System sınıfı

java.lang.Object

java.lang.System

public final class System

extends Object

System sınıfı, java.lang paketi içindedir. Platformdan bağımsız olarak sistem düzeyindeki eylemleri belirleyen dingin (static) bir sınıftır.

Bildiğiniz gibi, bütün java programları java.lang paketinin dışalımını (import) otomatik olarak yapar, onu ayrıca import etme gereği yoktur. Dolayısıyla, her java programı, System sınıfını içerir.

System sınıfı public damgalı olduğundan, program içindeki her kod ona erişebilir. Ama, ayrıca final damgalı olduğundan, kalıtım olamaz; yani onun alt sınıfları yaratılamaz. Bu nedenle onun metotları da, otomatik olarak, final damgalı olur, yani değiştirilemezler.

System'in üç tane sınıf değişkeni vardır: err, in, out. Bunlardan ilki, sistemde oluşan hataları bildirir, ikincisi giriş akımlarını, üçüncüsü çıkış akımlarını yapar. Değişkenlerin tanımları şöyledir.

```
public static final PrintStream err;        // hataları işaret eden referans
public static final InputStream in;         // giren veriyi işaret eden referans
public static final PrintStream out;        // çıkan veriyi işaret eden referans
```

Değişkenlerin nitemlerine bakarsak, şunları görürüz:

Hepsi public damgalı olduğu için, programdaki her kod err, in, out değişkenlerine erişebilir.

Hepsi static damgalı olduğundan, onlara bellekte ancak birer tane yer ayrılır, nesneler içinde kopyaları olamaz. Her birisinin birer değeri vardır. Program içindeki kodlar, o tek değerlere erişir.

Her üç değişken final damgalıdır. Dolayısıyla, program boyunca değerleri (işaret ettikleri adresler) değiştirilemez.

Buna göre, `System` sınıfının `InputStream` tipinden olan `System.in` değişkeni, `InputStream` sınıfına ait bir nesneyi işaret eden bir referans değişkenidir (işaretçi, pointer). Aynı şekilde, **`PrintStream`** tipinden olan `System.err` ve `System.out` değişkenleri **`PrintStream`** sınıfına ait nesneleri işaret eden referans değişkenleridir. Her iki sınıf byte akımı yapan sınıflardandır. Böyle olmakla birlikte, konsoldan karekter okuma/yazma eyleminde de kullanılabilirler. Gerçekten, `System.in` standart girişten (klâvye) veri alır. `System.err` byte akımında oluşabilecek hataları standart çıkışa (konsole) yollar. `System.out` standart çıkışa (monitör) veri yollar.

`System` sınıfının, sistem ile ilgili işleri yapan 30 dan çok metodu vardır. Örneğin, bu metotlar çevre değişkenlerine erişim sağlar, dosyaları ve kütüphaneleri yükler, array kopyalar, zaman bildirir, JVM'i sonlandırır, çöp toplayıcıyı çalıştırır, v.b.

Bunun yanında, `System` sınıfının bir düzine kadar sınıf fonksiyonu (class method) vardır. Onun değişkenleri ve metotları, bir nesne (object) yaratmaya gerek kalmadan doğrudan kullanılabilir; çünkü `System` sınıfı dingindir (static). Static bir sınıfa ana bellekte bir yer ayrılır, ayrıca onu temsil edecek nesne yaratmaya gerek kalmaz. Böylece, standart giriş/çıkış işlemleri için `java.lang.System` sınıfı kullanılabilir hale gelir. Daha önceki bölümlerde `System.out` sınıfını bir çok kez kullandık. `java.lang` paketi her java programında kendiliğinden yer aldığından, `java.lang.System` yerine `System` yazmak yeterlidir.

System.out

`System.out` değişkeni, standart çıkış akımıdır. PC'lerde standart çıkış birimi monitördür. `out` değişkeni System sınıfının static bir değişkeni olduğundan, derleyici ona ana bellekte bir yer ayırır; ayrıca nesne yaratmaya gerek yoktur. `out` değişkeni `PrintStream` tipindendir; dolayısıyla PrintStream sınıfının metotlarını kullanabilir. Şimdiye dek yazdığımız programlarda `print()`, `println()` ve `printf()` metotlarını sıkça kullandık. Başka bir deyişle `System.out.print()`, `System.out.println()` ve `System.out.printf()` metotlarını monitöre text yollamak için birer araç olarak kullandık. `Java.lang` paketi içinde olan ve giriş/çıkış işlemlerinde `byte akımını` sağlayan bu `System` araçları, doğrudan doğruya işletim sistemi ile ilişkili olduğundan önemlidirler. Ancak, java gelişkin veri akımı işlemleri için `java.io` paketini yaratmıştır.

System.in

`System` sınıfı standart giriş akışını (genellikle klâvye) temsil eden `in` adlı değişkeni içerir. Bu değişkenin tipi `InputStream` sınıfıdır. Dolayısıyla onun metotlarını kullanabilir. `System.in` sınıfı standart girişten veri akışını sağlar. Örneğin, `InputStream` sınıfının `read()` metodunu çağırdığında `System.in.read()` metodu çalışır. Bu deyimi sık sık kullanacağımız için, bunun nasıl olduğunu açıklamak yararlı olacaktır.

`in` değişkeni `System` sınıfının *static* bir öğesidir ve `InputStream` tipindendir. Dolayısıyla onu `System.in` diye çağırırız. Öte yandan, `InputStream` sınıfının `read()` metodu vardır. Dolayısıyla, `System.in.read()` deyimi söz konusu metottan başkası değildir. Benzer şeyin `System.out.print()` deyimi için de var olduğunu daha önce söylemiştik.

System.err

`System` sınıfı standart giriş/çıkış akışında oluşabilecek hataları tutan `err` değişkenini içerir. Bu değişkenin tipi `InputStream` sınıfıdır. `System.err` değişkeni giriş/çıkış akımında oluşabilecek hataları bildirir.

Örnekler:

Bu program, klâvyeden girilen karekterin `ascii` kodunu okur; yani `System.in.read()` metodunun değerleri 0 ile 255 sayıları arasındaki ascii kodlarıdır. Girilen veriyi (harf) yazdırdığımızda, çıktı o harfin ascii kodu olacaktır.

```
import java.io.IOException;

public class Demo {

	public static void main(String[] args) {
		int sayı;
		System.out.println("Bir karekter giriniz:");
		try {
		    sayı = System.in.read();
		    System.out.print("Girdiğiniz karekterin ascii kodu : ");
		    System.out.println(sayı);
		} catch (IOException e) {
		    System.out.println("Giriş okunmadı");
		}
	}
}
```

```
/**
Bir karekter giriniz:
t
Girdiğiniz karekterin ascii kodu : 116
*/
```

Bu program, klâvyeden girilen karekterin `ascii` kodunu `System.in.read()` metodu ile okuyor, sonra onu ilgili karektere dnüştürüyor ve ekrana girilen karektei yazıyor. Önceki programdan tek farkı

```
	ch = (char) System.in.read();
```

deyiminin yaptığı tip dönüşümüdür (casting).

```
import java.io.IOException;

public class Demo {

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

```
                char ch;
                System.out.println("Bir karekter giriniz:");
                try {
                        ch = (char) System.in.read();
                        System.out.print("Girdiğiniz karekter :" + ch);
                } catch (IOException e) {
                        System.out.println("Giriş okunmadı");
                }
        }
}
```

Program: Klâvyeden byte array'ine giriş

```
class KlavyedenOku2 {
  public static void main (String args[]) {
    byte dizin[] = new byte[80];
    try {
      System.in.read(dizin);
    }
    catch (Exception e) {
      System.out.println("Error: " + e.toString());
    }
    String str = new String(dizin, 0);
    System.out.println(str);
  }
}
```

`KlavyedenOku2` sınıfı `dizin` adlı `byte` tipinden 80 byte uzunlukta bir array yaratıyor; yani ana bellekte `dizin` için 80 karakter tutabilecek bir yer ayırıyor. `in` değişkeni `InputStream` tipinden olduğu için, onun metotlarını kullanabilir.

```
int read(byte[] b)
```

metodu, girilen veriyi, anabellekteki b adlı byte arrayi üzerine yazar. Bu demektir ki, `System.in.read(dizin)` deyimi klâvyeden `Enter` tuşuna basılana kadar girilen karakter dizisini bir seferde okur ve `dizin` array'ine atar. Girilen karakter dizisinde 80 karakter (byte) olmak zorunda değildir. Enter tuşuna basılınca, karakter dizisi sonlanmış olur. `Try/catc` hata işleme bloku `read()` metodunda oluşabilecek hatayı yakalar. Hiçbir hata oluşmazsa yalnızca `try` altbloku çalışır, `catch` altblokuna girilmez; catch altblokundan sonraki ilk deyime geçilir. Bir hata oluşursa hemen `catch` altblokuna geçer; bu bloktaki `catch(Exception e)` fonksiyonu, oluşan hatayı yakalar ve `e` değişkenine atar. `e.toString()` fonksiyonu, bu hatayı `String` tipine dönüştürür. `System.out.println("Error: " + e.toString())` fonksiyonu oluşan hatayı ekrana yazar ve sonraki deyim(ler)e geçmeden program akışını durdurur. Bir hata oluşmazsa, `catch` altblokundan sonraki ilk deyim olan `String str = new String(dizin, 0)` deyimine geçer.

Bu deyim `str` adlı `String` tipinden bir nesne (object) yaratır; yani ana bellekte ona bir yer ayırır. Buraya `dizin` array'ine atanan karakterleri aktarır. Ancak, bu atama işlemi sırasında 1 byte (8-bit) uzunlıktaki standart ASCII kodları 2-byte (16-bit) uzunluktaki Unicode karakter kodlarına dönüşür. Bu dönüşüm yapılırken, 8 bitlik ascii kodları unicode'un alt byte'ına yerleşir, üst byte 0 ile doldurulur. Bu işlem, ascii kodlarını unicode'a dönüştürmek için gerekli ve yeterlidir. Son deyimdeki `System.out.println(str)` fonksiyonu `str` parametresinin değerini standart çıkışa (ekrana) gönderir. Bu değer, klâvyeden girilen ve `dizin` array'inden aktarılan metindir.

Program: Klâvyeden Sayılı Giriş

```
class KlavyedenOku3 {
  public static void main (String args[]) {
    byte dizin[] = new byte[10];
    try {
      System.in.read(dizin, 0, 10);
    }
    catch (Exception e) {
      System.out.println("Error: " + e.toString());
    }
    String str = new String(dizin, 0);
    System.out.println(str);
  }
}
```

`KlavyedenOku3` sınıfı `KlavyedenOku2` sınıfına çok benzer, ama arada önemli bir fark vardır. Burada yaratılan `dizin[]` nesnesinin (array) uzunluğu 10 byte olarak belirlenmiştir. `InputStream` sınıfının `read(dizin, 0, 10)` metodu 10 byte okuyup *unicode*'e dönüştürerek *dizin* adlı array'e atayacaktır. Bu teknik, standart girişten istenilen sayıda karekter okumayı sağlar.

Program 5. Klâvyeden sayı girişi

```
class Demo {
      public static void main(String args[]) {
              char input = (char) -1;
              int say;
              System.out.println("0 ile 10 arasında bir sayı giriniz :");
              try {
                      input = (char) System.in.read();
              } catch (Exception e) {
                      System.out.println("Error: " + e.toString());
              }
              say = Character.digit(input, 10);
              if ((say > 0) && (say < 10)) {
                      for (int i = 1; i <= say; i++)
                              System.out.println(i);
              } else
```

```
                        System.out.println("Girdiğiniz sayı 0 ile 10
arasında değil!");
        }
}
```

```
/**
0 ile 10 arasında bir sayı giriniz :
6
1
2
3
4
5
6
*/
```

Program ByteOku.java

Bu program klavyeden girilen karekterleri byte olarak okur, karektere çevirir ve ekrana yazar.

```
import java.io.*;

public class ByteOku {

        public static void main(String[] args) throws IOException {
                byte veri[] = new byte[10];
                try {
                        System.out.print(" Bir şeyler yaz ve Entere bas");
                        System.in.read(veri);
                        System.out.println("Girdiğiniz en çok ilk 10
karekter:");
                        for (int i = 0; i < veri.length; i++) {
                /*
                * Konsola yazmak için, girilenleri karektere dönüştürmeliyiz
                */
                              System.out.print((char) veri[i]);
                               }
                                System.out.println();
                } catch (IOException e) {
                                System.out.print("Veri okunamadı");
                }
        }
```

`read()` metodunun -1 vermesi, giriş akımının sonuna gelindiği anlamına gelir. 8-bitlik byte verilerin 32-bitlik `int` adresinde asla işaret bitini 1 yapamayacağına dikkat ediniz. Ayrıca `Character` sınıfının

```
        static int digit(char ch, int radix)
```

metodu, `ch` karekterini `radix` tabanına göre tamsayıya dönüştürür. Tabii, `ch`'nın sayıya dönüşebilir bir veri olması gerekir.

```
class Demo {
	public static void main(String args[]) {
		char input = (char) -1;
		int numToCount;
		System.out.println("0 ile 10 arasında bir sayı giriniz :");
		try {
			input = (char) System.in.read();
		} catch (Exception e) {
			System.out.println("Error: " + e.toString());
		}
		numToCount = Character.digit(input, 10);
		if ((numToCount > 0) && (numToCount < 10)) {
			for (int i = 1; i <= numToCount; i++)
				System.out.println(i);
		} else
			System.out.println("Girdiğiniz sayı 0 ile 10
arasında değil!");
	}
```

```
/**
0 ile 10 arasında bir sayı giriniz :
6
1
2
3
4
5
6
*/
```

Alıştırmalar

1. Aşağıdaki program koşarken 123 sayısı girilirse, 49 değerini yazar. Neden?

```
import java.io.*;

class Demo {
	public static void main(String args[]) {
		int ch = 0;
		System.out.println("Enter value");

		try {
```

```
                        ch = System.in.read();
                        System.out.println("Value entered:" + ch);
                } catch (IOException io) {
                        System.out.println("Error from user");
                }
        }
}
```

Yanıt:

`System.in.read()` metodu her seferinde bir byte okur. Kullanıcı 123 girdiğinde, bunun ilk byte'ı olan 1 okunur. 1'in Unicode değeri 49 dur.