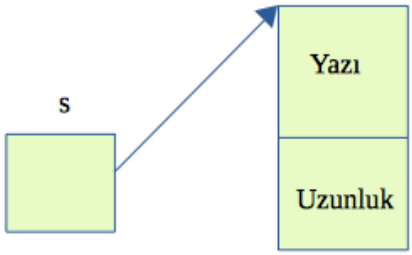
String Sınıfı

Yazılar üzerinde işlemler hemen hemen tüm uygulamalarda yapılmaktadır. Java'da yazılar üzerinde işlem yapan ve java.lang paketi içerisinde bildirilmiş String isimli bir sınıf bulunmaktadır. Bu sınıf java.lang paketi içerisinde bulunduğundan String ismi doğrudan kullanılmaktadır:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.lang.String s1;
		String s2;
		//…
	}
}
```

Bir String nesnesi yaratıldığında nesnenin içerisinde yazısı ve onun uzunluğu tutulur:



Bir String nesnesi String sınıfının çeşitli başlangıç metotlarıyla yaratılabilir. String nesneleri daha çok otomatik yaratılmaktadır. Java'da ne derleyici iki tırnak içerisinde bir yazı gördüğünde new operatörü ile bir String nesnesi yaratır, iki tırnak içerisindeki yazıyı bu nesneye kopyalar ve onun referansını verir. Yani, Java' da "ankara" gibi bir ifade "bir String nesnesi yarat, içerisine ankara yazısını yerleştir, onun referansını ver" anlamına gelmektedir. Dolayısıyla iki tırnak içerisindeki ifadeleri biz String sınıfına doğrudan atayabiliriz:

```
package csd;

class App {
	public static void main(String[] args)
	{
		String s;
		s = "ankara";

		System.out.println(s);	//println s ye bakarak nesnenin karakterlerini ekrana
basar
	}
}
```

System.out.println ve System.out.print metotları bir String referansı ile çağrıldıklarında o referansın gösterdiği nesne içerisindeki yazıyı ekrana yazdırırlar:

```
package csd;

class App {
	public static void main(String[] args)
	{
		String s;
		s = "ankara";
```

```
		System.out.println(s);
		System.out.print(s);
	}
}
```

System.out.printf ve System.out.format metotlarında %s format karakteri ile bir String ekrana yazdırılabilmektedir:

```
package csd;

class App {
	public static void main(String[] args)
	{
		String s;

		s = "Java";

		System.out.printf("Merhaba %s%n", s);
	}
}
```

String sınfının length metodu yazının karakter uzunluğunu döndürmektedir. Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		String s;

		s = "zonguldak";

		System.out.printf("%s yazısı %d karakterden oluşuyor", s, s.length());
	}
}
```

Scanner sınıfının nextLine metodu klavyeden bir yazı girilip “enter” tuşuna basılana kadar bekler. Girilen yazıyı bir String nesnesinin içerisine yerleştirir ve o String referansı ile geri döner. Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir Yazı giriniz");

		String s = kb.nextLine();

		System.out.printf("%s yazısı %d karakterden oluşuyor", s, s.length());

		kb.close();
	}
}
```

String sınfının charAt metodu parametresi ile aldığı indeksteki karakteri döndürür. Yazının ilk karakteri sıfır numaralı indekstedir.

```
public char charAt(int index)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		String s;

		s = "zonguldak";

		int len = s.length();

		for (int i = 0; i < len; ++i)
			System.out.println(s.charAt(i));
	}
}
```

Metot çağrılırken indeks değeri negatif ya da büyük verilirse bu durumda exception oluşur ve program çöker.

NOT: Bir String nesnesinin karakterleri nesne yaratıldıktan sonra değiştirilemez. String sınıfının bu çok önemli özelliğine İngilizce "immutable" denilmektedir. Bu nedenle ***String nesnesi içerisindeki yazı üzerinde işlem yapan metotlar asıl yazıyı değiştirmediklerinden değiştirilmiş yeni bir yazıyı içeren String referansı verirler.***

String sınıfının toLowerCase ve toUpperCase metotları küçük harfe ve büyük harfe dönüştürülmüş yeni bir String nesnesi verirler.

```
public String toLowerCase()
public String toUpperCase()
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir Yazı giriniz");
		String s = kb.nextLine();

		String k = s.toUpperCase();

		System.out.println(k);

		k = s.toLowerCase();

		System.out.println(k);

		kb.close();
	}
}
```

String sınıfının overload edilmiş iki tane substring metodu vardır.

```
public String substring(int beginIndex)
public String substring(int beginIndex, int endIndex)
```

Birinci substring metodu belli bir indeksten başlayarak sonuna kadar olan yazıyı alır ve bize bir String nesnesi olarak verir. İkinci substring metodu birinci parametresindeki beginIndex'ten itibaren endIndex – 1

‘e kadar yazıyı alarak yeni bir String nesnesi ile geri döndürür. Yeni yazının uzunluğu endIndex – beginIndex kadardır. Eğer index uzunluk bakımından sınır dışına çıkarsa exception oluşur.

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		String s = "istanbul";

		String k = s.substring(2);
		System.out.println(k); // tanbul

		k  = s.substring(2, 4);
		System.out.println(k); // ta
	}
}
```

Uzunluğu 0 (sıfır) olan String’e boş string denilmektedir. Boş string iki tane iki tırnak karakterinin bitişik yazılmasıyla da elde edilebilir. Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		String s;

		s = ""; // Boş string

		System.out.printf("Len:%d%n", s.length());
	}
}
```

Boş string de bellekte bir nesne olarak tahsis edilir. Bir string nesnesinin boş string olup olmadığı isEmpty metodu ile kolaylıkla test edilebilir. Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		String s;

		s = "";

		System.out.printf("%s", s.isEmpty() ? "Boş String" : s);
	}
}
```

String sınıfının yazı içerisindeki bir karakteri ya da bir yazıyı arayan indexOf metotları bulunmaktadır. Bu metotlar karakteri ya da yazıyı bulduklarında, ilk bulunan karakterin ya da ilk bulunan yazının ilk karakterin indeks numarasıyla geri dönerler. Bulamazsa -1 değeri ile geri dönerler. Overload edilmiş indexOf metotları şunlardır:

```
public int indexOf(int ch)
public int indexOf(int ch, int fromIndex)
```

```
public int indexOf(String str)
public int indexOf(String str, int fromIndex)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir Yazı giriniz");
		String s = kb.nextLine();
		char ch = 'ç';

		int index = s.indexOf(ch);

		System.out.println(index);

		if (index != -1)
			System.out.println("Aranan karakter var");
		else
			System.out.println("Aranan karakter yok");


		kb.close();
	}
}
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci Yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci Yazıyı giriniz");
		String s2 = kb.nextLine();

		int index = s1.indexOf(s2);

		System.out.println(index);

		if (index != -1)
			System.out.println("Aranan yazı var");
		else
			System.out.println("Aranan yazı yok");

		kb.close();
	}
}
```

Sınıfın belirli bir indeksten itibaren arama yapan indexOf metotları o indeksten itibaren ilk buldukları indeks numarasını ya da bulamazsa -1 değerini döndürürler.

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci Yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci Yazıyı giriniz");
		String s2 = kb.nextLine();

		int index = s1.indexOf(s2, 3);

		System.out.println(index);

		if (index != -1)
			System.out.println("Aranan yazı var");
		else
			System.out.println("Aranan yazı yok");

		kb.close();
	}
}
```

String sınıfının lastIndexOf metotları tamamen indexOf metotları gibi kullanılır. Ancak bu metotlar ilk bulunan yerin indeksiyle değil son bulunan yerin indeksiyle geri dönerler. Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci Yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci Yazıyı giriniz");
		String s2 = kb.nextLine();

		int index = s1.lastIndexOf(s2);

		System.out.println(index);

		if (index != -1)
			System.out.println("Aranan yazı var");
		else
			System.out.println("Aranan yazı yok");

		kb.close();
	}
}
```

Sınıf Çalışması: Klavyeden iki yazı isteyiniz. Birinci yazıda ikinci yazıdan kaç tane olduğunu bulan programı yazınız.

Çözüm:

```
package csd;

class App {
```

```
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Ana yazıyı giriniz");
		String text = kb.nextLine();

		System.out.println("Aranacak yazıyı giriniz");
		String searchText = kb.nextLine();

		int index = -1;
		int textCounter = 0;

		while ((index = text.indexOf(searchText, index + 1)) != -1)
			textCounter++;

		if (textCounter  > 0)
			System.out.printf("%s yazısı içerisinde %s yazısından %d adet var%n", text,
                                                                searchText,
textCounter);
		else
			System.out.println("Bulunamadı");


		kb.close();
	}
}
```

Alternatif bir çözüm şöyle olabilir:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Ana yazıyı giriniz");
		String text = kb.nextLine();

		System.out.println("Aranacak yazıyı giriniz");
		String searchText = kb.nextLine();
		int index, textCount;

		for (index = 0, textCount = 0; (index = text.indexOf(searchText, index)) != -1;
++textCount, +                          +index)
			;

		if (textCount  > 0)
			System.out.printf("%s yazısı içerisinde %s yazısından %d adet var%n", text,
searchText,                textCount);
		else
			System.out.println("Bulunamadı");

		kb.close();
	}
}
```

Sınıf Çalışması: Klavyeden mutlak bir yol ifadesi (absolute path) isteyiniz. Yol ifadesindeki dosya ismini uzantı hariç yazdırınız. Örneğin, "\home\csd\temp\test.dat" gibi bir girişte "test" yazısının yazdırılması gerekir. Yol ifadesinde dosyanın uzantısı olmak zorunda değildir.

Çözüm:

```
package csd;
```

```
import java.util.Scanner;

class App{
	public static void main(String [] args)
	{
		String path, fileName;
		int index;

		Scanner kb = new Scanner(System.in);
		System.out.println("Bir yol ifadesi (path) giriniz:");
		path = kb.nextLine();

		if((index = path.lastIndexOf('\\')) == -1)
			System.out.println("Girilen yol ifadesi geçerli değil!");


		 fileName = path.substring(index + 1);
		    if ((index = fileName.indexOf('.')) != -1)
		        fileName = fileName.substring(0, index);

		    System.out.println(fileName);

		kb.close();

	}
}
```

String sınıfının trim metodu baştaki ve sondaki boşluk karakterlerini (white space) atar. Metot boşluk karakterleri atılmış yeni bir yazıyı bize verir.

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir Yazı giriniz");
		String s = kb.nextLine();

		s = s.trim();

		System.out.println(s);

		kb.close();
	}
}
```

Sınıf Çalışması: Klavyeden girilen yazının tamamı boşluk karakterlerinden oluştuğu sürece yeni yazı isteyen ve girilen yazının tamamını büyük harfe çeviren programı yazınız.

Çözüm:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		String s;
```

```
		for (;;) {
			System.out.println("Bir yazı giriniz");
			s = kb.nextLine();

			if (!s.trim().isEmpty()) //trim metodunun döndürdüğü değer String türünden olduğundan dönen String için isEmpty çağrılır
				break;

			System.out.println("Giriş zorunludur");
		}

		s = s.toUpperCase();

		System.out.println(s);

		kb.close();
	}
}
```

2. Çözümü:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		String s;

		for (;;) {
			System.out.println("Bir yazı giriniz");
			s = kb.nextLine();

			if (s.trim().length() != 0)
				break;

		System.out.println("Giriş zorunludur");
		}

		s = s.toUpperCase();

		System.out.println(s);

		kb.close();
	}
}
```

3. Çözümü:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		String s;

		System.out.println("Bir yazı giriniz");

		while ((s = kb.nextLine()).trim().isEmpty())
			System.out.println("Giriş zorunludur");

		s = s.toUpperCase();
```

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

		kb.close();
	}
}
```

4. Çözümü:

```
package csd;

class App {
	public static boolean isAllWhiteSpace(String s)
	{
		return s.trim().isEmpty();
	}

	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		String s;

		for (;;) {
			System.out.println("Bir yazı giriniz");
			s = kb.nextLine();
			if (!isAllWhiteSpace(s))
				break;

			System.out.println("Giriş zorunludur");
		}

		s = s.toUpperCase();

		System.out.println(s);

		kb.close();
	}
}
```

Sınıf Çalışması: Klavyeden girilen email adres bilgisinin @ ve boşluk karakterlerine (white-space) göre geçerliliğinin kontrolünü yapan programı yazınız.

Çözüm:

```
package csd;

class App {
	public static String controlEmail(String email)
	{
		int index = email.indexOf('@');

		if (email.indexOf('') >= 0 || email.indexOf('\t') >= 0 || email.indexOf('\n') >= 0
				|| index < 0 || index != email.lastIndexOf('@') || index == 0
				|| index == email.length() - 1)
			return "";

		return email;
	}

	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("EMail adresinizi giriniz");
		String email = kb.nextLine();
```

```
		if (!controlEmail(email).isEmpty()) {
			int atIndex = email.indexOf('@');
			System.out.printf("UserName:%s%n", email.substring(0, atIndex));
			System.out.printf("ServerName:%s%n", email.substring(atIndex +1));
		}
		else
			System.out.printf("Geçersiz format");

		kb.close();
	}
}
```

String sınıfının contains isimli metodu, bir yazının belirli bir yazıyı içerip içermediğine göre true veya false ile geri dönmektedir.

```
public boolean contains(CharSequence s)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);
		System.out.println("Birinci yazıyı giriniz");
		String text = kb.nextLine();
		System.out.println("İkinci yazıyı giriniz");
		String searchText = kb.nextLine();

		if (text.contains(searchText))
			System.out.println("Yazı bulundu");
		else
			System.out.println("Yazı bulunamadı");

		kb.close();
	}
}
```

String türünden referansların adres karşılaştırılması == ve != ile yapılmaktadır. Bunun dışında string'ler >, <, >= ve <= operatörleriyle karşılaştırma işlemine sokulamazlar. Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci yazıyı giriniz");
		String s2 = kb.nextLine();

		if (s1 == s2) // Dikkat adresler karşılaştırılıyor
			System.out.println("Aynı nesne");
		else
			System.out.println("Aynı nesne değil");

		kb.close();
	}
}
```

Derleyici iki tırnak içerisinde özdeş Stringler gördüğünde bunlar için tek bir yer ayırır. Hepsi için ayrı nesneler yaratmaz.Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		String s1, s2;

		s1 = "ankara";
		s2 = "ankara";

		System.out.println(s1 == s2); //true
	}
}
```

String sınıfının compareTo metodu iki String'i sözlük sırasına göre karşılaştırır. Örneğin, s1.compareTo(s2) metot çağrılmasında metot çalışılan dile göre (tabloya göre) eğer s1 içerisindeki yazı s2 içerisindeki yazıdan sonra geliyorsa pozitif, önce geliyorsa negatif döndürür. Eğer tüm karakterler aynıysa sıfır değerini döndürür. Sıfır döndürmesi, equals(Object) metodunun true olarak döndürmesi ile aynı anlama gelmektedir.

```
public int compareTo(String anotherString)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		String s1, s2;

		s1 = "ankara";
		s2 = "Ankara";

		System.out.println(s1.compareTo(s2)); //32
	}
}
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci yazıyı giriniz");
		String s2 = kb.nextLine();

		if (s1.compareTo(s2) >= 0)
			System.out.printf("%s,%s%n", s2, s1);
		else
			System.out.printf("%s,%s%n", s1, s2);

		kb.close();
	}
}
```

String sınıfının compareToIgnoreCase metodu ise küçük – büyük harf duyarlılığı olmaksızın iki String'in karşılaştırılmasını sağlamaktadır.

```
public int compareToIgnoreCase(String str)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci yazıyı giriniz");
		String s2 = kb.nextLine();

		if (s1.compareToIgnoreCase(s2) == 0)
			System.out.println("Aynı yazı");
		else
			System.out.println("Farklı yazı");

		kb.close();
	}
}
```

İki String referansı + operatörüyle toplama işlemine sokulabilir. (Ancak çıkartma, çarpma ve bölme gibi başka işlemlere sokulamaz.) Bu durumda yeni bir String nesnesi yaratılır. Bu yeni nesne iki String nesnesinin içerisindeki yazıların birleşiminden oluşmaktadır. Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci yazıyı giriniz");
		String s2 = kb.nextLine();

		String s3 = s1 + s2;

		System.out.println(s3);

		kb.close();
	}
}
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci yazıyı giriniz");
		String s = kb.nextLine();
```

```
		System.out.println("Benim adım"+ "" + s);

		kb.close();
	}
}
```

Burada önce “Benim adım” yazısına boşluk eklenmiştir. Sonra bu elde edilen yeni yazıya s referansı eklenmiştir.

String ile herhangi bir türün toplanması işleminde yeni bir String nesnesi elde edilmektedir.

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir sayı giriniz");
		int val = kb.nextInt();

		int result = val * val;

		System.out.println("Sayının karesi: " + result);

		kb.close();
	}
}
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir sayı giriniz");
		int val = kb.nextInt();

		int result = val * val;

		System.out.println(val + " sayının karesi: " + result);

		kb.close();
	}
}
```

İki tırnak içerisindeki yazılar iki ayrı satıra yazılamaz. Tek satırda bulunmak zorundadır. Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		System.out.println("Bugun hava ç     //error
				ok güzel"+ "" + "mi?");

	}
}
```

Bu tür durumlarda + operatörü ile yazıları satırlara bölebiliriz. Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		System.out.println("Bugun hava ç"
				+ "ok güzel"+ "" + "mi?"); // Bugun hava çok güzel mi?

	}
}
```

İki tırnak ifadeleri String referansı belirttiğine göre doğrudan işleme sokulabilir. Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		System.out.println("ankara".length());

	}
}
```

a = a + b işlemi kısaca a += b şeklinde yazılabilmektedir. += operatörünü String'lerle de kullanabiliriz.

Sınıf Çalışması: Klavyeden "exit" yazısı girilene kadar alınan yazıların aralarına virgül ekleyerek yeni bir String'de birleştirip ekrana yazdıran programı yazınız.

Çözüm:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Yazıları girmeye başlayınız");
		String s, result = "";

		for (;;) {
			s = kb.nextLine();
			if (s.compareToIgnoreCase("exit") == 0)
				break;

			result += result.isEmpty() ? s : ", " + s;
		}

		if (!result.isEmpty())
			System.out.println(result);
		else
			System.out.println("Hiçbir yazı girilmedi");

		kb.close();
	}
}
```

2. Çözümü:

```
package csd;

class App {
```

```
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Yazıları girmeye başlayınız");
		String s, result = "";

		while ((s = kb.nextLine()).compareToIgnoreCase("exit") != 0)
			result += result.isEmpty() ? s : ", " + s;

		if (!result.isEmpty())
			System.out.println(result);
		else
			System.out.println("Hiç bir yazı girilmedi");

		kb.close();
	}
}
```

String sınıfının concat isimli metodu bir yazının sonuna başka bir yazıyı eklemek için kullanılır. Metodun parametresi eklenecek yazıyı belirtir. Metot eklenmiş yeni yazıyla geri döner.

```
public String concat(String str)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci sayıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci sayıyı giriniz");
		String s2 = kb.nextLine();

		String s3 = s1.concat(s2); // s3 = s1 + s2;

		System.out.println(s3);

		kb.close();
	}
}
```

Sınıf Çalışması: Klavyeden girilen bir yazının tersini ekrana yazdıran programı yazınız.

Çözüm:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir yazı giriniz");
		String s = kb.nextLine();

		for (int i = s.length() - 1; i >= 0; --i)
			System.out.print(s.charAt(i));
```

```
		kb.close();
	}
}
```

2.Çözüm:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir yazı giriniz");
		String str = kb.nextLine();

		String revStr = "";

		for (int i = str.length() - 1; i >= 0; --i)
			revStr += str.charAt(i);

		System.out.println(revStr);

		kb.close();
	}
}
```

Sınıf Çalışması: Klavyeden girilen bir yazının palindrom olup olmadığını test eden isPalindrom metodunu yazınız ve test ediniz. Not: Boşluk kontrolü yapılmayacak.

Çözüm:

```
package csd;

class App {
	public static String reverse(String s)
	{
		String result = "";

		for (int i = s.length() - 1; i >= 0; --i)
			result += s.charAt(i);

		return result;
	}

	public static boolean isPalindrom(String s)
	{
		//Boşluklar atılmıyor
		return reverse(s).compareTo(s) == 0;
	}

	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir yazı giriniz");
		String str = kb.nextLine();

		if (isPalindrom(str))
			System.out.println("Palindrom");
		else
			System.out.println("Palindrom değil");

		kb.close();
	}
}
```

2.Çözümü: (Hızlı versiyon)

```
package csd;

class App {
	public static String reverse(String s)
	{
		String result = "";

		for (int i = s.length() - 1; i >= 0; --i)
			result += s.charAt(i);

		return result;
	}

	public static boolean isPalindrom(String s)
	{
		for (int i = 0; i < s.length() / 2; ++i)
			if (s.charAt(i) != s.charAt(s.length() - 1 - i))
				return false;

		return true;
	}

	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir yazı giriniz");
		String str = kb.nextLine();

		if (isPalindrom(str))
			System.out.println("Palindrom");
		else
			System.out.println("Palindrom değil");

		kb.close();
	}
}
```

3. Çözümü: (Boşluklar temizleniyor.)

```
package csd;

class App {
	public static String reverse(String s)
	{
		String result = "";

		for (int i = s.length() - 1; i >= 0; --i)
			result += s.charAt(i);

		return result;
	}

	public static boolean isPalindrom(String s)
	{
		s = clearWS(s);

		for (int i = 0; i < s.length() / 2; ++i)
			if (s.charAt(i) != s.charAt(s.length() - 1 - i))
				return false;

		return true;
	}

	public static String clearWS(String s)
	{
```

```
		int len = s.length();
		String result = "";

		for (int i = 0; i < len; ++i)
			if (!Character.isWhitespace(s.charAt(i)))
				result += s.charAt(i);

		return result;
	}

	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir yazı giriniz");
		String str = kb.nextLine();

		if (isPalindrom(str))
			System.out.printf("\"%s\" Palindromdur%n", str);
		else
			System.out.printf("\"%s\" Palindrom değildir%n", str);


		kb.close();
	}
}
```

String sınıfının startsWith metodu bir String'in belirli bir yazıyla başlayıp başlamadığını test eder. İki parametreli metodunda ise, birinci parametresindeki yazıyı, ikinci parametresinde belirtilen indexten itibaren arar.

```
public boolean startsWith(String prefix)
public boolean startsWith(String prefix, int toffset)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Web adresini giriniz");
		String s = kb.nextLine();

		if (!s.startsWith("http://"))
			s = "http://" + s;

		System.out.println(s);

		kb.close();
	}
}
```

String sınıfının equals metodu, String'i belirtilen Object ile karşılaştırır. Eğer argüman null değil ve karakterleri aynı ise true döndürür. Diğer durumlarda false ile geri döner.

```
public boolean equals(Object anObject)
```

Örneğin:

```
package csd;
```

```
class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci yazıyı giriniz");
		String s2 = kb.nextLine();

		if (s1.equals(s2))
			System.out.println("Aynı");
		else
			System.out.println("Aynı değil");

		kb.close();
	}
}
```

String sınıfının equalsIgnoreCase metodu ise iki String'i büyük-küçük harf duyarlılığı olmaksızın karşılaştırmaktadır. Eğer argüman null değil ve karakterler aynı ise true, aksi durumlarda false dönmektedir.

```
public boolean equalsIgnoreCase(String anotherString)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci yazıyı giriniz");
		String s2 = kb.nextLine();

		if (s1.equalsIgnoreCase(s2))
			System.out.println("Aynı");
		else
			System.out.println("Aynı değil");

		kb.close();
	}
}
```

Character sınıfının toUpperCase metodu, bir karakterin Unicode tablosundaki büyük harfine çevirir.

```
public static char toUpperCase(char ch)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		char ch = 'a';
		char upperCh = Character.toUpperCase(ch);

		System.out.println(upperCh);
	}
}
```

Character sınıfının toLowerCase metodu, bir karakterin Unicode tablosundaki küçük harfine çevirir.

```
public static char toLowerCase(char ch)
```

Sınıf Çalışması: Klavyeden girilen yazının boşluklarını tümüyle atarak ve ilk harfini büyük geri kalanlarını küçük olacak şekilde yeni bir yazı üreten programı yazınız.

Çözüm:

```
package csd;

class App {
	public static String clearWS(String s)
	{
		int len = s.length();
		String result = "";

		for (int i = 0; i < len; ++i)
			if (!Character.isWhitespace(s.charAt(i)))
				result += s.charAt(i);

		return result;
	}

	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir yazı giriniz");
		String s = kb.nextLine();

		s = clearWS(s);

		if (!s.isEmpty())
			s = Character.toUpperCase(s.charAt(0)) +
				(s.length() != 1 ? s.substring(1).toLowerCase() : "");

		System.out.println(s);

		kb.close();
	}
}
```

2. Çözüm:

```
package csd;

class App {
	public static String clearWS(String s)
	{
		int len = s.length();
		String result = "";

		for (int i = 0; i < len; ++i)
			if (!Character.isWhitespace(s.charAt(i)))
				result += s.charAt(i);

		return result;
	}

	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);
```

```
		System.out.println("Bir yazı giriniz");
		String s = kb.nextLine();

		s = clearWS(s);

		if (!s.isEmpty())
			s = s.substring(0, 1).toUpperCase() +
				(s.length() != 1 ? s.substring(1).toLowerCase() : "");

		System.out.println(s);

		kb.close();
	}
}
```

String sınıfının valueOf metotları, aldığı argümanları String' e çevirirler:

```
public static String valueOf(boolean b)
public static String valueOf(char c)
public static String valueOf(char[] data)
public static String valueOf(char[] data, int offset, int count)
public static String valueOf(double d)
public static String valueOf(float f)
public static String valueOf(int i)
public static String valueOf(long l)
public static String valueOf(Object obj)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir sayı giriniz");
		int val = kb.nextInt();

		String s = String.valueOf(val);

		System.out.println(s);

		kb.close();
	}
}
```

Integer sınıfının parseInt metodu aldığı String argümanını işaretli decimal integer'a parse etmektedir. String argümanın sadece baş karakteri hariç tüm karakterleri decimal olmak zorundadır. Baş karakteri eksi işareti(-) ise negatif değere, artı işareti(+) ise pozitif değere çevirmektedir.

```
public static int parseInt(String s)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir sayı giriniz");
		String s = kb.nextLine();
```

```
		int val = Integer.parseInt(s);

		System.out.println(val * val);

		kb.close();
	}
}
```

Double sınıfının parseDouble metodu, aldığı String argümanı parse ederek yeni bir double ile geri döner.

```
public static double parseDouble(String s) throws NumberFormatException
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir sayı giriniz");
		String s = kb.nextLine();

		double val = Double.parseDouble(s);

		System.out.println(val * val);

		kb.close();
	}
}
```

String sınıfının format metodu

```
public static String format(Locale l, String format, Object... args)
public static String format(String format, Object... args)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Reel kısmı giriniz");
		double re = kb.nextDouble();

		System.out.println("Sanal kısmı giriniz");
		double im = kb.nextDouble();

		String s = String.format("%f + %f.i%n", re, im);

		System.out.println(s);

		kb.close();
	}
}
```

String sınıfının replace metotları, bir yazı içerisindeki belli bir karakteri başka bir karakterle ya da yazı içerisindeki belli bir yazıyı başka bir yazı ile yer değiştirmek için kullanılır.

```
public String replace(char oldChar, char newChar)
```

```
public String replace(CharSequence target, CharSequence replacement)
public String replaceAll(String regex, String replacement)
public String replaceFirst(String regex, String replacement)
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir yazı giriniz");
		String s = kb.nextLine();

		s = s.replace('a', 'b');

		System.out.println(s);

		kb.close();
	}
}
```

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci yazıyı giriniz");
		String s2 = kb.nextLine();

		System.out.println("Üçüncü yazıyı giriniz");
		String s3 = kb.nextLine();

		s1 = s1.replaceAll(s2, s3);

		System.out.println(s1);

		kb.close();
	}
}
```

Sınıf Çalışması: Klavyeden girilen bir yazı içerisinden yine klavyeden girilen ikinci yazıyı silen programı yazınız.

Çözüm:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci yazıyı giriniz");
		String s1 = kb.nextLine();
```

```
		System.out.println("İkinci yazıyı giriniz");
		String s2 = kb.nextLine();

		s1 = s1.replaceAll(s2, "");

		System.out.println(s1);

		kb.close();
	}
}
```

Sınıf Çalışması: Klavyeden girilen bir yazı içerisinden yine klavyeden girilen ikinci yazıdan ilkini silen programı yazınız.

Çözüm:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci yazıyı giriniz");
		String s2 = kb.nextLine();

		int index = s1.indexOf(s2);

		if (index >= 0)
			s1 = s1.substring(0, index) + s1.substring(index + s2.length());

		System.out.println(s1);

		kb.close();
	}
}
```

2. Çözüm: Bulunan kelimeden sonra boşluk varsa atılıyor.

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Birinci yazıyı giriniz");
		String s1 = kb.nextLine();

		System.out.println("İkinci yazıyı giriniz");
		String s2 = kb.nextLine();

		int index = s1.indexOf(s2);

		if (index >= 0)
			s1 = s1.substring(0, index)
			+ (s1.length() != index + s2.length() &&
			s1.charAt(index + s2.length()) == '' ? s1.substring(index + s2.length()     +
1) : s1.substring(index + s2.length()));

		System.out.println(s1);
```

```
		kb.close();
	}
}
```

Java 1.7 den itibaren String türü switch deyimi ile de kullanılabilmektedir. case bölümlerinde String atomu (iki tırnak içerisinde yazı) olması zorunludur.

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Bir yazı giriniz");
		String s = kb.nextLine();
		String q = "quit";

		switch (s) {
		case "list":
			System.out.println("Listele");
			break;
		case "clear":
			System.out.println("Ekranı temizle");
			break;
		case "cp":
			System.out.println("Kopyala");
			break;
		case q:                                  // error!
			System.exit(-1);
		default:
			System.out.println("Geçersiz Komut");
		}

		kb.close();
	}
}
```

Anahtar Notlar: String türünden final olarak bildirilmiş değişkenler de switch deyiminin case bölümlerinde kullanılabilmektedir. Bu konu ileride ele alınacaktır.

String atomları içerisinde (iki tırnak içerisinde) ters bölü bazı karakterlerle birlikte kullanıldığında daha önceden belirlenmiş işlemleri yapmaktadırlar. Bunlara escape squence denilmektedir. Bazıları şunlardır:

\t	tab karakteri
\n	newline (line feed) karakteri
\r	carriage-return karakteri
\f	form-feed karakteri
\e	escape karakteri

Örneğin:

```
package csd;

class App {
	public static void main(String[] args)
	{
		String s;

		s = "C:\test\names.txt";
```

```
            System.out.println(s);          // C: est
                                            // ames.txt
        }
}
```

Escape squence kullanılmak istenmiyorsa ters bölü işareti çift yazılmak zorundadır. Örneğin:

```
package csd;

class App {
        public static void main(String[] args)
        {
                String s = "C:\\test\\names.txt";

                System.out.println(s);    // C:\test\names.txt
        }
}
```

Örneğin:

```
package csd;

class App {
        public static void main(String[] args)
        {
             String s;

             s = "C:\oest\names.txt"; //Error!Tek ters bölüden sonra escape sequence gelmiyor

             System.out.println(s);
        }
}
```

String atomları içerisinde (iki tırnak içerisinde) tek tırnak karakteri ters bölü ile kullanılabilir. Örneğin:

```
package csd;

class App {
        public static void main(String[] args)
        {
                String s;

                s = "'\'ankara";

                System.out.println(s); // ''ankara
        }
}
```

Sınıf Çalışması: Dosya yol ifadeleri üzerinde işlemler yapan aşağıdaki metotları Util isimli sınıfta yazınız ve test ediniz:

- getFileExtension: Metot parametresi ile aldığı yol ifadesine ilişkin dosyanın uzantısını döndürecektir.

- getFileName: Metot parametresi ile aldığı yol ifadesine ilişkin dosyanın ismini döndürecektir.

- getFileNameWithoutExtension: Metot parametresi ile aldığı yol ifadesine ilişkin dosyanın uzantısız ismini döndürecektir.

- getParentPath: Metot parametresi ile aldığı yol ifadesine ilişkin dosyanın bulunduğu dizini döndürecektir.

Çözüm:

```
package csd;
```

```
class App {
	public static void main(String [] args)
	{
		java.util.Scanner kb = new java.util.Scanner(System.in);

		System.out.println("Yol ifadesini giriniz");
		String path = kb.nextLine();

		System.out.printf("File Name:%s%n", Util.getFileName(path));
		System.out.printf("File Extension:%s%n", Util.getFileExtension(path));
		System.out.printf("Just File Name:%s%n", Util.getFileNameWithoutExtension(path));
		System.out.printf("Parent:%s%n", Util.getParentPath(path));

		kb.close();
	}
}

class Util {
	public static String getParentPath(String path)
	{
		int index = getLastSeparatorIndex(path);

		return index == -1 ? "" : path.substring(0, index);
	}

	public static int getLastSeparatorIndex(String path)
	{
		int index1=  path.lastIndexOf('\\');
		int index2=  path.lastIndexOf('/');

		return index1 > index2 ? index1 : index2;
	}

	public static String getFileName(String path)
	{
		int index = getLastSeparatorIndex(path);

		return path.substring(index + 1);
	}

	public static String getFileExtension(String path)
	{
		String fileName = getFileName(path);
		int index = fileName.lastIndexOf('.');

		return index == -1 ? "" : fileName.substring(index);
	}

	public static String getFileNameWithoutExtension(String path)
	{
		String fileName = getFileName(path);
		int index = fileName.lastIndexOf('.');

		return index == -1 ? fileName : fileName.substring(0, index);
	}
}
```