File I/O in Java

Md. Saidul Hoque Anik
onix.hoque.mist@gmail.com

The `File` class

Retrieves information about files

```
import java.io.File;

public class LabDemo {
    public static void main(String []args)
    {
        File f = new File("sample.txt");
        System.out.println(f.exists());
    }
}
```

The `File` class

Methods of File class

`boolean canRead()`	Returns `true` if a file is readable by the current application; `false` otherwise.
`boolean canWrite()`	Returns `true` if a file is writable by the current application; `false` otherwise.
`boolean exists()`	Returns `true` if the file or directory represented by the `File` object exists; `false` otherwise.
`boolean isFile()`	Returns `true` if the name specified as the argument to the `File` constructor is a file; `false` otherwise.
`boolean isDirectory()`	Returns `true` if the name specified as the argument to the `File` constructor is a directory; `false` otherwise.
`boolean isAbsolute()`	Returns `true` if the arguments specified to the `File` constructor indicate an absolute path to a file or directory; `false` otherwise.

The `File` class

Methods of File class

`String getAbsolutePath()`	Returns a `String` with the absolute path of the file or directory.
`String getName()`	Returns a `String` with the name of the file or directory.
`String getPath()`	Returns a `String` with the path of the file or directory.
`String getParent()`	Returns a `String` with the parent directory of the file or directory (i.e., the directory in which the file or directory is located).
`long length()`	Returns the length of the file, in bytes. If the `File` object represents a directory, an unspecified value is returned.
`long lastModified()`	Returns a platform-dependent representation of the time at which the file or directory was last modified. The value returned is useful only for comparison with other values returned by this method.
`String[] list()`	Returns an array of `String`s representing a directory's contents. Returns `null` if the `File` object does not represent a directory.

Writing in a file

`FileWriter` class

```
public static void main(String []args)
{
    File f = new File("sample.txt");
    FileWriter fr;
    fr = new FileWriter(f);
    fr.write("Hello World");
    fr.close();
    System.out.println("Done");
}
```

Writing in a file

`FileWriter` class

```
public static void main(String []args)
{
    File f = new File("sample.txt");
    FileWriter fr;
    try {
        fr = new FileWriter(f);
        fr.write("Hello World");
        fr.close();
        System.out.println("Done");
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Reading from a file

FileReader Class

```
public static void main(String []args)
{
    File f = new File("sample.txt");
    FileReader fr;
    try {
        fr = new FileReader(f);
        char c[] = new char[5];
        fr.read(c);
        System.out.println(c);
        fr.read(c);
        System.out.println(c);
        fr.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Reading from a file

Reading the whole file

```
public static void main(String []args)
{
    File f = new File("sample.txt");
    FileReader fr;
    try {
        fr = new FileReader(f);
        int c = fr.read();
        while (c != -1)
        {
            System.out.printf("%c", c);
            c = fr.read();
        }
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

BufferedReader, a better way to read

BufferedReader Class

```
public static void main(String []args)
{
    File f = new File("sample.txt");
    FileReader fr;
    try {
        fr = new FileReader(f);
        BufferedReader br = new BufferedReader(fr);
        System.out.println(br.readLine());
        fr.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

BufferedReader, a better way to read

Reading the whole file, line by line

```
public static void main(String []args)
{
    File f = new File("sample.txt");
    FileReader fr;
    try {
        fr = new FileReader(f);
        BufferedReader br = new BufferedReader(fr);

        String s = br.readLine();
        while (s != null)
        {
            System.out.println(s);
            s = br.readLine();
        }
        fr.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

`Scanner class`, an even better way of reading

Reading the whole file, line by line

```
public static void main(String []args)
{
    File f = new File("sample.txt");
    FileReader fr;
    try {
        fr = new FileReader(f);
        Scanner br = new Scanner(fr);

        while (br.hasNextLine())
        {
            System.out.println(br.nextLine());
        }

        fr.close();
        br.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

`Scanner class`, an even better way of reading

We can even read word by word

```
public static void main(String []args)
{
    File f = new File("sample.txt");
    FileReader fr;
    try {
        fr = new FileReader(f);
        Scanner br = new Scanner(fr);

        while (br.hasNext())
        {
            System.out.println(br.next());
        }

        fr.close();
        br.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Reading and Writing

String and other types

```
public static void main(String []args)
{
    File f = new File("sample.txt");
    FileWriter fr;
    try {
        fr = new FileWriter(f);
        fr.write("Hello ");
        fr.write("10 ");
        fr.write("3.1416 ");
        fr.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

Continue to next slide…

Reading and Writing

String and other types

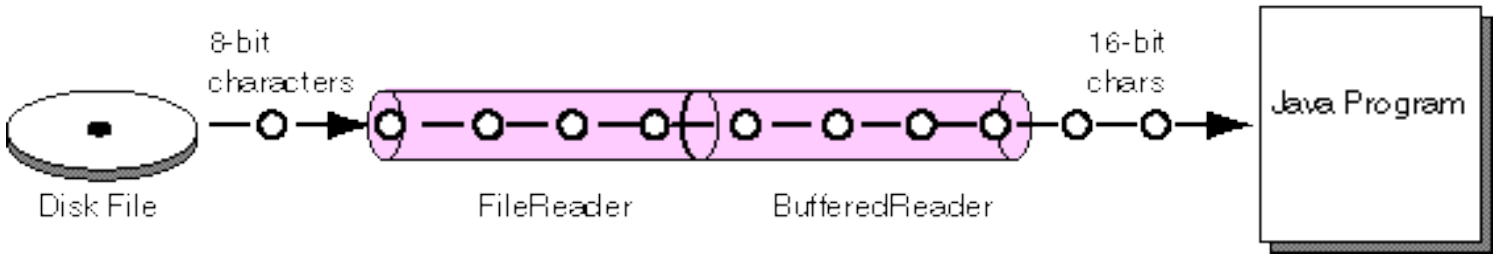
```
        FileReader fr_;
        try {
            fr_ = new FileReader(f);
            Scanner br = new Scanner(fr_);
            String s = br.next();
            int i = br.nextInt();
            double d = br.nextDouble();

            double sum = d + i*1.0;
            System.out.println(s + " " + sum);
            fr_.close();
            br.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

…continued from previous slide.

FileReader vs. BufferedReader

FileReader vs. BufferedReader

BufferedReader improves efficiency



http://www.oopweb.com/Java/Documents/JavaNotes/Volume/chap84/ch84_3.html

BufferedReader vs. Scanner

BufferedReader vs. Scanner

BufferedReader just reads the stream

Scanner is able to parse (and tokenize) the stream

Task 1

Implement an Address Book Class that will be able to

- Save the following information:
  - Name,
  - Phone number,
  - Address

    Address will have the following fields:
    - House Number,
    - Street name,
    - City
- Display the information

Task 2

Open the given booklist.csv file and

- Show the contents of the file in tabular manner
- Show the books in category (paperback, hardback, ebook)
- Store the contents in arrays of objects
- Implement SeachByName and SearchByAuthor
- Show Books having price greater than a specific value. Take the value as input

Task 3

Implement a Word Processor Class that will be able to

- Show the contents of a file (Take the filename as input)
- Show the number of lines, number of words and number of characters
- Find whether a specific word is present in the document or not
- Replace the specific word with another word

Task 4

Implement a Desktop Search Program That will take

- An input string: “Line”
- A Directory: “C:\New Folder”

The program will search in all the text files of that directory, and print the filename if the file contains that string.