

Iterators and File Input

CSC 2014 – Java Bootcamp

Dr. Mary-Angela Papalaskari

Department of Computing Sciences

Villanova University

Course website:

www.csc.villanova.edu/~map/2014/

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & Loftus

Iterators

- **Iterating:** to process a collection of items, one at a time
- Typical iterator methods:
 - **next()** returns the next item
 - **hasNext()** - returns true if there is at least one more item to process
- Several classes in the Java standard class library are iterators

Tokens

- items to be processed are called **tokens**
 - Examples: words, numbers, components of a url...
- The **Scanner** class is an iterator
 - **next()** returns the next scanned token (a String)
 - **nextLine()** returns the rest of the line (until the next new line).
 - **hasNext()** returns true if there is more data to be scanned
- Variations of **next()** and **hasNext()** methods:

nextInt()	hasNextInt()
nextDouble()	hasNextDouble()

Using Scanner to read from a file

- Create a File object:

```
File myFile = new File("sample.txt");
```

- Create a Scanner to read from the File object:

```
Scanner fileScan = new Scanner (myFile);
```

- Use **next()** to obtain next token

- Use **nextLine()** to obtain entire line of text (until \n)

- Use **hasNext()** to test whether you are done

File Input Example: FileInput.java

```
import java.util.Scanner;
import java.io.*;

public class FileInput
{
    //-----
    // Reads text from a file and prints it in uppercase.
    //-----
    public static void main (String[] args) throws IOException
    {
        String line;

        File myFile = new File("sample.txt");
        Scanner fileScan = new Scanner (myFile));

        // Read and process each line of the file
        while (fileScan.hasNext())
        {
            line = fileScan.nextLine();
            System.out.println (line.toUpperCase());
        }

    }
}
```

File Input Example: FileInput.java

```
import java.util.*;
import java.io.*;

public class FileInput {
    //-----//
    // Reads text
    //-----//
    public static void main() {
        String line;
        File myFile = new File("sample.inp");
        Scanner fileScan = new Scanner (myFile));
        // Read and process each line of the file
        while (fileScan.hasNext())
        {
            line = fileScan.nextLine();
            System.out.println (line.toUpperCase());
        }
    }
}
```

sample.inp

Computers are useless. They can only give you answers.
Pablo Picasso (1881 - 1973)

Run Output

COMPUTERS ARE USELESS. THEY CAN ONLY GIVE YOU ANSWERS.
PABLO PICASSO (1881 - 1973)

Try this: What gets printed?

```
*****  
// SomethingToDoWithFiles.java      Author: MAP  
*****  
import java.util.Scanner;  
import java.io.*;  
  
public class SomethingToDoWithFiles  
{  
    public static void main (String[] args) throws IOException  
    {  
        String line1, line2;  
        Scanner fileScan1, fileScan2;  
  
        fileScan1 = new Scanner (new File("sample1.inp"));  
        fileScan2 = new Scanner (new File("sample2.inp"));  
  
        while (fileScan1.hasNext() && fileScan2.hasNext())  
        {  
            line1 = fileScan1.nextLine();  
            line2 = fileScan2.nextLine();  
            System.out.println (line1 + line2 + line1);  
        }  
        System.out.println(fileScan1.hasNext() ? "ping!": "pong!");  
    }  
}
```

sample1.inp

do
re
mi

sample2.inp

fa
sol

Scanner – another example: reading from a file AND from a String

- Suppose we wanted to read and process a list of URLs (or other data items) stored in a file
- One scanner can be set up to read each line of the input until the end of the file is encountered
- Another scanner can be set up to process each line, i.e., separating the components of each URL (at each occurrence of '/')
- Example:
URL: `www.linux.org/info-gnu.html`
This URL specifies a path consisting of the following components:
 - `www.linux.org`
 - `info`
 - `gnu.html`
- See [URLDissector.java](#)

```
//*****  
//  URLDissector.java      Author: Lewis/Loftus  
//  
//  Demonstrates the use of Scanner to read file input and parse it  
//  using alternative delimiters.  
//*****  
  
import java.util.Scanner;  
import java.io.*;  
  
public class URLDissector  
{  
    //-----  
    //  Reads urls from a file and prints their path components.  
    //-----  
    public static void main (String[] args) throws IOException  
    {  
        String url;  
        Scanner fileScan, urlScan;  
  
        fileScan = new Scanner (new File("urls.inp"));  
  
    continue
```

continue

```
// Read and process each line of the file
while (fileScan.hasNext())
{
    url = fileScan.nextLine();
    System.out.println ("URL: " + url);

    urlScan = new Scanner (url);
    urlScan.useDelimiter("/");
    // Print each part of the url
    while (urlScan.hasNext())
        System.out.println ("    " + urlScan.next());

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

default delimiter is **white space**,
we are changing this here

Sample Run

```
continue
```

```
// Read
```

```
while
```

```
{
```

```
url
```

```
Sys
```

```
www.google.com  
www.google.com
```

```
URL: www.linux.org/info/gnu.html
```

```
www.linux.org
```

```
info
```

```
gnu.html
```

```
url
```

```
url
```

```
URL: thelyric.com/calendar/
```

```
thelyric.com
```

```
calendar
```

```
//
```

```
whi
```

```
Sys
```

```
URL: www.cs.vt.edu/undergraduate/about
```

```
www.cs.vt.edu
```

```
undergraduate
```

```
about
```

```
}
```

```
}
```

```
URL: youtube.com/watch?v=EHCRimwRGLs
```

```
youtube.com
```

```
watch?v=EHCRimwRGLs
```