The class DataInputStream allows to read primitive Java data types from an underlying input stream. It reads the file line by line. In the given example, we have created an instance of DataInputStream class which wraps the FileInputStream. The method available() returns 0 if the file does not have more lines. The method readLine() reads the line from the file.

import java.io.\*;  
  
public class FileDataInputStream {  
        public static void main(String[] args) throws Exception {  
                File file = new File("C:/data.txt");  
                DataInputStream dis = new DataInputStream(new FileInputStream(file));  
                while (dis.available() != 0) {  
                        System.out.println(dis.readLine());  
                }  
                dis.close();  
        }  
}

In this example we are reading  number of lines.

In this example we are reading a file from system .Then we are reading number of lines in this file. **FileReader** class provides **readLine()** method.The  **readLine()** method is used to read the data line by line. **LineNumberReader** class is provide **getLineNumber()** method to get the number of line. We are using **File(String** file\_name**)** constructor to make an object of file. We are reading file by use of **FileReader(File file)** constructor. We are using **LineNumberReader(FileReader** filereader**)** constructor to create an object of **LineNumberReader** class.

**The methods used in this example:**

**getLineNumber():**This method is used to get the number of line.

**readLine():**This method is used to read the data line by line. Then returns the data of a line as a single string.

String indexOf() method example:-

String class method indexOf() is use to find the starting character position of the substring from the string. The indexOf() method start to search from beginning of the string and matching the substring from substring, if substring is match then it return a index position of the starting character of the substring within string.

int [**indexOf**](http://download.oracle.com/javase/1.4.2/docs/api/java/lang/String.html#indexOf%28int,%20int%29)(int ch, int fromIndex)   
          Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.

Here, you will get to know about the **indexOf(String str, int fromIndex)** method through the following java program. In the program code given below, we have taken a String. To find the index of any string we have applied **indexOf(String str, int fromIndex);** method and then we have passed the Substring from the specified string with the index number in that to find its index starting from the specified index number as shown in the output. And if we take index number to be 23 then we get -1 as a result that shows the given index number doesn't exist in a string.

**NOTE: This method returns the index of the first occurrence of the given substring, starting at the specified index within the string.**

**Here is the code of the program:**

|  |  |
| --- | --- |
| |  | | --- | | **public class**IndexOfStr {   **public static void**main(String args[]) {   String s = "That was the breaking News";   System.out.println(s);   System.out.println("indexOf(News, 5) -> " + s.indexOf("News", 5));   }   } | |

**Output of the program:**

|  |
| --- |
| **C:\unique>javac IndexOfStr.java  C:\unique>java IndexOfStr That was the breaking News indexOf(News, 5) -> 22  C:\unique>javac IndexOfStr.java  C:\unique>java IndexOfStr That was the breaking News indexOf(News, 23) -> -1** |

String toUpperCase() and toLowerCase() methods example:-

String class provides methods toUpperCase() and toLowerCase(). Method toUpperCase() convert alphabetical characters in the string to uppercase and method toLowerCase() convert alphabetical characters in the string to  lowercase. We have used these two methods in this

The split() method in Java is used to split the given string followed by regular expression in   
Java. For example you can check the string for "," "/" etc.. and split the string into multiple   
Java String Objects. See the example given below...  
  
Syntax of Java split() method...  
**String[] split(String regex) Syntax  
or   
String[] split(String regex, int limit)**

a simple example of split() method in Java

public class example  
{  
public static void main(String[] args)  
{  
String data = "one two three four";  
String[] items = data.split(" ");  
  
for (String item : items)  
{  
System.out.println(item);  
}  
}  
}

the Output is:  
one  
two  
three  
four

In this section, you will get the detailed explanation about the ***equalsIgnoreCase(String Str)*** method of **String** class. We are going to use ***equalsIgnoreCase(String Str)*** method of String class in Java. The description of the code is given below for the usage of the method.  
  
**Description of the code:**

As shown in the example below we have taken three strings i.e. **str1, str2** and **str3**. Then we have applied the ***equalsIgnoreCase(String Str)*** method to compare these three strings. This method shows whether two String objects contain the same data, ignoring the case of the letters in the String. Hence we get the following output.

**If atleast one of the following is true then only two characters c1 and c2 are considered the same, ignoring case.**

* **Applying the method Character.toUpperCase(char) to each character produces the same result.**
* **Applying the method Character.toLowerCase(char) to each character produces the same result.**
* **The two characters are the same (as compared by the == operator).  
    
  Parameters:  
  anotherString -** the String to compare other String.  
     
  **Returns true if the argument is not null and the Strings are equal, ignoring case. Otherwise returns true.**

**Here is the code of the program:**

|  |  |
| --- | --- |
| |  | | --- | | **public class**equalsIgnore{   **public static void**main(String[] args){   String str1 = "bon voyage";   String str2 = "Bon Appetit";   String str3 = "Bon Voyage";     **boolean**equals1 = str1.equalsIgnoreCase(str2);   **boolean**equals2 = str1.equalsIgnoreCase(str3);     System.out.println("\"" + str1 + "\" equals \"" + str2    + "\"? " + equals1);   System.out.println("\"" + str1 + "\" equals \"" + str3    + "\"? " + equals2);   } } | |

**Here is the code of the program:**

|  |
| --- |
| **C:\unique>javac equalsIgnore.java  C:\unique>java equalsIgnore "bon voyage" equals "Bon Appetit"? false "bon voyage" equals "Bon Voyage"? true** |

The Vector class implements a growable array of objects. Like an array, it contains components that can be accessed using an integer index. However, the size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created.

public Vector createData()  
{  
    Vector result = new Vector();  
    Vector rec = new Vector();  
    Object[] value = new Object[] {"Employee Name",f.ename.getSelectedItem()};  
    rec.add(value);  
    value = new Object[] {"Mary-Kate Olsen", "Ashley Olsen"};  
    rec.add(value);  
    result.add(rec);  
    rec = new Vector();  
    value = new Object[] {"Charlie Read", "Craig Read"};  
    rec.add(value);  
    value = new Object[] {"Tegan Quin", "Sara Quin"};  
    rec.add(value);  
    result.add(rec);  
    return result;  
}

void [addElement](http://download.oracle.com/javase/1.4.2/docs/api/java/util/Vector.html#addElement%28java.lang.Object%29)([Object](http://download.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html) obj)   
          Adds the specified component to the end of this vector, increasing its size by one.