Java: Sayıları Biçemli Yazma

Sayıların çıktıya istenen biçemde gönderilmesi için, çıktının istenen *string (text)* biçemine dönüştürülmesi gerekir. Java bu işi değişik yöntemlerle yapabilir.

## Java’da sayıyı string’e dönüştürme yolları

*Concatenation (+):* string birleştirme operatörü (+), işleme giren öğelerden (operand) en az birisi *string* olduğunda birleşime girenlerin hepsini *string* varsayar ve çıktıya *string* olarak gönderir. (örneğin, "Kişinin Yaşı = " + 21).

*java.text.DecimalFormat*  sınıfı sayıları binlik hanelere ayırmak, kesir kısmının hane sayısını belirlemek, bilimsel notasyonla yazmak, para birimi koymak gibi yerelleştirme işlemlerini yapan çok sayıda metoda sahiptir.

Sayıyı ilgili sınıfa gömen sınıf metotlarını kullanmak, örneğin,

Integer.toString(*i*).

printf()*metodu*. Java 5 ile gelen bu yöntem sayıyı istenen biçeme dönüştürmek için kolay yöntemler sunmaktadır.

*Dönüşüme gerek duyulmazsa: Bazı*  *sistem* metodları her hangi bir veri tipini *text* olarak çıktıya gönderir. Örneğin, System.out.println() metodu hiç bir dönüşüme gerek olmadığı zaman sayıyı olduğu gibi çıktıya gönderir. .

Şimdi bu yöntemleri örneklerle ele alacağız.

## Concatenation (+)

Sayıyı string’e dönüştürmenin kolay yöntemlerinden birisidir. Dönüştürülmek istenen sayı bir string ile (+) operatörü yardımıyla birleştirilir. Uygun bir string olmadığında ("") boş stringi kullanılabilir.

Bilindiği gibi (+) string birleştirme (concatanation) operatörü, işleme girenlerden birisi *String* ise, ötekilerin hepsini Stringe dönüştürür.

**package** sayıBiçemleme;

**public** **class** Concat01 {

**public** **static** **void** main(String[] args) {

**int** x = 73;

String s, s0, s1, s2, s3, s4;

// s0 = x; // HATA, String tipine sayı atanamaz

s1 = "" + x; // 73 sayısını "73" String tipine dönüştürüyor

System.*out*.println(s1);

s2 = x + " yıl"; // s2 değişkenine "73 yıl" stringini atıyor

System.*out*.println(s2);

s3 = "" + 7.6; // s3 değişkenine "7.6" stringini atıyor

System.*out*.println(s3);

s4 = "" + 1.0 / 7.0; // s4 değişkenine "0.14285714285714285" stringini atıyor

System.*out*.println(s4);

}

/\*

Çıktı:

73 yıl

7.6

0.14285714285714285

\*/

Duyarlılık (Precision): Concatenation (+) operatörü, kesirli sayıların, kesir ayracından (onlu çekesi, decimal point) sonra kaç haneli yazılacağına kendisi karar verir; programcının isteğine bırakmaz. Öntanımlı (default) olarak, kesir ayracından sonra, kesirde varsa 17 hane yazar. Tabii, sayının ondalık kısmının hane sayısı 17 den az ise, çıktıya mevcut haneler gider.

## java.text.DecimalFormat

java.text.DecimalFormat

sınıfının API yapısı şöyledir:

## java.text Class DecimalFormat

[java.lang.Object](http://download.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html)

extended by[java.text.Format](http://download.oracle.com/javase/1.4.2/docs/api/java/text/Format.html)

extended by[java.text.NumberFormat](http://download.oracle.com/javase/1.4.2/docs/api/java/text/NumberFormat.html)

extended byjava.text.DecimalFormat

Bu sınıf, sayıları biçemli yazmak için çok sayıda metoda sahiptir. Bunlaın tam listesi

<http://download.oracle.com/javase/1.4.2/docs/api/java/text/DecimalFormat.html>

web sitesinden görülebilir. Burada önemli bazı metotları örneklerle ele alacağız.

Java 1.1 sürümüne eklenen *java.text* paketinin üç alt sınıfı vardır:

  java.text.Format  
  java.text.NumberFormat  
  java.text.DecimalFormat

#### Uyarı:

Farklı kültürler, sayıda kesir ayracını, binlikler ayracını, para birimini vb farklı yazarlar. *java.text* paketinin asıl amacı, sayıların farklı kültürlere göre yazılışını yapmaktır. Buna *“internationalization”* ya da *locale* deniliyor. Ayrıca, sayının tam kısmının ve kesirli kısmının istenen biçemde bir text’e dönüşmesini, bilimsel notasyonla yazılmasını vb sağlar.

#### Farklı kültürlerde (.) ve (,) simgeleri farklı anlamlar taşıyabilir. Bütün kültürlerde, sayısal girdiler için (.) ondalık kesir ayracıdır. Çünkü girdi, derleyici bağımlıdır ve derleyiciler Amerikan standardını kullanır. Buna göre (.) ondalık ayracıdır. Çıktıya gelince durum değişebilir. Çıktı bir text olarak alındığı için, farklı kültürlere uyan nakışlar elde edilebilir. Örneğin, Türkçe’de (,) ondalık ayracı ve (.) binlikler ayracıdır. Amerikan standardında ise (.) ondalık ayracı ve (,) binlikler ayracıdır.

Yukarıda, *Java 5* sürümünün, kesirli sayıları istenen biçime dönüştürmek için *printf()* metodunu getirdiğini söylemiştik. Onu ayrı bir alt konu olarak ele alacağız.

Özel Karekterler

Java sayıları biçemlemek için aşağıdaki semboller kullanır.

|  |  |  |  |
| --- | --- | --- | --- |
| Sembol | Yeri | Yerel mi? | Anlamı |
| 0 | Sayı | Evet | Basamak (hane, digit); sayıda yeterli basamak yoksa, yerine 0 yazılır |
| # | Sayı | Evet | Basamak (hane, digit); sayıda yeterli basamak yoksa, yeri boş kalır |
| . | Sayı | Evet | Kesir ayracı |
| - | Sayı | Evet | Eksi işareti |
| , | Sayı | Evet | Binliklere gruplama sembolü |
| E,e | Sayı | Evet | Bilimsel gösterimde, sayının 10 üstü ile çarpımındaki üstü belirler |
| ; | Altbiçem sınır | Evet | Pozitif ya da negatif sayı gösterme biçemini belirler |
| % | Önel ya da sonal takı | Evet | 100 ile çarparak yüzde oranını belirler |
| \u2030 | Önel ya da sonal takı | Evet | 1000 ile çarparak binde oranını belirler |
| ¤ (\u00A4) | Önel ya da sonal takı | Hayır | Para (Currency) işareti |
| ' | Prefix or suffix | Hayır | Sayının önüne ya da sonuna getirilecek literal karekterleri belirler |

## java.text.DecimalFormat Sınıfı

*java.text.DecimalFormat* sınıfındaki *format()* metodu ile çıktıyı biçemleyebiliriz. Double tiplerinde öntanımlı (default) ondalık hane sayısı, onlu çekesinden sonra 17 hanedir Ayrıca *DecimalFormat(nakış)* kurucusunda *nakış* parametresi yerine istenen biçem yazılabilir. Bunun nasıl olduğunu aşağıdaki örneklerde göreceğiz.

#### Çıktıyı nakışlama (pattern)

Çıktıyı “0” ile nakışlama (pattern)

Sayının çıktısının alacağı biçem “0” simgeleri ile nakışlanarak (pattern kurularak) belirlenebilir. Örneğin, “000.000,00” nakışı sayının tam kısmının altı haneli olmasını ve binliklere ayrılmasını; sayının ondalık kısmının ise iki haneli olmasını belirtir. Sayının tam kısmında nakıştaki kadar hane varsa onlar yerlerini alır. Sayının tam ve kesir kısmında, nakıştakinden daha az hane varsa, onların yerine “0” konuşlanır. Sayının tam kısmında nakıştakinden daha çok hane varsa, derleyici nakışı dikkate almaz, sayının tam kısmının bütün hanelerini eksiksiz yazar. Sayının kesirli kısmında nakıştakinden daha çok hane varsa, sayının kesir kısmı alınan hane sayısına yuvarlanır. Yuvarlama işleminde, atılan kesir kısmının en soldaki basamağı 0-4 arasında ise hepsi atılır. En soldaki basamak 5-9 arasında ise, kalan kısmın en sağdaki basamağı 1 artırılır. Bu işlem, kesirler atılırken sayının en yakın basamağa yuvarlanması işlemidir. Ayrıca nakış (pattern) içine kesir ayracı ve binlikler ayraçları konulabilir. Aşağıdaki tablo “0” ile nakışlamanın etkilerini göstermektedir.

|  |  |  |
| --- | --- | --- |
| Sayı | Nakış (pattern) | Çıktı |
| 12345 | “000” | 12345 |
| 1234.6 | “000000” | 012345 |
| 1234.56789 | “000.000,00” | 001.234,57 |
| 1234.56 | “000.000,000” | 001.234,560 |

Biçem01.java

import java.text.DecimalFormat;

import java.text.NumberFormat;

public class Biçem01 {

    public static void main(String[] args) {

    NumberFormat nakış = new DecimalFormat("0000000");

    String number = nakış.format(2500);

    System.out.println("Sayının önüne 0 lar konulur: " + number);

    }

}

/\*

Çıktı:

Sayının önüne 0 lar konulur: 0002500

\*/

Biçem02.java

**package** sayıBiçemleme;

/\*

\* DecimalFormat() metodu

\*/

**import** java.text.DecimalFormat;

**public** **class** Biçem02 {

**public** **static** **void** main(String[] args) {

DecimalFormat nakış = **new** DecimalFormat("0.000");

**for** (**int** i = 1; i <= 10; i++) {

**double** sayı = 1.0 / i;

System.*out*.println("1/" + i + " = " + nakış.format(sayı) + ", " + sayı);

}

}

}

/\*

Çıktı:

1/1 = 1,000, 1.0

1/2 = 0,500, 0.5

1/3 = 0,333, 0.3333333333333333

1/4 = 0,250, 0.25

1/5 = 0,200, 0.2

1/6 = 0,167, 0.16666666666666666

1/7 = 0,143, 0.14285714285714285

1/8 = 0,125, 0.125

1/9 = 0,111, 0.1111111111111111

1/10 = 0,100, 0.1

\*/

Biçem03.java

/\*

\* Bu program kesirli sayıları tamsayıya yuvarlar

\* 0 simgesi sayının bir hanesi yerine geçer.

\* Yeterince hane yoksa, 0 ların konumu “0” ile doldurulur

\*/

**package** sayıBiçemleme;

**import** java.text.NumberFormat;

**import** java.text.DecimalFormat;

**public** **class** Biçem03 {

**public** **static** **void** main(String[] args) {

String s;

NumberFormat nf1 = **new** DecimalFormat("000000");

s = nf1.format(-1234.567); // -001235

System.*out*.println(s);

NumberFormat nf2 = **new** DecimalFormat("000000.00");

s = nf2.format(12.34567); // 000012,35

System.*out*.println(s);

}

}

/\*

-001235

000012,35

\*/

Çıktıyı “#” ile nakışlama (pattern)

Sayının çıktısının alacağı biçem, “0” yerine “#” simgeleri ile de nakışlanabilir. “#” simgesi sayının bir hanesi yerine geçer. “0” ile yapılan nakışı aynen yapar. Ondan tek farkı, sayının haneleri yetmediği zaman, nakışta “#” konumlarının boş kalmasıdır. Örneğin, “###.###,##” nakışına 1234.7 sayısı gönderilirse, çıktı 1.234,7 olur. Kesir kısmının yuvarlanması aynen “0” nakışında olduğu gibidir. İstenirse “0” simgeleri ile “#” simgelerinden oluşan anlamlı karma nakışlar (pattern) düzenlenebilir.

Aşağıdaki tablo “0” ile nakışlamanın etkilerini göstermektedir.

|  |  |  |
| --- | --- | --- |
| Sayı | Nakış (pattern) | Çıktı |
| 12345 | “###” | 12345 |
| 1234.6 | “######” | 12345 |
| 1234.56789 | “###.###,##” | 1.234,57 |
| 1234.56 | “###.###,###” | 1.234,560 |

Aşağıdaki programdaki nakışlar tamsayı istemektedir. Dolayısıyla, kesirler en yakın tamsayıya yuvarlanarak çıktıya gönderilir.

Biçem04.java

/\*

\* Bu program sayıları tamsayıya yuvarlar

\* simgesi sayının bir hanesi yerine geçer.

\* Sayıda yeterli hane yoksa # konumu boş kalır

\*/

**package** sayıBiçemleme;

**import** java.text.DecimalFormat;

**public** **class** Biçem04 {

**public** **static** **void** main(String[] args) {

String s;

DecimalFormat nakış = **new** DecimalFormat("##");

s = nakış.format(-1234.567); // -1235

System.*out*.println(s);

s = nakış.format(0); // 0

System.*out*.println(s);

s = nakış.format(12345); // 12345

System.*out*.println(s);

nakış = **new** DecimalFormat("##00");

s = nakış.format(0); // 00

System.*out*.println(s);

}

}

/\*

-1235

0

12345

00

\*/

##### Biçem05.java

/\*

\* \* Sayıyı binliklerine ayırma

\*/

**package** sayıBiçemleme;

**import** java.text.DecimalFormat;

**public** **class** Biçem05 {

**public** **static** **void** main(String[] args) {

String s;

DecimalFormat nakış = **new** DecimalFormat("#,###,###");

s = nakış.format(-1234.567); // -1,235

System.*out*.println(s);

s = nakış.format(-1234567.890); // -1,234,568

System.*out*.println(s);

}

}

/\*

-1.235

-1.234.568

\*/

##### Biçem06.java

/\*

\* \* (;) simgesi negatif sayıların nasıl gösterileceğini belirler

\*/

**package** sayıBiçemleme;

**import** java.text.DecimalFormat;

**public** **class** Biçem06 {

**public** **static** **void** main(String[] args) {

String s;

DecimalFormat nakış = **new** DecimalFormat("#;(#)");

s = nakış.format(-1234.567); // (1235)

System.*out*.println(s);

}

}

/\*

(1235)

\*/

##### Biçem07.java

/\*

\* \* (') simgesi literal sembolleri işaretlemek için kullanılır

\*/

**package** sayıBiçemleme;

**import** java.text.DecimalFormat;

**public** **class** Biçem07 {

**public** **static** **void** main(String[] args) {

String s;

DecimalFormat nakış = **new** DecimalFormat("'#'#");

s = nakış.format(-1234.567); // -#1235

System.*out*.println(s);

nakış = **new** DecimalFormat("'abc'#");

s = nakış.format(-1234.567); // -abc1235

System.*out*.println(s);

nakış = **new** DecimalFormat("'TL '#");

s = nakış.format(1234.567); // TL1235

System.*out*.println(s);

nakış = **new** DecimalFormat("#' YTL'");

s = nakış.format(1234.567); // 1235TL

System.*out*.println(s);

}

}

/\*

-#1235

-abc1235

YTL 1235

1235 YTL

\*/

## java.text.NumberFormat

Bu sınıfın API yapısı şöyledir:

## java.text Class NumberFormat

[java.lang.Object](http://download.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html)

extended by[java.text.Format](http://download.oracle.com/javase/1.4.2/docs/api/java/text/Format.html)

extended byjava.text.NumberFormat

public abstract class NumberFormat

extends [Format](http://download.oracle.com/javase/6/docs/api/java/text/Format.html)

NumberFormat bütün sayısal biçemlerin soyut (abstract) sınıfıdır. Sayıları biçemlemek için gerekli arayüze ve metotlara sahiptir. Ayrıca NumberFormat sınıfı sayıları her kültüre (yerelleşme, locale) göre yazabilir ve o biçemlerden sayıya dönüşüm (*parsing*) yapabilir.

İşletim sisteminin yereline (*locale*) göre bir x sayısını bir stringe dönüştürmek için

birString = NumberFormat.getInstance().format(x);

deyimi kullanılır. Örneğin, aşağıdaki program, sayıları Türkçe standardında yazmaktadır.

###### FormatA.java

/\*

\* Bu program kesirli sayıları tamsayıya yuvarlar

\*/

**package** sayıBiçemleme;

**import** java.text.NumberFormat;

**import** java.text.DecimalFormat;

**public** **class** FormatA {

**public** **static** **void** main(String[] args) {

NumberFormat nf = **new** DecimalFormat("000000");

String s = nf.format(-1234.567); // -001235

System.*out*.println(s);

}

}

/\*

-001235

\*/

##### FormatB.java

/\*

\* Bu program kesirli sayıları Türkçe standardına göre yazar

\*/

package sayıBiçemleme;

import java.text.NumberFormat;

import java.text.DecimalFormat;

public class FormatB {

public static void main(String[] args) {

// (.) simgesi girdi için kesir ayracıdır

String s;

NumberFormat nakış = new DecimalFormat(".00");

s = nakış.format(-.567);

System.out.println(s); // -,57

nakış = new DecimalFormat("0.00");

s = nakış.format(-.567);

System.out.println(s); // -0,57

nakış = new DecimalFormat("#.#");

s = nakış.format(-1234.567);

System.out.println(s); // -1234,6

nakış = new DecimalFormat("#.######");

s = nakış.format(-1234.567); // -1234,567

System.out.println(s);

}

}

/\*

-,57

-0,57

-1234,6

-1234,567

\*/

Farklı bir kültüre göre yazmak için

getInstance()

metodu çağrılır. Örneğin, yukarıdaki sayıları Fransız kültürüne göre yazdırmak için

NumberFormat nf = NumberFormat.getInstance(Locale.FRENCH);

deyimi kullanılır.

Stringe dönüşmüş bir sayıyı tekrar sayı tipine dönüştürmek için *NumberFormat* sınıfı içindeki *parse()* metodu kullanılır:

bSayı = nf.parse(bString);

Bu sınıfın içinde

*getInstance*

*getNumberInstance*

*getIntegerInstance*

*getCurrencyInstance*

*getPercentInstance*

metotlarının, adlarından anlaşılan farklı işlevleri vardır. Bu işlevlerin neler olduğunu görmek için yukarıdakine benzer programlar yazınız.

#### *getInstance() metodu*

*NumberFormat* sınıfı içindeki *getInstance()* metodu çıktıyı farklı ülke ve kültürlere göre ayarlamayı sağlar. Parametresiz kullanıldığında öntanımlı ülke ve kültüre göre çıktı alınır. Parametre kullanılarak istenilen ülke ve kültüre uyan çıktı elde edilebilir.

Locale01.java

/\*

\* Bu program kesirli sayıları Türkçe standardına göre yazar

\*/

**package** sayıBiçemleme;

**import** java.text.NumberFormat;

**public** **class** Locale01 {

**public** **static** **void** main(String[] args) {

**double**[] dizi = {8, 12345678, 11.3,25.67,36.12345,40.0,58.987654};

NumberFormat nf = NumberFormat.*getInstance*();

**for** (**int** i = 0; i < dizi.length; ++i) {

System.*out*.println(nf.format(dizi[i]) );

}

}

}

/\*

8

12.345.678

11,3

25,67

36,123

40

58,988

\*/

Locale02.java

/\*

\* Bu program kesirli sayıları farklı kültürlere göre yazar

\*/

package sayıBiçemleme;

import java.text.NumberFormat;

import java.util.Locale;

public class Locale02 {

public static void main(String args[]) throws Exception {

NumberFormat numberFormat = NumberFormat.getInstance();

numberFormat.setParseIntegerOnly(false);

double dSayı = 1234.0567;

numberFormat = NumberFormat.getNumberInstance(Locale.US);

System.out.println("ABD Gösterimi (US) : "

+ numberFormat.format(dSayı));

numberFormat = NumberFormat.getNumberInstance(Locale.FRANCE);

System.out.println("Fransız gösterimi (FRANCE): "

+ numberFormat.format(dSayı));

numberFormat = NumberFormat.getNumberInstance(Locale.GERMAN);

System.out.println("Alman gösterimi (GERMAN) : "

+ numberFormat.format(dSayı));

NumberFormat nakış = NumberFormat.getNumberInstance(Locale.ITALY);

try {

String number = nakış.format(1234.0567);

System.out.println("İtalyan gösterimi (ITALY) : " + number);

} catch (NumberFormatException e) {

e.printStackTrace();

}

nakış = NumberFormat.getNumberInstance(Locale.JAPAN);

try {

String number = nakış.format(1234.0567);

System.out.println("Japon gösterimi (JAPAN) : " + number);

} catch (NumberFormatException e) {

e.printStackTrace();

}

}

}

/\*

ABD Gösterimi (US) : 1,234.057

Fransız gösterimi (FRANCE): 1 234,057

Alman gösterimi (GERMAN) : 1.234,057

İtalyan gösterimi (ITALY) : 1.234,057

Japon gösterimi (JAPAN) : 1,234.057

\*/

## StringBuffer ve StringBuilder append() metodu

StringBuilder sınıfının append metodu (+) birleştirme operatörü gibi işlev görür.

**package** sayıBiçemleme;

**public** **class** StringBuilder01 {

**public** StringBuilder01() {

String s;

}

**public** **static** **void** main(String[] args) {

StringBuilder sb = **new** StringBuilder();

**int** i = 42;

sb.append("Çıktı : ");

sb.append(i); // i sayısını stringe dönüştürür ve sb ye ekler

sb.append(", ");

sb.append(1.0/3.0);

System.*out*.println(sb);

System.*out*.println(sb.toString());

String s = sb.toString();

System.*out*.println(s);

}

}

/\*

\* Çıktı : 42, 0.3333333333333333

\* Çıktı : 42, 0.3333333333333333

\* Çıktı : 42, 0.3333333333333333

\*/

Örnekler

NumberFormat:

setParseIntegerOnly(boolean value)

/\*

\* Bu program kesirli sayıları farklı kültürlere göre yazar

\*/

**package** sayıBiçemleme;

**import** java.text.NumberFormat;

//import java.text.DecimalFormat;

**import** java.util.Locale;

**public** **class** Local01 {

**public** **static** **void** main(String args[]) **throws** Exception {

NumberFormat numberFormat = NumberFormat.*getInstance*();

numberFormat.setParseIntegerOnly(**false**);

**double** dSayı = 9876.0123;

numberFormat = NumberFormat.*getNumberInstance*(Locale.*US*);

System.*out*.println("Yerel biçim (US) : "

+ numberFormat.format(dSayı));

numberFormat = NumberFormat.*getNumberInstance*(Locale.*FRANCE*);

System.*out*.println("Yerel biçim (FRANCE): "

+ numberFormat.format(dSayı));

numberFormat = NumberFormat.*getNumberInstance*(Locale.*GERMAN*);

System.*out*.println("Yerel biçim (GERMAN): "

+ numberFormat.format(dSayı));

}

}

/\*

Yerel biçim (US) : 9,876.012

Yerel biçim (FRANCE) : 9 876,012

Yerel biçim (GERMAN) : 9.876,012

\*/

NumberFormat: setMinimum… ve setMaximum… Metotları

Aşağıdaki metotlar sayının biçeminde hane sayıları için minimum ve maksimum değerleri koyar.

setMinimumIntegerDigits(int hane);

setMinimumFractionDigits(int hane);

setMaximumFractionDigits(int hane);

Bunun nasıl olduğunu aşağıdaki örnekten görebiliriz.

import java.text.NumberFormat;  
  
public class FormatC {  
  
  public static void main(String[] av) {  
    double data[] = { 0, 3, 11d / 3, 2000.9876543 };  
    NumberFormat form = NumberFormat.getInstance();  
  
    //  999.99[99] biçiminde yazar  
    form.setMinimumIntegerDigits(3);  
    form.setMinimumFractionDigits(2);  
    form.setMaximumFractionDigits(4);  
  
    // Now print using it.  
    for (int i = 0; i < data.length; i++)  
    System.out.println(data[i] +

"\tsayısının Çıktısı: " +

form.format(data[i]));  
  }  
}

/\*  
0.0 sayısının Çıktısı: 000,00

3.0 sayısının Çıktısı: 003,00

3.6666666666666665 sayısının Çıktısı: 003,6667

2000.9876543 sayısının Çıktısı: 2.000,9877

\*/

NumberFormat:

parse(String kaynak) Metodu

import java.text.NumberFormat;  
import java.text.ParseException;  
  
public class FormatD {  
  
  public static void main(String[] av) {  
  
    String input = "76054.984";  
    NumberFormat nf = NumberFormat.getInstance();  
  
    try {  
      Number d = nf.parse(input);  
      System.out.println(input +

" stringini " +

d +

" olarak okur ve \n " +

nf.format(d) +

“biçiminde yazar);  
    } catch (ParseException pe) {  
      System.err.println(input + "okunamıyor!");  
    }  
  }  
}

/\*

76054.984 stringini 76054984 olarak okur ve

76.054.984biçiminde yazar

\*/

NumberFormat:

getNumberInstance(Locale yerel)

Çıktıyı istenen ülke ya da kültüre göre yazar.

**import**java.text.NumberFormat;  
**import**java.util.Locale;  
  
**public class**FormatE {  
  **public static void**main(String args[]) **throws**Exception {  
    NumberFormat numberFormat = NumberFormat.getInstance();  
    numberFormat.setParseIntegerOnly(**false**);  
    **double**dSayı = 12345.07938;  
  
    numberFormat = NumberFormat.getNumberInstance(Locale.US);  
    System.out.println("Yerel (US): " + numberFormat.format(dSayı));  
      
  }  
}

/\*   
Yerel (US) : 12,345.079

Yerel (FRANCE): 12 345,079  
\*/

NumberFormat:

getInstance() Metoduna örnek

**import** java.awt.FlowLayout;

**import** java.awt.Font;

**import** java.text.Format;

**import** java.text.NumberFormat;

**import** java.util.Locale;

**import** javax.swing.BoxLayout;

**import** javax.swing.JFormattedTextField;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JPanel;

**public** **class** FormatF {

**public** **static** **void** main(String args[]) **throws** Exception {

JFrame frame = **new** JFrame("Sayı Girişi");

frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

Font font = **new** Font("SansSerif", Font.*BOLD*, 16);

JLabel label;

JFormattedTextField input;

JPanel panel;

BoxLayout layout = **new** BoxLayout(frame.getContentPane(),

BoxLayout.*Y\_AXIS*);

frame.setLayout(layout);

Format general = NumberFormat.*getInstance*();

label = **new** JLabel("Genel/Anlık:");

input = **new** JFormattedTextField(general);

input.setValue(79546.639);

input.setColumns(20);

input.setFont(font);

panel = **new** JPanel(**new** FlowLayout(FlowLayout.*RIGHT*));

panel.add(label);

panel.add(input);

frame.add(panel);

frame.pack();

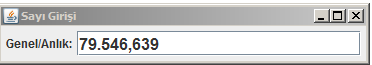
frame.setVisible(**true**);

}

}

/\*

Çıktı:



\*/

NumberFormat:

getCurrencyInstance(Locale.UK)

**import**java.awt.FlowLayout;  
**import**java.awt.Font;  
**import**java.text.Format;  
**import**java.text.NumberFormat;  
**import**java.util.Locale;  
  
**import**javax.swing.BoxLayout;  
**import**javax.swing.JFormattedTextField;  
**import**javax.swing.JFrame;  
**import**javax.swing.JLabel;  
**import**javax.swing.JPanel;  
  
**public class**MainClass {  
  **public static void**main(String args[]) **throws**Exception {  
  
    JFrame frame = **new**JFrame("İngiltere Para Birimi");  
    frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
    Font font = **new**Font("SansSerif", Font.BOLD, 16);  
  
    JLabel label;  
    JFormattedTextField input;  
    JPanel panel;  
  
    BoxLayout layout = **new**BoxLayout(frame.getContentPane(), BoxLayout.Y\_AXIS);  
    frame.setLayout(layout);  
  
    Format currency = NumberFormat.getCurrencyInstance(Locale.UK);  
    label = **new**JLabel("UK Currency:");  
    input = **new**JFormattedTextField(currency);  
    input.setValue(2424.50);  
    input.setColumns(20);  
    input.setFont(font);  
    panel = **new**JPanel(**new**FlowLayout(FlowLayout.RIGHT));  
    panel.add(label);  
    panel.add(input);  
    frame.add(panel);  
  
    frame.pack();  
    frame.setVisible(**true**);  
  }  
}

|  |  |
| --- | --- |
| |  | | --- | |  | |