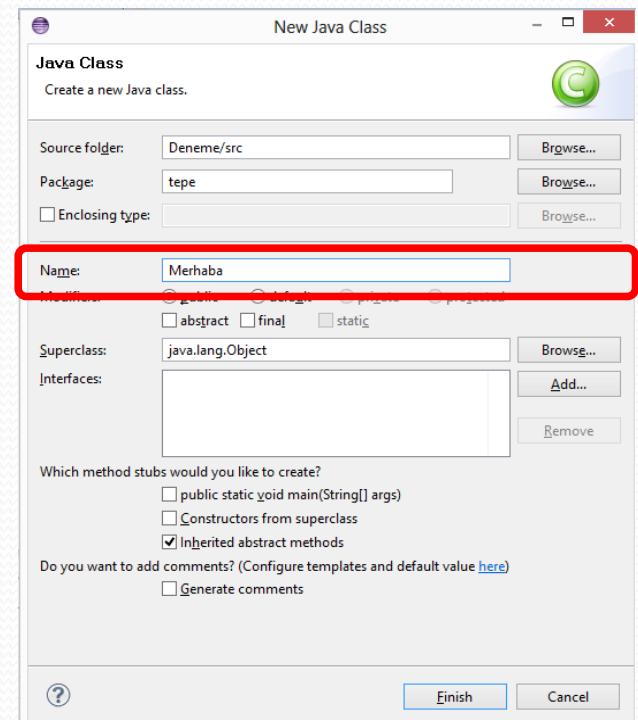
Paketimize isim verdikten sonra “Finish” butonuna tıklıyoruz ve aşağıda görüldüğü gibi paket ismi kaynak klasörün altında görünüyor.



# Dosya Oluşturmak



Paket isminin üzerine sağ tıklayarak yeni bir “Class” dosyası oluşturuyoruz ve dosyamıza isim veriyoruz.





# Kod Yazmak

- Yukarıda görüldüğü gibi artık paketimizin altında bir java dosyası görünüyor ve kod penceresinde de kaynak kodu görebiliyoruz. Artık bu orta pencere üzerinde kodumuzda istediğimiz değişiklikleri yapabiliriz.

The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. Below the menu is a toolbar with various icons. On the left, the Package Explorer view shows a project named 'Deneme' with a 'src' folder containing a 'tepe' package, which has a 'Merhaba.java' file selected. The central area is the Merhaba.java code editor, displaying the following code:

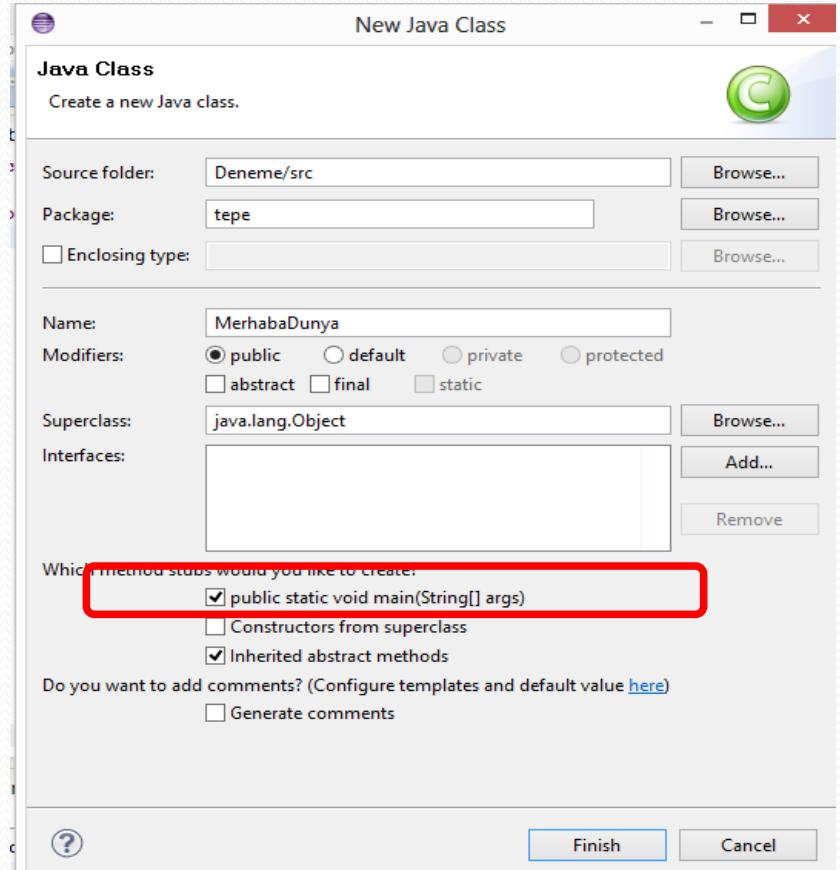
```
package tepe;

public class Merhaba { }
```



# Kodu Çalıştırmak

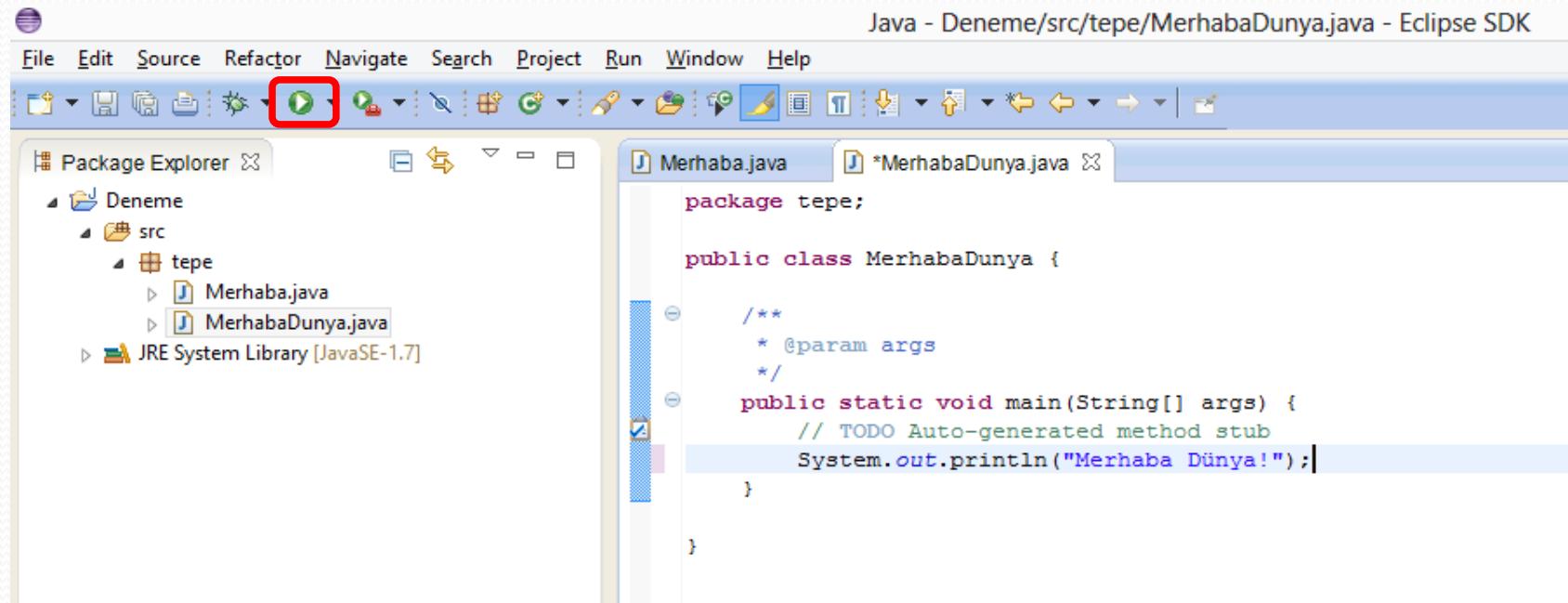
- Bir önceki yansındaki sınıf çalıştırılabilir değildi. Sınıfın çalıştırılabilir olması için oluşturma ekranında ilgili yeri tıklıyoruz ve main() bloğunun olmasını sağlıyoruz.





# Kodu Çalıştırmak

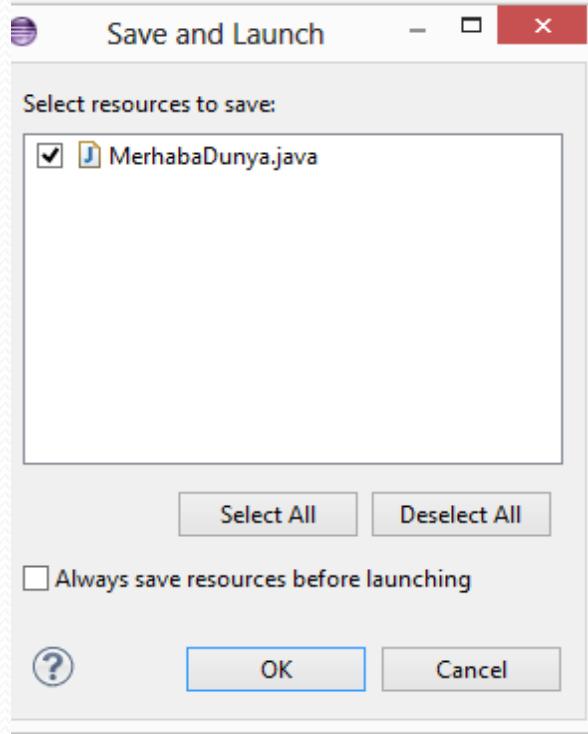
- Aşağıdaki kod çalıştırılabilir bir koddur; çünkü bir `main()` bloğuna sahiptir. Eclipse'de kodu çalıştırabilmek için araç çubuğundaki sağ ok simgesine tıklıyoruz (**Run**).





# Kodu Çalıştırmak

- Dosyayı çalıştırınca Eclipse dosyayı önce kaydetmemiz için uyarıyor. Kaydediyoruz ve çalıştırıyoruz.



```
Merhaba.java MerhabaDunya.java
package tepe;

public class MerhabaDunya {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Merhaba Dünya!");
    }
}
```

MerhabaDunya [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (25 Şub 2013 18:58:54)  
Merhaba Dünya!



# Hataları Görmek

- Eclipse'in en önemli özelliklerinden birisi koddaki hatayı anında göstermesidir. Örneğin satır sonundaki noktalı virgülü unuttuğumuz zaman görüntü aşağıdaki gibi olur:

The screenshot shows the Eclipse IDE interface with a Java file named "MerhabaDunya.java" open. The code is as follows:

```
package tepe;

public class MerhabaDunya {

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

A red circular icon with a white 'X' is positioned next to the opening brace of the main method, indicating a syntax error. The code "System.out.println("Merhaba Dünya!");" is highlighted in blue, suggesting it is the source of the error.



# Hataları Görmek

- Kodu bir önceki yansında olduğu şekliyle çalıştırduğumızda aşağıdaki gibi hata alırız:

The screenshot shows the Eclipse IDE's Console view. The title bar includes tabs for 'Problems', '@ Javadoc', 'Declaration', and 'Console'. The main area displays the following text:

```
<terminated> MerhabaDunya [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (25 Şub 2013 19:05:25)
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
  Syntax error, insert ";" to complete BlockStatements

  at tepe.MerhabaDunya.main (MerhabaDunya.java:10)
```



# Temel Hatalar

- Kod yazarken pek çok hata ile karşılaşmamız olasıdır. Bu hatalar eğer **sentaks** (sözdizim) hatası ise Eclipse bize hemen bir hata mesajı verecektir. Eğer **mantıksal** bir hata yapmış isek bunu **runtime** (çalıştırma) sırasında görürüz.
- **Runtime** hataları programa göre çok çeşitli olmakla birlikte **sentaks** hataları çok değişmemektedir.
- Bazı temel sentaks hatalarını gözden geçirelim.



# Temel Hatalar - 1

- Büyük-küçük harf hataları - 1:

The screenshot shows the Eclipse IDE interface with a Java file named "MerhabaDunya.java" open. The code contains a syntax error due to a capitalization mistake. The class declaration "Public class MerhabaDunya {" is highlighted in red, indicating a syntax error. A tooltip message "Syntax error on token \"Public\", public expected" is displayed above the error. The rest of the code is correctly written, including the main method and its body.

```
MerhabaDunya.java

package tepe;

Public class MerhabaDunya {
    Syntax error on token "Public", public expected
        public static void main(String[] args) {

            System.out.println("Merhaba Dünya!");
        }
}
```



# Temel Hatalar - 2

- Büyük-küçük harf hataları - 2:

The screenshot shows a Java code editor in the Eclipse IDE. The file is named "MerhabaDunya.java". The code is as follows:

```
package tepe;

public class MerhabaDunya {

    public static void main(string[] args) {
        string cannot be resolved to a type
        System.out.println("Merhaba Dünya!");
    }
}
```

The word "string" in the line "string cannot be resolved to a type" is highlighted in red, indicating a syntax error. A tooltip or status bar message "string cannot be resolved to a type" is displayed over the word "string".



# Temel Hatalar - 3

- Büyük-küçük harf hataları - 3:

The screenshot shows the Eclipse IDE interface with a Java file named "MerhabaDunya.java" open. The code is as follows:

```
package tepe;

public class MerhabaDunya {

    public static void main(String[] args) {
        system.out.println("Merhaba Dünya!");
    }
}
```

The word "system" in the line "system.out.println("Merhaba Dünya!");" is highlighted in red, indicating a spelling error. A tooltip box appears over the word "system" with the text "system cannot be resolved".



# Temel Hatalar - 4

- Sınıf ismini dosya isminden farklı vermek:

The screenshot shows the Eclipse IDE interface with a Java code editor. The file is named "MerhabaDunya.java". The code defines a package named "tepe" and a public class named "HelloWorld". The class contains a main method that prints "Merhaba Dünya!". A red box highlights the class definition, and a tooltip message is displayed: "The public type HelloWorld must be defined in its own file".

```
package tepe;

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



# Temel Hatalar - 5

- Eksik parantez - 1:

The screenshot shows an Eclipse IDE window with the file `*MerhabaDunya.java` open. The code is as follows:

```
package tepe;

public class MerhabaDunya {

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

A red error indicator is visible next to the opening brace of the `main` method. A tooltip message is displayed over the error: **Syntax error, insert ")" to complete MethodDeclaration**. The word `System` is also highlighted in blue.



# Temel Hatalar - 6

- Eksik parantez - 2:

The screenshot shows the Eclipse IDE interface with a Java file named "MerhabaDunya.java" open. The code is as follows:

```
package tepe;

public class MerhabaDunya {

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

A red error indicator is present at the start of the opening brace of the main method. A tooltip message is displayed over the cursor area: "Syntax error on token "[" expected after this token".



# Temel Hatalar - 7

- Eksik parantez - 3:

The screenshot shows an Eclipse IDE window with a Java file named "MerhabaDunya.java". The code is as follows:

```
package tepe;

public class MerhabaDunya {

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

A yellow tooltip box highlights the closing brace of the main method with the text: "Syntax error, insert ")" to complete ClassBody".



# Temel Hatalar - 8

- Tırnak işaretlerini unutmak:

MerhabaDunya.java

```
package tepe;

public class MerhabaDunya {

    public static void main(String[] args) {
        System.out.println(Merhaba Dünya!);
```

Multiple markers at this line

- Syntax error on token(s), misplaced construct(s)
- Merhaba cannot be resolved to a variable
- Syntax error, insert ";" to complete BlockStatements
- Syntax error, insert "AssignmentOperator Expression" to complete Assignment



# Debug



# Debug Nedir?

- Debug etmek, hatalardan arındırmak anlamına gelmektedir.
- Bu işlemin özelliği programın çalışması esnasında hataları görmektir.
- Debug işlemi ile programımızı adım adım çalıştırabilir, her bir adımda bellekteki değişkenleri, bunların içeriklerini vs. görebiliriz.
- Eclipse'in debug özelliğini kullanmak için basit dört işlem yapan bir program yazalım.

# Debug Etmek

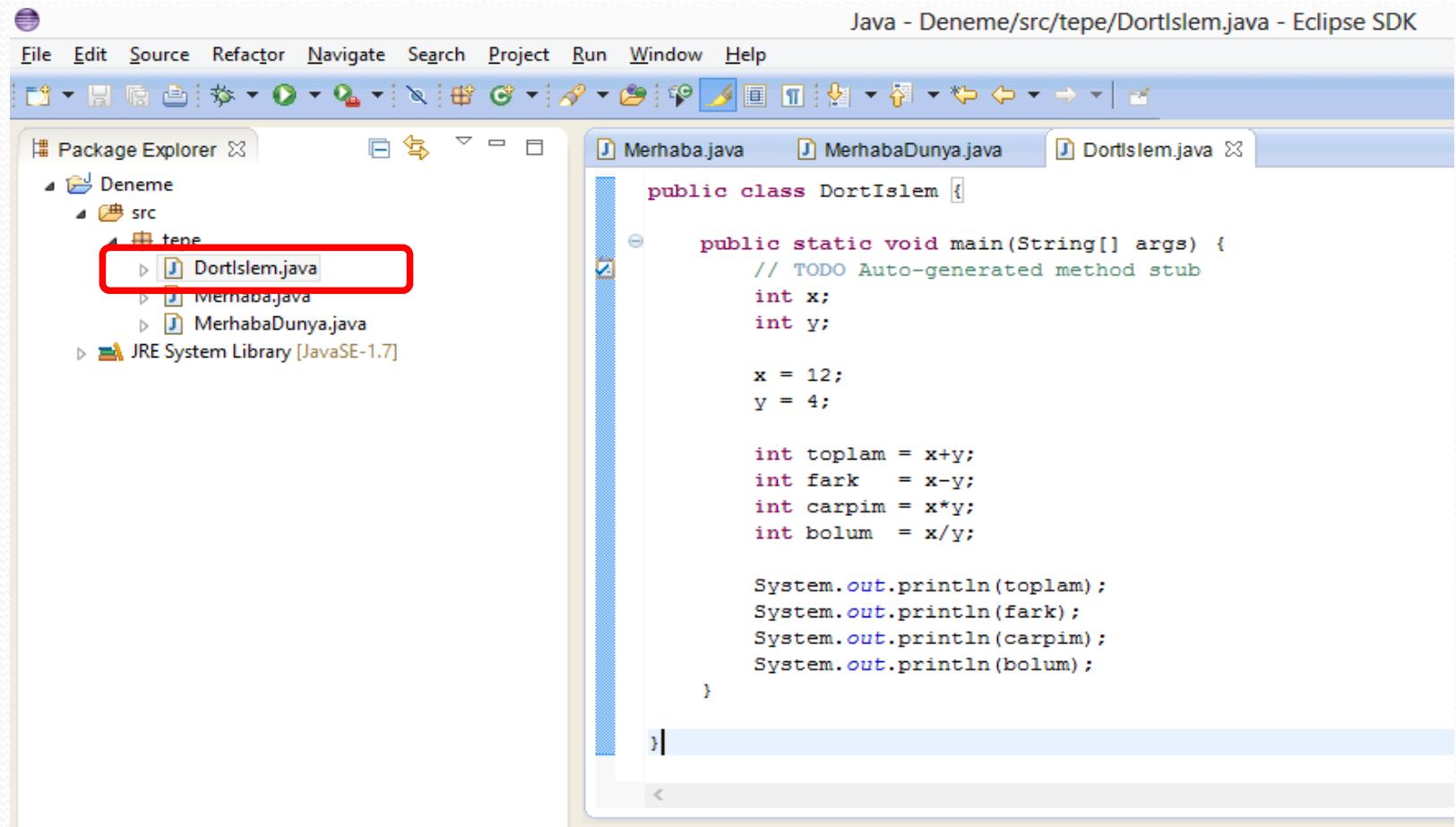


Java - Deneme/src/tepe/DortIslem.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Merhaba.java MerhabaDunya.java DortIslem.java

```
public class DortIslem {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int x;  
        int y;  
  
        x = 12;  
        y = 4;  
  
        int toplam = x+y;  
        int fark = x-y;  
        int carpim = x*y;  
        int bolum = x/y;  
  
        System.out.println(toplam);  
        System.out.println(fark);  
        System.out.println(carpim);  
        System.out.println(bolum);  
    }  
}
```





# Debug Etmek

The screenshot shows the Eclipse IDE interface. In the top tab bar, there are three tabs: Merhaba.java, MerhabaDunya.java, and Dortslem.java (the active tab). The code editor displays the following Java code:

```
public class Dortslem {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int x;  
        int y;  
  
        x = 12;  
        y = 4;  
  
        int toplam = x+y;  
        int fark = x-y;  
        int carpim = x*y;  
        int bolum = x/y;  
  
        System.out.println(toplam);  
        System.out.println(fark);  
        System.out.println(carpim);  
        System.out.println(bolum);  
    }  
}
```

In the bottom right corner of the code editor, there is a red rectangular box highlighting the 'Console' tab in the bottom-left corner of the interface.

The 'Console' tab is active, showing the output of the program execution:

```
16  
8  
48  
3
```

Programı normal bir şekilde “Run” ile çalıştırıldığımızda yandaki gibi bir çıktı elde ederiz.

# Debug Etmek



Java - Deneme/src/tepe/DortIslem.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Merhaba.java MerhabaDunya.java DortIslem.java

```
public class DortIslem {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int x;  
        int y;  
  
        x = 12;  
        y = 4;  
  
        int toplam = x+y;  
        int fark = x-y;  
        int carpim = x*y;  
        int bolum = x/y;  
  
        System.out.println(toplam);  
        System.out.println(fark);  
        System.out.println(carpim);  
        System.out.println(bolum);  
    }  
}
```

Programı debug etmek için ise “Run” butonunun hemen solundaki **böcek simgeli “Debug”** butonuna tıklıyoruz.



# Debug Etmek

The screenshot shows the Eclipse IDE interface with the following details:

- File Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Toolbar:** Standard Eclipse toolbar icons.
- Package Explorer:** Shows a project named "Deneme" with a "src" folder containing a "tepe" package. Inside "tepe", there are three files: "DortIslem.java", "Merhaba.java", and "MerhabaDunya.java". A "JRE System Library [JavaSE-1.7]" entry is also listed.
- Editor:** The "DortIslem.java" file is open, displaying the following Java code:

```
public class DortIslem {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int x;  
        int y;  
  
        x = 12;  
        y = 4;  
  
        int toplam = x+y;  
        int fark = x-y;  
        int carpim = x*y;  
        int bolum = x/y;  
  
        System.out.println(toplam);  
        System.out.println(fark);  
        System.out.println(carpim);  
        System.out.println(bolum);  
    }  
}
```
- Console:** The "Console" tab is selected, showing the output of the program:

```
<terminated> DortIslem [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (25 Şub 2013 19:36:47)  
16  
8  
48  
3
```

Herhangi bir işlem yapmadan “Debug” butonuna tıkladığımızda sanki “Run” butonuna tıklamışız gibi programı çalıştırır.



# Debug Etmek

- Programı en başından itibaren debug edebilmek için küçük bir ayar yapmamız gerekiyor.
- "Debug" butonunun yanındaki oka tıklıyoruz ve "Debug Configurations'a giriyoruz.
- Burada "**Stop in main**" kutucuğunu işaretliyoruz. Böylece, programımız çalışmaya başladıkтан sonra main() bloğuna gelince duracaktır.



# Debug Etmek

Debug Configurations

Create, manage, and run configurations

Debug a Java application



Name: Dortslem

Main Arguments JRE Classpath Source Environment Common

Project: Deneme

Main class: tepe.Dortslem

Include system libraries when searching for a main class

Include inherited mains when searching for a main class

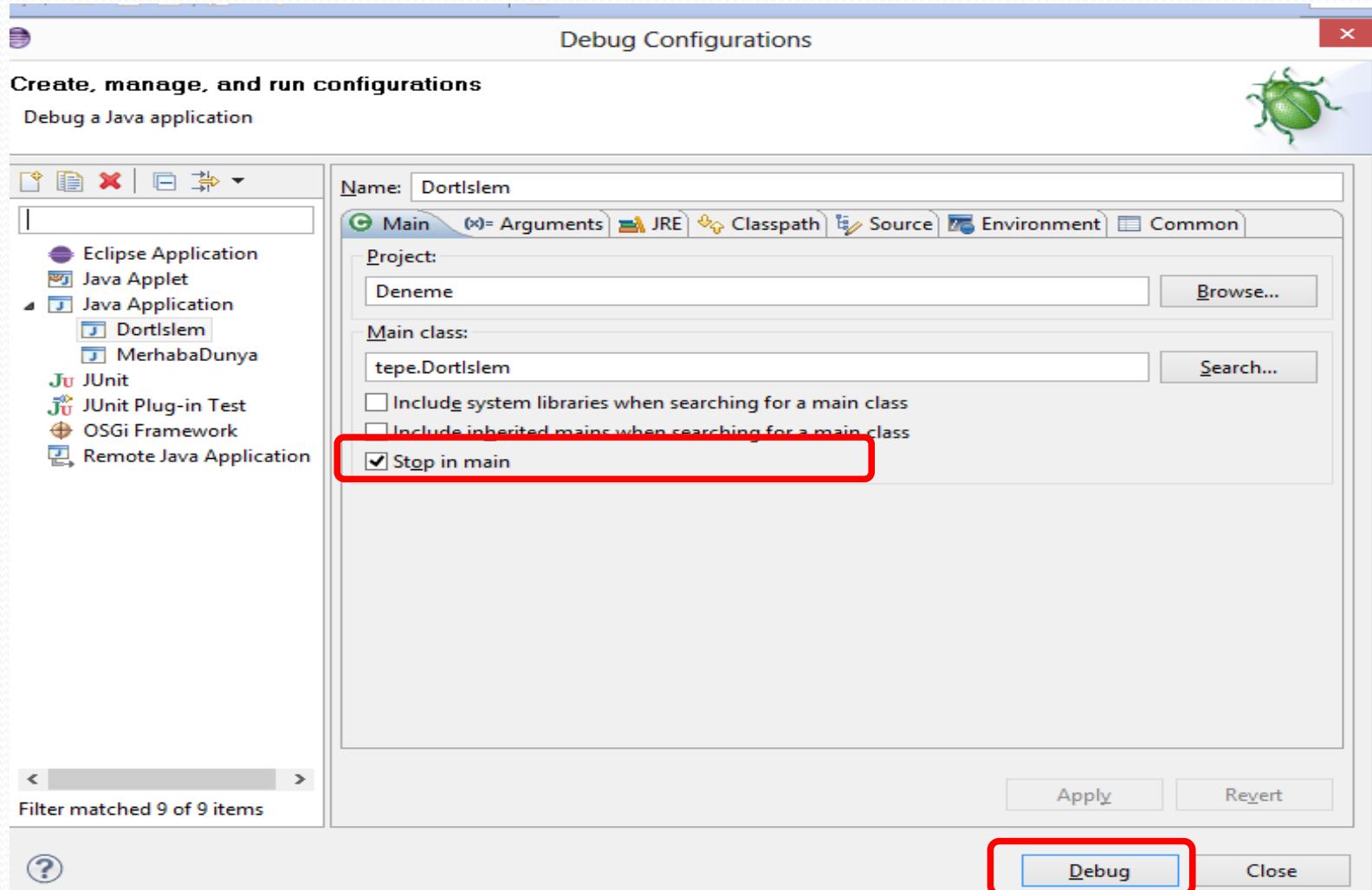
Stop in main

Apply Revert

Filter matched 9 of 9 items

?

Debug Close



The screenshot shows the 'Debug Configurations' dialog in Eclipse. On the left, there's a toolbar with icons for creating, managing, and running configurations. Below it is a tree view of configuration types: Eclipse Application, Java Applet, Java Application (with 'Dortslem' selected), JUnit, JUnit Plug-in Test, OSGi Framework, and Remote Java Application. The main area is titled 'Dortslem'. It has tabs for Main, Arguments, JRE, Classpath, Source, Environment, and Common. Under Main, the 'Project' dropdown is set to 'Deneme' and the 'Main class' dropdown is set to 'tepe.Dortslem'. There are two unchecked checkboxes: 'Include system libraries when searching for a main class' and 'Include inherited mains when searching for a main class'. A third checkbox, 'Stop in main', is checked and highlighted with a red rectangle. At the bottom right are 'Apply' and 'Revert' buttons, and at the very bottom are '?', 'Debug' (which is also highlighted with a red rectangle), and 'Close' buttons.



# Debug Etmek

- İşlemi yaptıktan sonra “Debug” butonuna bastığımızda perspektifimiz değişecektir; java perspektifinden debug perspektifine geçeriz.
- Burada perspektif, düzen, yerleşim anlamındadır.
- Bu yeni görünümde araç çubuğunda karşımıza yeni butonlar çıkacak. Bu butonları kullanarak programımızda adım adım ilerleyebiliriz.

# Debug Etmek



Örneğin ilk olarak F5 butonuna basarak bir satır komut çalıştıralım. Sağ üst taraftaki bölmede x değişkeninin değerinin 12 olduğunu görüyoruz.

Debug - Deneme/src/tepe/DortIslem.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Run Window Help

Quick Access

Debug X Step Into (F5)

DortIslem [Java Application]  
tepe.DortIslem at localhost:50792  
Thread [main] (Suspended)  
DortIslem.main(String[]) line: 11  
C:\Program Files\Java\jre7\bin\javaw.exe (25 Şub 2013 19:45:13)

(x)= Variables X Breakpoints

Name	Value
args	String[0] (id=16)
x	12

Merhaba.java MerhabaDunya.java DortIslem.java

```
public class DortIslem {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int x;  
        int y;  
  
        x = 12;  
        y = 4;  
  
        int toplam = x+y;  
    }  
}
```

Outline X tepe  
DortIslem  
main(String[]):void



# Debug Etmek

Bir sonraki adımda da y değişkeni 4 değerini alacaktır.

Debug - Deneme/src/tepe/DortIslem.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Debug

- DortIslem [Java Application]
  - tepe.DortIslem at localhost:52544
    - Thread [main] (Suspended)
      - DortIslem.main(String[]) line: 13
  - C:\Program Files\Java\jre7\bin\javaw.exe (26 Şub 2013 00:08:38)

Variables

Name	Value
args	String[0] (id=16)
x	12
y	4

Merhaba.java MerhabaDunya.java DortIslem.java

```
int x;
int y;

x = 12;
y = 4;

int toplam = x+y;
int fark = x-y;
int carpim = x*y;
int bolum = x/y;

System.out.println(toplam);
```

Outline

  - tepe
  - DortIslem
    - main(String[]): void

# Debug Etmek



Birkaç adım sonra tüm değişkenlerin değerini görebilir hale geliyoruz.

The screenshot shows the Eclipse IDE interface in the Debug perspective. The title bar reads "Debug - Deneme/src/tepe/DortIslem.java - Eclipse SDK". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations, search, and navigation. The left side features a "Debug" view showing the application "DortIslem [Java Application]" running at "localhost:52544" with a suspended thread at line 20 of "DortIslem.main(String[])". Below it is the command line "C:\Program Files\Java\jre7\bin\javaw.exe (26 Şub 2013 00:08:38)". The center of the screen displays the "Variables" view with the following table:

Name	Value
args	String[0] (id=16)
x	12
y	4
toplam	16
fark	8
carpim	48
bolum	3

The bottom left shows the code editor for "DortIslem.java" with the following code:

```
int toplam = x+y;
int fark = x-y;
int carpim = x*y;
int bolum = x/y;

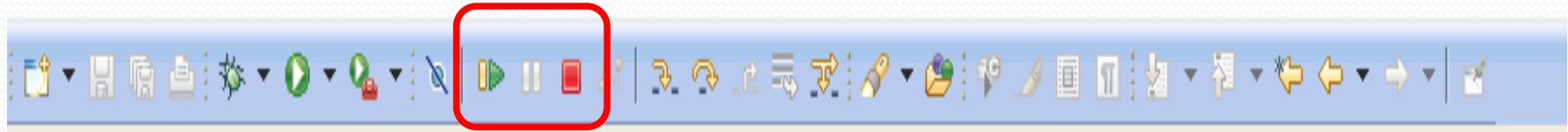
System.out.println(toplam);
System.out.println(fark);
System.out.println(carpim);
System.out.println(bolum);
```

The line "System.out.println(carpim);" is highlighted with a green background. The bottom right shows the "Outline" view, which lists the package "tepe" and the class "DortIslem" with its main method.

# Debug Etmek



- Bu şekilde tek tek tüm değişkenlerin değerleri incelenebilir, program adım adım takip edilebilir.
- Programı çalıştırmadan sonlandırmak için “**Terminate**” butonuna tıklayabiliriz.
- Programı bir sonraki breakpoint'e kadar çalıştmak için ise “**Resume**” butonuna tıklayabiliriz. Peki breakpoint ne demek?





# Debug Etmek

- Breakpoint, programda durdurup incelemek istediğimiz yere koyduğumuz işaretlerdir diyebiliriz.
- Bir satırın solunda, pencerenin kenarına çift tıkladığımızda bu satıra bir breakpoint koymuş oluyoruz.
- Bunu yaptığımız zaman ilgili yerde mavi bir daire görürüz. Breakpoint'i kaldırmak için de aynı işlemi uyguluyoruz.

```
public class DortIslem {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int x;  
        int y;  
  
        x = 12;  
        y = 4;  
  
        int toplam = x+y;  
        int fark = x-y;  
        int carpim = x*y;  
        int bolum = x/y;  
  
        System.out.println(fark);  
        System.out.println(carpim);  
        System.out.println(bolum);  
    }  
}
```



# Debug Etmek

Eğer daha önce Debug işlemini main() bloğunda bekleyecek şekilde ayarlamamış olsaydık programımızı debug eder etmez yazdırma satırına gelecektik.

Ancak daha önce bu ayarı yaptıktı ve programı debug ettiğimizde program main() içindeki ilk çalıştırılabilir satırda bekler. “Resume” butonuna bastığımızda ise yazdırma satırına gelir ki bu noktada tüm değişkenlerin değer almış olduğunu görebiliriz.

Tekrar “Resume” butonuna bastığımızda ise programın sonuna kadar çalışıp sonlandığını görürüz; çünkü programımıza başka breakpoint koymadık.

# Debug Etmek



- Debug yaparken araç çubuğunda görünen butonlar ve önemli butonların fonksiyonları aşağıdaki gibidir:

