JAVA - IO

Introduction

JDK has two sets of I/O packages:

1. the Standard I/O (in package java.io), introduced since JDK 1.0 for stream-based I/O, and

2. the New I/O (in packages java.nio), introduced in JDK 1.4, for more efficient buffer-based I/O.

JDK 1.5 introduces the formatted text-I/O via new classes java.util.Scanner and Formatter, and C-like printf() and format() methods for formatted output using format specifiers.

JDK 1.7 enhances supports for file I/O via the so-called NIO.2 (non-blocking I/O) in new package java.nio.file and its auxiliary packages. It also introduces a new try-with-resources syntax to simplify the coding of close() method

Files and Directory

- java.io.File <= jdk 1.7 => java.nio.File.Path (overcome the limitation of java.io.File
- A path string is used to locate a file or a directory) - Platform dependent
- A path could be absolute (beginning from the root) or relative (which is relative to a reference directory).
- Directory Separator: in static fields **File.separator (as String)** and File.separatorChar. [They failed to follow the Java naming convention for constants adopted since JDK 1.2.] As mentioned, Windows use backslash '\'; while Unixes/Mac use forward slash '/'.
- Path Separator: in static fields **File.pathSeparator (as String)** and File.pathSeparatorChar. As mentioned, Windows use semi-colon ';' to separate a list of paths; while Unixes/Mac use colon ':'.

Continued...

```
public File(String pathString)

public File(String parent, String child)

public File(File parent, String child)

// Constructs a File instance based on the given path string.

  public File(URI uri)

 // Constructs a File instance by converting from the given file-URI "file://...."
```

Continued...

```
File file = new File("in.txt");     // A file relative to the current working directory

File file = new File("d:\\myproject\\java\\Hello.java"); // A file with absolute path

File dir  = new File("c:\\temp");   // A directory
```

For applications that you intend to distribute as JAR files, you should use the URL class to reference the resources

```
java.net.URL url = this.getClass().getResource("icon.png");
```

Verifying Properties of a File/Directory

```
public boolean exists()         // Tests if this file/directory exists.

public long length()            // Returns the length of this file.

public boolean isDirectory()    // Tests if this instance is a directory.

public boolean isFile()         // Tests if this instance is a file.

public boolean canRead()        // Tests if this file is readable.

public boolean canWrite()       // Tests if this file is writable.

public boolean delete()         // Deletes this file/directory.

public void deleteOnExit()      // Deletes this file/directory when the program terminates.

public boolean renameTo(File dest) // Renames this file.
```

List Directory

```
public boolean mkdir()          // Makes (Creates) this directory.


public String[] list()      // List the contents of this directory in a String-array

public File[] listFiles()   // List the contents of this directory in a File-array
```

Example

```
// Recursively list the contents of a directory (Unix's "ls -r" command).

import java.io.File;

public class ListDirectoryRecusive {

   public static void main(String[] args) {

      File dir = new File("d:\\myproject\\test"); // Escape sequence needed for '\'

      listRecursive(dir);

   }
```

```
 public static void listRecursive(File dir) {

      if (dir.isDirectory()) {

         File[] items = dir.listFiles();

         for (File item : items) {


System.out.println(item.getAbsoluteFile());

            if (item.isDirectory())
listRecursive(item); // Recursive call
}       }    }}
```

(Advanced) List Directory with File

You can apply a filter to list() and listFiles(), to list only files that meet a certain criteria.

```
public String[] list(FilenameFilter filter)
```

```
public File[] listFiles(FilenameFilter filter)
```

```
public File[] listFiles(FileFilter filter)
```

The interface java.io.FilenameFilter declares one abstract method:

```
public boolean accept(File dirName, String fileName)
```