File I/O

Part 2: Text Input/Output

Chapter 2, Core Java, Volume II &

Chapter 15, Java How to Program

Contents

Text Input/Output

- Text files contain characters.
- When saving text strings, we must consider Character Encoding.
  - Example: integer 1234 is saved as a sequence of bytes 00 00 04 D2(hex) in binary format
  - Example: Using UTF-16: It is encoded as : 00 31 00 32 00 33 00 34(hex) in text format
- Java uses Unicode for characters.
  - Internally, Java uses the UTF-16 encoding.
  - Web pages commonly use UTF-8.
  - Files on desktop computers sometimes use legacy encodings (e.g. Windows 1252).
- Readers/writers convert between bytes and characters.
- Always specify the character encoding!
  - Use `StandardCharsets.UTF_8` for `Charset` parameters,
  - Use `"UTF-8"` for `String` parameters.
  - To get the default Charset

```
Charset cs = Charset.defaultCharset();
String name = cs.displayName();    // x-windows-949 for windows (Korean version)
```

Text Input/Output

- You can obtain a Reader for any byte-based input stream:

```
InputStream inStream = ....................................................; // byte-based stream
Reader in = new InputStreamReader(inStream, charset);
```

- The read() method reads one char value.
  - That's too low-level for most purposes.
- You can read a line from a file:

```
BufferedReader in = new BufferedReader(new FileReader(filenameString));
String s = in.readLine();
```

- You can read a file as a string:

```
String content = new String( Files.readAllBytes(path), charset);
```

- You can get all lines as a list or stream:

```
List<String> lines = Files.readAllLines(path, charset);
```

Text Input/Output

- Use a Scanner to split input into numbers, words, and so on:

```
Scanner in = new Scanner(Paths.get("myfile.txt"), StandardCharsets.UTF-8);
while (in.hasNextDouble())
{
  double value = in.nextDouble() ;
  ……..……;
}
```

- To read words, set the delimiter to with spaces before and after a comma :

```
in.useDelimiter("\\s*,\\s*");     // \s matches a space whitespace character
while (in.hasNext()) {
   String word = in.next();
   …
```

Text Input/Output

- To open a file to write:

```
PrintWriter out = new PrintWriter(filenameString, charsetStr);
PrintWriter out = new PrintWriter(Files.newBufferedWriter(path, charset));
```

- To write text:

```
out.print, out.println, or out.printf
```

- Remember to close the file as follows :

```
try (PrintWriter out = ...)
{
    ……..;
}
```

- If you already have the entire output in as string, or a collection of lines, call:

```
Files.write(path, contentString.getBytes(charset));
Files.write(path, lines, charset);
```

- You can also append output to a file:

```
Files.write(path, lines, charset, StandardOpenOption.APPEND);
```

Text Input/Output

- If you want to capture the output in a string, not a file, use a StringWriter:

```
StringWriter writer = new StringWriter();
PrintWriter out = new PrintWriter(writer));
// write text to out here
```

- Now you can process the stack trace as string:

```
String s = writer.toString();
```

Example: Text File

```
import java.io.*;
import java.time.*;
import java.util.*;
public class TextFileTest
{
   public static void main(String[] args) throws IOException
   {
      Employee[] staff = new Employee[3];
      staff[0] = new Employee("Carl Cracker", 75000, 1987, 12, 15);
      staff[1] = new Employee("Harry Hacker", 50000, 1989, 10, 1);
      staff[2] = new Employee("Tony Tester", 40000, 1990, 3, 15);
```

Example: Text File

```
   // save all employee records to the file employee.dat
    try (PrintWriter out = new PrintWriter("employee.dat", "UTF-8"))
    {
       writeData(staff, out);
    }
    // retrieve all records into a new array
   try (Scanner in = new Scanner(new FileInputStream("employee.dat"), "UTF-8"))
    {
       Employee[] newStaff = readData(in);

       for (Employee e : newStaff) // print the newly read employee records to the screen
          System.out.println(e);
    }
} //end of main() method
```

Example: Text File

```
// Writes all employees in an array to a print writer
private static void writeData(Employee[] employees, PrintWriter out) throws IOException
{
   // write number of employees
   out.println(employees.length);
   for (Employee e : employees)
       writeEmployee(out, e);
}
// Writes employee data to a print writer
public static void writeEmployee(PrintWriter out, Employee e)
{
  out.println(e.getName() + "|" + e.getSalary() + "|" + e.getHireDay());
}
```

```
Carl Cracker|75000.0|1987-12-15
```

Example: Text File

```
// Reads an array of employees from a scanner
 private static Employee[] readData(Scanner in)
{
   // retrieve the array size
   int n = in.nextInt();
   in.nextLine(); // consume newline
   Employee[] employees = new Employee[n];
   for (int i = 0; i < n; i++)
   {
      employees[i] = readEmployee(in);
   }
   return employees;
}
```

```
// The format of the file mployee.dat

3
Carl Cracker|75000.0|1987-12-15
3Harry Hacker|50000.0|1989-10-01
Tony Tester|40000.0|1990-03-15
```

Example: Text File

```
// Reads employee data from a buffered reader
   public static Employee readEmployee(Scanner in)
   {
      String line = in.nextLine();
      String[] tokens = line.split("\\|");
      String name = tokens[0];
      double salary = Double.parseDouble(tokens[1]);
      LocalDate hireDate = LocalDate.parse(tokens[2]);
      int year = hireDate.getYear();
      int month = hireDate.getMonthValue();
      int day = hireDate.getDayOfMonth();
      return new Employee(name, salary, year, month, day);
   }
} // end of TextFileTest
```

Example: Text File

```
import java.time.*;
public class Employee
{
   private String name;
   private double salary;
   private LocalDate hireDay;
   public Employee(String n, double s,  int year,
    int month, int day) {
      name = n;salary = s;
      hireDay = LocalDate.of(year, month, day);
   }
   public String getName() {
      return name;
   }
   public double getSalary() {
      return salary;
   }
```

```
   public LocalDate getHireDay() {
      return hireDay;
   }
   public double getSalary() {
      return salary;
   }
   public LocalDate getHireDay() {
      return hireDay;
   }
   public void raiseSalary(double byPercent) {
      double raise = salary * byPercent / 100;
      salary += raise;
   }
   public String toString() {
      return getClass().getName() + "[name=" + name + ",
      salary=" + salary + ",hireDay=" + hireDay + "]";
   }
}
```