

# Merhaba Java !

Yeni Başlayanlar için Java dökümanı



```
System.out.println("Merhaba Java");
```

Ömer Raif Tekin | raifpy | [git.io/Jf81Q](https://git.io/Jf81Q)

## Hakkında :

Doküman Libreoffice kullanılarak Gnu/Linux ortamında yazılmıştır . Oracle Java yerine Open Jdk tercih edilmiştir . Yazar izni olmadan ticari amaçlar için kullanılamaz . Dijital “.pdf” format halini istediğiniz yerde , istediğiniz kadar paylaşabilirsiniz .

## Bir takım notlar :

- PDF boyunca kullanılan işletim sistemi Gnu Linux olacaktır . Javada işletim sistemleri arasında farklılık gösteren kodlar yok , ginede kendini tekel haline getirmeye çalışan Windows işletim sistemi yerine tamamen özgür sizi takip etmeyen , **gizliliğinize saygı duyan** GNU işletim sistemini kullanabilirsiniz .
- PDF LibreOffice Writer kullanılarak yazılmıştır . Bir takım acemice hatalar olabilir ..
- PDF boyunca OpenJdk 11 kullanılacaktır . Bu sürüm ile birlikte java kodlarını kısmen de olsa derlemeden , doğrudan java ile çalıştırabiliriz (OOP programlama da derleme mecbur hale gelecektir)
- Şahsen bütün kodlarımı \*şirketlerinden kendilerinden nefret etsem de \* Visual Studio Code üzerinden yapıyorum . Aynı IDE 'yi kullanmak zorunda değilsiniz elbette
- Kodlar doğrudan kopyalanmasın , birebir yazarak öğrenilsin diyerek metin olarak değil , görsel olarak paylaşmayı tercih ettim . Kullandığım site ise carbon.now.sh ..

Bu doküman bir Java öğrencisinin öğrendiği bilgileri kendi yorumları ile defterine aktardığı notları dijital ortama aktarılmış halidir . Python3 Istihzadan etkilenen bir lise öğrencisinin Türkçe yazılı kaynak araması sonucu ortaya çıkmıştır .

Fırat Aydın (Python3\_Istihza) 'ya itafen ..

# Java Nedir

Java her işletim sisteminde bulunan , JVM yapısı sayesinde her işletim sisteminde aynı kodu çalıştırabilen 1995'de sunulan tamamen Nesne Tabanlı bir dildir .

Java kodlarının işletim sisteminde çalışabilmesi için Java Development Kit (JDK) 'nın yüklü olması gerekir .

Dünyanın en çok satılan oyunu (Minecraft) java dili ile yazılmıştır .

Java dili ile yazıldığı için bütün işletim sistemlerinde çalışabilir \* .

Java kodları hem derlenir hem interpreter tarafından yorumlanarak çalışır .

Java işlemci boyutunda derlenmediği için C türevi dillerden çok daha yavaştır .

Lakin interpreter tarafından yorumlandığı için tüm işletim sistemlerinde aynı kod çalışabilir ...

# Merhaba Java

O halde daha sonra da bol bol kullacağımız düzende ilk Merhaba Dünya örneğimizi yazalım ..



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

Haydi Kodları Açıklayalım :

```
System.out.println("Merhaba Dünya");  
// Ekrana Merhaba Dünya\n yaz
```

Kodlarımızı MerhabaJava.java adlı bir betiğe kayıt edelim ve javac ile derleyelim :

```
raif@raif:~/MerhabaJava$ javac MerhabaJava.java
```

Burada asıl dikkat etmemiz gereken 1.satırdaki "**public class MerhabaJava**" nesnesi .

Şayet sınıf adımız ile betik adı aynı olmak **zorunda** .

Yani MerhabaJava betiğimizi **MerhabaJava.java** yerine **MerhabaJava1.java** olarak kayıt edip derlemeye çalışırsam

```
raif@raif:~/MerhabaJava$ javac MerhabaJava1.java  
MerhabaJava1.java:1: error: class MerhabaJava is public, should be declared in a  
file named MerhabaJava.java  
public class MerhabaJava{  
        ^  
1 error
```

Görüldüğü üzere derleyici bize hata döndürdü .

- Java'da sık sık hata alacağız . Bu sebepten Java derleyicisinin hangi satırdan hoşlanmadığını anlamamız çok önemli

Peki öyleyse hata çıktımızda birkaç yeri vurgulayarak anlamaya çalışalım :

```
raif@raif:~/MerhabaJava$ javac MerhabaJava1.java
MerhabaJava1.java:1: error: class MerhabaJava is public, should be declared in a
file named MerhabaJava.java
public class MerhabaJava{
      ^
1 error
```

Hatamız 1.satırda , public olan sınıfın dosya ismi ile aynı olması gerektiğini ifade ediyor .

Hatalardan arandıktan sonra ilk Merhaba Dünya'mızı çalıştıralım ;

```
1) javac <betik.java>
2) java <betik>
```

- javac ile derlediğimiz java betiği .class olarak kayıt edilir . Lakin çalıştırırken .class 'ı belirtmeyiz

```
raif@raif:~/MerhabaJava$ java MerhabaJava
Merhaba Dünya
```

- *JDK yüklü herhangi bir cihazda derlenmiş (.class) dosyayı hiçbir değişiklik yapmadan çalıştırabilirsiniz*
- *Kodların sınıflarını , nesnelerini değiştirerek nerede , neyi değiştirdiğiniz zaman ,hangi hatayı alacağınızı deneyimleyebilirsiniz . Kodun bütün nitelikleri ilerledikçe yerine oturacaktır*

## Yorum Satırı

Yorum satırları kodların okunmasını kolaylaştırır , not alma imkanı sağlar aynı zamanda bozuk yada iptal olması gereken kodu kolaylıkla pasif hale getirebilir

C syntaxlı dillerde yorum satırı eklemek için // yazmamız yeterli olacaktır .



```
public class MerhabaJava{
    public static void main(String[] arguman){
        System.out.println("Ben bir palyaçoym"); // Diyor Manga
        //System.out.prina)= ; // Olamaz yanlış kod !
    }
}
```

Yalnız // sadece başına geldiği satırı pasif hale getirir . İstedğimiz alanı pasif hale getirmek için /\* - \*/ ikilisini kullanırız ..



```
public class MerhabaJava{
    public static void main(String[] arguman){
        /*
        System.out.println("Ben bir palyaçoyum"); // Diyor Manga
        //System.out.prina)= ; // Olamaz yanlış kod !
        */
        System.out.println("Bu görselleri carbon.now.sh ile yapıyorum .");
    }
}
```

## Kaçış Dizileri

Java'da ekranda bir takım verileri göstermek için `System.out.println()` ; methodunu kullanmıştık .

Şimdiye kadar bu methodu String veri türleri içinde kullandık . String veri türlerinde kaçış dizileri adını verdiğimiz \* bir takım özel karakterler vardır . Bu karakterler String ("tırnak içindeki her veri") içerisinde kullanılır . "`\`" ile çağırılır ...

- `\n`      Alt satıra geç
- `\t`      Bir tab boşluk bırak
- `\r`      Kendinden önceki bir satırı sil
- `\"`      " yazdır
- `\\`      \ yazdır

İlk kodumuzda , ilk programımızı " Merhaba Dünya " olarak tırnak içersinde yazıyorduk . Peki "Merhaba Dünya" içerisinde " işareti koymaya çalışırsak ne olur ?



```
public class MerhabaJava{  
    public static void main(String[] arguman){  
        System.out.print("Merhaba " Dünya");  
    }  
}
```

Java Söz dizimine uymadığı görselden de belli oluyor . Gine de derlemeye çalışalım

```
raif@raif:~/MerhabaJava$ javac MerhabaJava_kacis1.java  
MerhabaJava_kacis1.java:3: error: ')' expected  
    System.out.println("Merhaba " Dünya");  
                        ^  
MerhabaJava_kacis1.java:3: error: unclosed string literal  
    System.out.println("Merhaba " Dünya");  
                        ^  
MerhabaJava_kacis1.java:3: error: not a statement  
    System.out.println("Merhaba " Dünya");  
                        ^  
3 errors
```

İşte tam burada hata almamak için kaçış dizilerini kullanıyoruz .

"Merhaba " Dünya" olan kodu "Merhaba \" Dünya" olarak değiştirmemiz yeterli olacaktır



```
public class MerhabaJava{  
    public static void main(String[] arguman){  
        System.out.print("Merhaba \" Dünya");  
    }  
}
```

```
raif@raif:$ javac MerhabaJava_kacis1.java && java MerhabaJava_kacis1  
Merhaba " Dünya
```

(&& bağlacı ile tek komut ile derleyip çalıştırabiliriz)

(işlemi yap && ilk işlem yapıldı ise bana geç)

Peki \ eklemenden de tırnak işareti koyabilir miyim ?

" koymak yerine ' koyarsan neden olmasın :)

```
System.out.println("Merhaba ' Dünya");
```

Böylece tırnaklar birbirine karışmayacaktır .

Bir `print()` ile alt alta yazı yazmak istiyor olabiliriz ;



```
public class MerhabaJava{
    public static void main(String[] arguman){
        System.out.print("Merhaba \n Dünya");
    }
}
```



```
public class MerhabaJava{
    public static void main(String[] arguman){
        System.out.println("Merhaba Dünya\n");
        System.out.println("Merhaba Dünya\tTAB");
        System.out.println("Merhaba Dünya\rMerhaba Java");
        System.out.println("Merhaba Dünya \" Tırnak");
        System.out.println("Merhaba Dünya\\");
    }
}
```

```
raif@raif:$ javac MerhabaJava_kacis1.java && java MerhabaJava_kacis1
Merhaba Dünya
```

```
Merhaba Dünya      TAB
Merhaba Javaa
Merhaba Dünya " Tırnak
```



[hatırlatma] sınıf ismi ile (class) betik ismi aynı olmak zorunda !

Örneklerde betik sınıf MerhabaJava iken terminalden çalıştırdığım başka bir isime ait . Buna takılmayın ..

## Ayrıca

`System.out.println();` ile `System.out.print();` arasındaki tek fark `\n` olanın sonunda varsayıli olarak `\n` olmasıdır .



```
public class MerhabaJava{  
    public static void main(String[] arguman){  
        System.out.print("Merhaba Dünya\n");  
        System.out.println("Merhaba Dünya")  
    }  
}
```

- `System.out.print();` Değeri doğrudan ekrana yazar , alt satıra atlamaz
- `System.out.println();` Değeri doğrudan ekrana yazar , alt satıra atlar

```
raif@raif:$ javac MerhabaJava_kacis1.java && java MerhabaJava_kacis1  
Merhaba Dünya  
Merhaba Dünya
```

Birde `\n` kaçış dizisini kaldırıp `print();` ile `println();` arasındaki farkı anlayalım .



```
public class MerhabaJava{  
    public static void main(String[] arguman){  
        System.out.print("Merhaba Dünya");  
        System.out.println("Merhaba Dünya");  
    }  
}
```

```
raif@raif:$ javac MerhabaJava_kacis1.java && java MerhabaJava_kacis1
Merhaba DünyaMerhaba Dünya
```

`System.out.print();` alt satıra geçmediği için ilk Merhaba Dünya ile ikinci Merhaba Dünya birleşik yazıldı.

## Java'da Değişkenler

Her dilde olduğu gibi Java'da da değişkenler var . Daha önceden Python bilginiz var ise

```
degiskenadi = degiskendegeri
```

şeklinde değişken yaratıp içerisine değişken atayabiliyorduk .

Lakin Java'da işler biraz daha karışık . Tanımlamak istediğimiz değişkenin türünü , değişkeni atamadan önce belirtmek durumundayız . Böylece Java ram'de değişken için bir yer açacaktır .

## İlkel Değişkenler

Java'da sayı (1, 2, 3 ..) | char ('a' 'b' '\u0000' .. ) | boolean (true false) tutan her değişkene ilkel diyoruz . İlkel diyerek onları aşağılıyoruz ama ilkel değişkenler olmasaydı neredeyse hiçbir şey yapamazdık .

*Java'da değişken tanımlamak için*

```
degiskentur degiskenadi;
```

*Tanımlanan değişkene sonradan değer atamak için*

```
degiskenadi = degiskenin_alabilecegi_turden_deger ;
```

*Değişkeni oluşturduğumuz gibi değer atayabiliriz :*

```
degiskentur degiskenadi = degiskenin_alabilecegi_turden_deger;
```



```
public class ilkeldegiskenler{
    public static void main(String[] args) {        // Ilkel değişkenler
        boolean boolean_degisken = true;

        boolean boolean_degisken_2;
        boolean_degisken_2 = false;

        byte byte_degisken = 10;
        short short_degisken = 10;
        char char_degisken = 'a';
        int int_degisken = 10;
        long long_degisken = 10;
        float float_degisken = 10.0f;
        double double_degisken = 10.0;

        System.out.println(boolean_degisken);
        System.out.println(byte_degisken);
        System.out.println(short_degisken);
        System.out.println(char_degisken);
        System.out.println(int_degisken);
        System.out.println(long_degisken);
        System.out.println(float_degisken);
        System.out.println(double_degisken);

    }
}
```

Tekrardan hatırlatalım :

```
degiskenturu degiskenadi = degiskenin_alabilecegi_deger ;
```

## Değişken kuralları ;

- Değişkenler atanırken aralarında boşluluk **olmamalıdır** .

```
degiskentur degisken adi = degiskenin_alabilecegi_deger ;
```

- Değişken atanırken Türkçe karakter **kullanılmamalıdır** .

```
degiskentur değişkenadı = degiskenin_alabilecegi_deger ;
```

- Değişken atanırken herhangi bir değişken adı ile aynı isim **kullanılmamalıdır** .

degiskentur degiskentur = degiskenin\_alabilecegi\_deger ;

- Değişken atanırken herhangi bir sınıf adı ile aynı isim **kullanılmamalıdır** .

degiskentur sınıfınismi = degiskenin\_alabilecegi\_deger ;

- Değişken atanırken değişken adı rakam ile **başlamamalıdır** .

degiskentur 1degiskentur = degiskenin\_alabilecegi\_deger;



```
public class MerhabaJava{
    public static void main(String[] arguman){
        degiskenturu degiskenturu = degiskenin_alabilecegi_deger;
        degiskenturu 000 = degiskenin_alabilecegi_deger;
        degiskenturu MerhabaJava = degiskenin_alabilecegi_deger;
        degiskenturu degisken turu = degiskenin_alabilecegi_deger;
        degiskenturu deęiřkentürü = degiskenin_alabilecegi_deger;
    }
}
```

## Aralarındaki fark ne ?

İlke Değişkenler başlığındaki kod görselini iyice incelersek bazılarına harf bazılarına yazı\* bazılarına tam sayı bazılarına ise ondalık sayı girdiğimizi görebiliyoruz . Bu demek oluyor ki hepsi birbirinden özel . Peki neden tam sayıları tutan değişken türlerinden birden fazla var ? Neden bir tanesi yetmedi ki ? Java geliştirilmeye başlandığı zaman (1995) bilgisayar donanımları günümüzdeki kadar gelişmiş değildi .

Tam sayı türlerinin her birinin max ve min değerleri farklı . Buna bağlı olarak da her biri farklı ram tüketiyor .

Donanım yetersiz olduğu zamanlarda en düşük bellek kullanan yapıların kullanılması zorunluydu . Günümüzde böyle bir sorun kalmadığı için artık pek birşey ifade etmiyorlar . (Çoğu Java programcısı programlarında tam sayı vermek için int 'i kullanır ..)

Type	Size	Range	Default
boolean	1 bit	true or false	false
byte	8 bits	[-128, 127]	0
short	16 bits	[-32,768, 32,767]	0
char	16 bits	['\u0000', '\uffff'] or [0, 65535]	'\u0000'
int	32 bits	[-2,147,483,648 to 2,147,483,647]	0
long	64 bits	$[-2^{63}, 2^{63}-1]$	0
float	32 bits	32-bit IEEE 754 floating-point	0.0
double	64 bits	64-bit IEEE 754 floating-point	0.0

İnternette arakladığım bir görsel . Demek istediğimi çok güzel anlatıyor

## boolean Değişkeni

İçerisinde sadece `true` yada `false` saklanabilen 1 bit (8byte) yer kaplayan mantıksal değişken.

Bu değişkeni en çok mantıksal operatörlerde kullanacağız . Haydi tanımlayalım ;



```
public class boolean_degisken{
    public static void main(String[] arguman) {
        boolean bolbol;
    }
}
```

boolean\_degisken.java

Görseldeki hali ile `bolbol` isimli `boolean` türünde bir değişken yarattık

Ama bu değişkene herhangi bir değer atamadık . Yani ham bir boolean değişkenimiz var .

Haydi bu değeri olmayan değişkeni `System.out.println()` ; ile ekrana yazmaya çalışalım



```
public class boolean_degisken{
    public static void main(String[] arguman) {
        boolean bolbol;
        System.out.println(bolbol);
    }
}
```

javac ile derlemeye çalıştığımızda ise :

```
raif@raif:$ javac boolean_degisken.java
boolean_degisken.java:4: error: variable bolbol might not have been initialized
    System.out.println(bolbol);
                        ^
1 error
```

Şimdi ise bolbol değişkenine true değerini atayıp tekrar deneyelim



```
public class boolean_degisken{
    public static void main(String[] args) {
        boolean bolbol;
        bolbol = true;
        System.out.println(bolbol);
    }
}
```

İlk olarak boolean türünde bolbol değişkenimizi tanımladık , ardından değişkenimize true değerini verdik . Çünkü boolean değişkenler sadece true yada false değerlerini alabilirler ..

Derleyip çalıştırdığımızda ise

```
raif@raif:$ javac boolean_degisken.java && java boolean_degisken
true
```

Atadığımız true değerinin paşalar gibi buradayım dediğini görebiliyoruz .

Aynı şekilde `bolbol = false;` deseydik false değeri çıkacaktı .

Peki bu boolean ne işimize yarayacak ?

Değişkenleri öğrendikten sonra mantıksal operatörlere geçeceğiz . (`if else`)

Mantıksal operatörlerdeki her işlemler sonucun true yada false çıkması ile işliyor .

Orada bize bir hayli yararı olacaktır :)

## Tam Sayı Değişkenleri

Java'da tam sayıları (5 , 100, 909999 ..) değişkenlere atamak için kullandığımız değişken türleridir .

Tanımlanırken 10 , 500 , 1000000 olarak tanımlanabilirler . Değişken cinsinin alabileceği maksimum değerleri alabilirler . Ayrıca 1000 yerine 1\_000 | 1000000 yerine 1\_000\_000 olarak da ifade edilebilirler

## byte Değişken

byte değişkenler sayı tutan değişkenlerden bir tanesidir . Değer aralığı yukarıdaki tabloda da verildiği gibi

-128 ile 127 arasındadır . Bu aralığı +1 yada -1 aşarsa Java hata verecektir :



```
public class byte_degisken{
    public static void main(String[] arguman) {
        byte byte_degisken = 128;
        System.out.println(byte_degisken);
    }
}
```

```
raif@raif:$ java byte_degisken.java
```

```
byte_degisken.java:3: error: incompatible types: possible lossy conversion from
int to byte
```

```
    byte byte_degisken = 128;
```

^

1 error  
error: compilation failed

Bu durum sadece byte için değil bütün sayılar için geçerlidir . Bir sayı değişkenine alabileceğinden fazla sayı veremezsiniz ..

- PDF başında yazmıştım , OpenJDK11 'in bir özelliği olarak kodları derlemeden de java kod.java olarak açabiliyorum . Pek sağlıklı değil ama ..

```
public class byte_degisken{  
    public static void main(String[] arguman) {  
        byte byte_degisken = 127;  
        System.out.println(byte_degisken);  
    }  
}
```

byte 'a Byte tam sayı biçiminin kabul edebileceği bir değer verip verdiğimiz değeri  
System.out.println(); ile ekrana yazmaya çalışırsak

```
raif@raif:~/MerhabaJava$ java byte_degisken.java  
127
```

## int Değişken

Bir Integer değişken ile Short değişken arasında tek fark alabildikleri değerdir .

Type	Size	Range	Default
boolean	1 bit	true or false	false
byte	8 bits	[-128, 127]	0
short	16 bits	[-32,768, 32,767]	0
char	16 bits	['\u0000', '\uffff'] or [0, 65535]	'\u0000'
int	32 bits	[-2,147,483,648 to 2,147,483,647]	0
long	64 bits	$[-2^{63}, 2^{63}-1]$	0
float	32 bits	32-bit IEEE 754 floating-point	0.0
double	64 bits	64-bit IEEE 754 floating-point	0.0

Integer (int) değişkenler yaptığımız - yapacağımız programlarda işimize oldukça yarayacaktır



Mantık byte ile aynı olsa da (tüm tam sayı türleri ile) ginede bir örnek de onun için yazalım



```
public class int_degisken{
    public static void main(String[] args) {
        int int_degisken = 10;
        Integer int_degisken2 = 10;

        System.out.println(int_degisken);
        System.out.println(int_degisken2);
    }
}
```

Fark ettiyseniz short değişkenini atarken küçük harf ile `short short_deger;` şeklinde atamıştık

Burada ise `int` 'in altında `Integer` olarak da atanmış . Aralarında herhangi bir fark var mı ?

Hayır , aralarında herhangi bir fark yok ..

`short short_degisken = 10;` ile

`Short shport_degisken = 10 ;` arasında herhangi bir fark yok .

Peki tam sayı alması gereken `int`'a ondalık sayı girersek ne olur ?

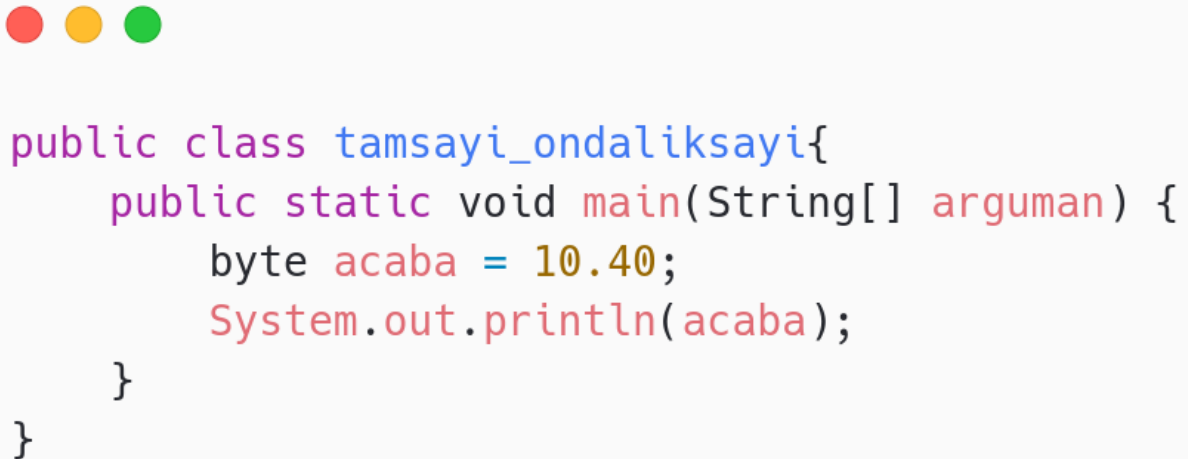


```
public class tamsayi_ondaliksayi{
    public static void main(String[] arguman) {
        int acaba = 10.40;
        System.out.println(acaba);
    }
}
```

Çalıştırmaya çalıştığımda ise :

```
raif@raif:~/MerhabaJava$ java tamsayi_ondaliksayi.java
tamsayi_ondaliksayi.java:3: error: incompatible types: possible lossy conversion
from double to int
    int acaba = 10.40;
                  ^
1 error
error: compilation failed
```

hm , belki de sorun Integer'da dır . Haydi Byte ile şansımızı bir daha deneyelim



```
public class tamsayi_ondaliksayi{
    public static void main(String[] arguman) {
        byte acaba = 10.40;
        System.out.println(acaba);
    }
}
```

```
raif@raif:~/MerhabaJava$ java tamsayi_ondaliksayi.java
tamsayi_ondaliksayi.java:3: error: incompatible types: possible lossy conversion
from double to int
    byte acaba = 10.40;
                  ^
1 error
error: compilation failed
```

Demek oluyor ki şimdilik ondalık sayıyı tam sayı yapamıyoruz . Değişkenler konusu bittikten sonra bunun aslında çok basit olduğunu anlayacaksınız . Şimdilik sadece devam edelim ..

## char Değişkeni

char değişkeninde işler biraz daha değişiyor . char değişkeni ne tam sayı taşımak için de ondalık sayı .

Adından da belli olduğu üzere sadece ama sadece karakter (sembol) taşımak için kullanılıyor .



```
public class char_degisken{
    public static void main(String[] args) {
        char char_degisken_1 = 'a';    // karakter
        char char_degisken_2 = '\u0061'; // Unicode | UTF-16

        System.out.println(char_degisken_1);
        System.out.println(char_degisken_2);
    }
}
```

```
raif@raif:~/MerhabaJava$ java char_degisken.java
```

```
a
```

```
a
```

Char diğer ilkel değişkenlere göre oldukça farklı . Sadece bir adet karakter alabiliyor .

Klavye karakteri alabildiği gibi Unicode karakterleri de alabiliyor . Unicode karakterlerini alabiliyor olması karakter aralığının çok geniş olduğu anlamına geliyor . Yani Unicode emojileri bile alabilir ..

Lakin dikkatinizi çekmek istediğim bir önemli kısım daha var . Char değişkenini atarken tek tırnak ( ' ) kullanıyoruz . Tek tırnak yerine çift tırnak ( " ) kullanırsak hata verecektir .



```
public class char_degisken{
    public static void main(String[] args) {
        char char_degisken_1 = 'a';    // karakter
        char char_degisken_2 = "\u0061";

        System.out.println(char_degisken_1);
        System.out.println(char_degisken_2);
    }
}
```

```
raif@raif:~/Masaüstü/MerhabaJava_betik$ java char_degisken.java
```

```
char_degisken.java:4: error: incompatible types: String cannot be converted to
```

**char**

```
char char_degisken_2 = "\u0061";  
                        ^
```

1 error

error: compilation failed

## double Değişken

Tam sayılar ve char'lerin üzerinden hızlıca geçtiğimize göre sıra double değişkenlerde .

Double ve Float değişkenlerde ondalık sayı ekleyebiliyoruz . Aynı şekilde tam sayı da ekleyebiliriz ama eklediğimiz tam sayının sonuna .0 ekleyerek ondalık sayı yapacaktır :