Software Practice 1 - File I/O

- Stream I/O
- Buffered I/O
- File I/O with exceptions
- CSV format
- Practice#6

Prof. Joonwon Lee

T.A. Jaehyun Song
Jongseok Kim (42)

T.A. Sujin Oh
Junseong Lee (43)

Basic I/O in Java

- For printing
  - System.out.println ()
  - System.err.println ()

- For scanning from file
  - …?

Standard Streams

- A feature of many operating systems
- Read input from the keyboard and write output to the display
- Also support I/O on files and between programs
  - But these two features are controlled by the command line interpreter, not program

Standard Streams in Java

- **Standard input**
  - `System.in`
    - `object of InputStream with console in`
- **Standard output**
  - `System.out`
    - `object of PrintStream with console out`
    - `PrintStream is inherited from OutputStream`
- **Standard error**
  - `System.err`
    - `object of PrintStream with console err`

Read Data with System.in

```
public class Main {
   public static void main (String[] args) {
      int data = System.in.read ();
      while (data != -1) {
         char theChar = (char) data;
         data = System.in.read ();
      }
   }
}
```

Write Data with System.out

```
public class Main {
    public static void main (String[] args) {
        int[] data = {1, 2, 3};
        System.out.println (data);
    }
}
```

Write Data with System.err

```
public class Main {
    public static void main (String[] args) {
        String errMsg = “Example of System.err!”;
        System.err.println (data);
    }
}
```

How about File I/O?

- **Same with Standard Streams**
  - FileInputStream
    - Stream for reading data from file
  - FileOutputStream
    - Stream for writing data from file
  - ~~FileErrorStream(?)~~
    - File I/O does not have error I/O

- **Usages are also same with Standard Streams!**

* https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html

FileInputStream

- Obtains input bytes from a file in a file system
- Reads streams of raw bytes such as image
- Inherited from class InputStream
- Also has same way to read data

* https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html

Read Data with FileInputStream

```
public class Main {
    public static void main (String[] args) {
        InputStream fis = new FileInputStream (“path”);
        int data = fis.read ();
        while (data != -1) {
            char theChar = (char) data;
            data = fis.read ();
        }
        fis.close ();
    }
}
```

InputStreamReader

- **A bridge from byte streams to character streams**
  - Reads bytes and decodes them into characters using a specified charset

- **Generate reader of the given InputStream object**
- **Declared in java.io package**
- **Inherited from class java.io.Reader**

- **Provides lock for reading data from single resource**

* https://docs.oracle.com/javase/7/docs/api/java/io/InputStreamReader.html

Constructor of Stream Reader

- **InputStreamReader**

Constructor and Description
InputStreamReader(InputStream in) Creates an InputStreamReader that uses the default charset.
InputStreamReader(InputStream in, Charset cs) Creates an InputStreamReader that uses the given charset.
InputStreamReader(InputStream in, CharsetDecoder dec) Creates an InputStreamReader that uses the given charset decoder.
InputStreamReader(InputStream in, String charsetName) Creates an InputStreamReader that uses the named charset.

* https://docs.oracle.com/javase/7/docs/api/java/io/InputStreamReader.html

Read Data with Stream Reader

- **InputStreamReader**

Modifier and Type	Method and Description
void	close() Closes the stream and releases any system resources associated with it.
String	getEncoding() Returns the name of the character encoding being used by this stream.
int	read() Reads a single character.
int	read(char[] cbuf, int offset, int length) Reads characters into a portion of an array.
boolean	ready() Tells whether this stream is ready to be read.

* https://docs.oracle.com/javase/7/docs/api/java/io/InputStreamReader.html

Example of InputStreamReader

```
public class Main {
   public static void main (String[] args) {
      Reader isr = new InputStreamReader (System.in);
      int data = isr.read ();
      String str = “”;
      while (data != -1) {
         str += (char) data;
         data = isr.read ();
      }
      isr.close ();
   }
}
```

Example of InputStreamReader

```
public class Main {
   public static void main (String[] args) {
      InputStream fis = new FileInputStream (“path”);
      Reader isr = new InputStreamReader (fis);
      int data = isr.read ();
      String str = “”;
      while (data != -1) {
         str += (char) data;
         data = isr.read ();
      }
      isr.close ();
   }
}
```

BufferedReader

- **Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines**

- **The buffer size may be specified, or the default size may be used**
  - The default size is large enough to most purposes

* https://docs.oracle.com/javase/7/docs/api/java/io/BufferedReader.html

Example of BufferedReader

```
public class Main {
   public static void main (String[] args) {
      InputStream fis = new FileInputStream (“path”);
      Reader isr = new InputStreamReader (fis);
      Reader br  = new BufferedReader (isr);

      String data = br.readline ();

      br.close ();
   }
}
```

FileOutputStream

- An output stream for writing data to a File or to a FileDescriptor
- Whether or not a file is available or may be created depends upon the underlying platform
- Inherited from class OutputStream
- Also has same way to write data

* https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html

Write Data with FileOutputStream

```
public class Main {
    public static void main (String[] args) {
        OutputStream fos = new FileOutputStream (“path”);
        byte[] data = {‘a’, ‘b’, ‘c’};
        fos.write (data);
        fos.write (data[0]);
        fos.flush ();
        fos.close ();
    }
}
```

OutputStreamWriter

- **A bridge from character streams to byte streams**
  - Characters written to it are encoded into bytes using a specified charset

- **Generate writer of the given OutputStream object**
- **Declared in java.io package**
- **Inherited from class java.io.Writer**

- **Provides lock for writing data to single resource**
- **Highly recommend to use "BufferedWriter"!**

* https://docs.oracle.com/javase/7/docs/api/java/io/OutputStreamWriter.html

Constructor of Stream Writer

- **OutputStreamWriter**

Constructor and Description
OutputStreamWriter(OutputStream out) Creates an OutputStreamWriter that uses the default character encoding.
OutputStreamWriter(OutputStream out, Charset cs) Creates an OutputStreamWriter that uses the given charset.
OutputStreamWriter(OutputStream out, CharsetEncoder enc) Creates an OutputStreamWriter that uses the given charset encoder.
OutputStreamWriter(OutputStream out, String charsetName) Creates an OutputStreamWriter that uses the named charset.

* https://docs.oracle.com/javase/7/docs/api/java/io/OutputStreamWriter.html

Write Data with Stream Writer

- **OutputStreamWriter**

Modifier and Type	Method and Description
void	close() Closes the stream, flushing it first.
void	flush() Flushes the stream.
String	getEncoding() Returns the name of the character encoding being used b y this stream.
void	write(char[] cbuf, int off, int len) Writes a portion of an array of characters.
void	write(int c) Writes a single character.
void	write(String str, int off, int len) Writes a portion of a string.

* https://docs.oracle.com/javase/7/docs/api/java/io/OutputStreamWriter.html

Example of OutputStreamWriter

```
public class Main {
   public static void main (String[] args) {
      Writer osw = new OutputStreamWriter (System.out);
      String data = “abc”;
      osw.write (data, 0, 3);
      osw.write (data, 0, 1);
      osw.flush ();
      osw.close ();
   }
}
```

Example of OutputStreamWriter

```
public class Main {
   public static void main (String[] args) {
      OutputStream fos = new FileOutputStream (“path”);
      Writer osw = new OutputStreamWriter (fos);
      String data = “abc”;
      osw.write (data, 0, 3);
      osw.write (data, 0, 1);
      osw.flush ();
      osw.close ();
   }
}
```

BufferedWriter

- **Writes text to a character-output stream, buffering characters so as to provide for the efficient writing of single characters, arrays, and strings**

- **The buffer size may be specified, or the default size may be used**
  - The default size is large enough to most purposes

* https://docs.oracle.com/javase/7/docs/api/java/io/BufferedReader.html

Example of BufferedWriter

```
public class Main {
    public static void main (String[] args) {
        OutputStream fos = new FileOutputStream (“path”);
        Writer osw = new OutputStreamWriter (fos);
        Writer bw  = new BufferedWriter (osw);
        String data = “abc”;
        bw.write (data, 0, 3);
        bw.write (data, 0, 1);
        bw.flush ();
        bw.close ();
    }
}
```

File does not exist!

- If file not exists, reader (and some cases writer also) cannot do anything
- When program tries to access the file, error will occur
- Preparing for this error, we have to handle the exception with try-catch!

File Related Exceptions

- IOException
  - Signals that an I/O exception of some sort has occurred
  - This class is the general class of exceptions produced by failed or interrupted I/O operations

- FileNotFoundException
  - Signals that an attempt to open the file denoted by a specified pathname has failed

BufferedReader with exception

```
public class Main {
   public static void main (String[] args) {
      Reader isr, br;
      InputStream fis;
      try {
         fis = new FileInputStream (“path”);
         isr = new InputStreamReader (fis);
         br  = new BufferedReader (isr);

         String data = br.readline ();

      } catch (FileNotFoundException | IOException e) {
         e.printStaceTrace ();
      } finally {
         br.close ();
      }
   }
}
```

BufferedWriter with exception

```
public class Main {
   public static void main (String[] args) {
      Writer osw, bw;
      OutputStream fos;
      try {
         fos = new FileOutputStream (“path”);
         osw = new OutputStreamWriter (fos);
         bw  = new BufferedWriter (osw);
         String data = “abc”;
         osw.write (data, 0, 3);
         osw.write (data, 0, 1);
      } catch (IOException e) {
         e.printStackTrace ();
      } finally {
         bw.close ();
      }
   }
}
```

CSV format

- Comma-separated values file stores tabular data in plain text
- Each line of the file is a data record
- Each record consists of one or more fields, separated by commas
- The use of the comma as a field separator is the source of the name for the file format

Example of CSV format

- **Biochemical Oxygen Demand**

  -
    ```
    "","Time","demand"
    "1",1,8.3
    "2",2,10.3
    "3",3,19
    "4",4,16
    "5",5,15.6
    "6",7,19.8
    ```

Example of CSV format

- **Student Admissions at UC Berkeley**

  -
    ```
    "","Admit","Gender","Dept","Freq"
    "1","Admitted","Male","A",512
    "2","Rejected","Male","A",313
    "3","Admitted","Female","A",89
    "4","Rejected","Female","A",19
    ```

Example of CSV format

- **Matrix Representation**

  -
    ```
    1,2,3,4,5
    2,3,4,5,6
    3,4,5,6,7
    ```

How to read them?

- Optionally if the file contains the label, read them first in the outside of loop body
- Read the contents in the loop body with readline method of BufferedReader class
- Handle the I/O related exceptions for missed contents reading or etc.

- https://www.mkyong.com/java/how-to-read-and-parse-csv-file-in-java/

[Lab – Practice #6]

- **Read CSV file without any errors**
- **Calculate the multiplication of two matrices**
- **Implement 3 classes**
  - CSVMatrixReader
  - CSVMatrixWriter
  - CSVMatrixCalculator

[Lab – Practice #6]

- Read number from CSV input file
- All value is large or equal to 0
- Matrix number has float type
- Write only dot(.) or number to output file

[Submit]

- **Upload to i-Campus**
  - Compress your 3 java files to zip file
    - CSVMatrixReader.java
    - CSVMatrixWriter.java
    - CSVMatrixCalculator.java
  - Do not use any package keyword
  - File name: studentID_lab06.zip

- **Due date**
  - Today 23:59:59
    - Class 42 (04/16 Monday)
    - Class 43 (04/18 Wednesday)
  - Penalty: -10% of each lab score per one day