Object Oriented Programming

ICT2122

Introduction to File Handling in Java

P.H.P. Nuwan Laksiri
Department of ICT
Faculty of Technology
University of Ruhuna

Lesson 06

Recap - Exceptions

- What is an exception
- Reasons for exceptions
- Exception handling
- Types of exceptions
- Throwable class
- Catch exceptions
- Finally block
- Throw/Throws
- Declaring own exceptions

Outline

- Getting to Know About Streams
- *Streams*
  - *Character Streams*
  - *Binary Streams*
- *Character Streams*
  - *Reading*
  - *Writing*
- *Binary Streams*
  - *Reading*
  - *Writing*

Getting to Know About Streams

- A stream is simply a ***flow of characters to and from a program***.
- The other end of the stream can be anything that can accept or generate a stream of characters
  - including a console window,
  - a printer,
  - a file on a disk drive,
  - or even another program.
- Streams have no idea of the structure or meaning of your data
  - a stream is just a sequence of characters

Java I/O Streams

- **Character Streams**
  - Character streams read and write text characters that represent strings
  - You can connect a character stream to a text file to store text data on a hard drive
- **Binary Streams**
  - Binary streams read and write individual bytes that represent primitive data types.
  - You can connect a binary stream to a binary file to store binary data on a hard drive

Character Streams - Reading

- **File**
  - The File class represents a file on a hard drive
  - The main purpose of the File class is to identify the file you want to read from or write to

```
File(String path)
File(URI uri)
File(File parent, String child)
File(String parent, String child)
```

Character Streams - Reading

- **FileReader**
  - The FileReader class provides basic methods for reading data from a character stream that originates from a file
  - It provides methods that let you read data one character at a time.

```
FileReader(File file)
FileReader(String path)
```

Character Streams - Reading

- **BufferedReader**
  - This class “wraps” around the FileReader class to provide more efficient input.
  - This class adds a buffer to the input stream that allows the input to be read from the hard drive in large chunks rather than a byte at a time
  - The BufferedReader class lets you read data one character at a time or a line at a time.

BufferedReader(Reader in)

Creating a BufferedReader

- Create a File object for the file

  File f = new File(“student.txt");

- Create a FileReader object

  FileReader fr = new FileReader(f);

- Pass FileReader object to the BufferedReader constructor to create a BufferedReader object

  BufferedReader in = new BufferedReader(fr);

Reading from a Character Stream

- Use the *readLine()* method
- This method returns null when the end of the file is reached

```
String line = in.readLine();
while (line != null)
{
 System.out.println(line);
 line = in.readLine();
}
```

Reading from a Character Stream

- Hands on Session
  - Create a file named "Student.txt" in your computer
    - Refer "Student"
    - Refer "MyFileHandling"

- Don't forget to close the stream

  in.close()

Character Streams - Writing

- **FileWriter**
  - The FileWriter class connects to a File object but provides only rudimentary writing ability

```
FileWriter(File file)
FileWriter(File file, boolean append)
FileWriter(String path)
FileWriter(String path, boolean append)
```

Character Streams - Writing

- **BufferedWriter**
  - This class connects to a FileWriter and provides output buffering.
  - Without the buffer, data is written to the hard drive one character at a time.
  - This class lets the program accumulate data in a buffer and writes the data only when the buffer is filled or when the program requests that the data be written

BufferedWriter(Writer out)

Character Streams - Writing

- **PrintWriter**
  - This class connects to a Writer, which can be aBufferedWriter, a FileWriter, or any other object that extends theabstract Writer class.
  - Most often, you connect this class to a Buffered Writer.

```
PrintWriter(Writer out)
PrintWriter(Writer out, boolean flush)
```

Connecting a PrintWriter to a text file (Replacing)

- Create a File object for the file

  File f = new File(“student.txt");

- Create a FileWriter object

  FileWriter fw = new FileWriter (f);

- Create a BufferedWriter object

  BufferedWriter bw = new BufferedWriter (fw);

- Pass BufferedWriter object to the PrintWriter constructor to create a PrintWriter object

  PrintWriter out = new PrintWriter (bw);

Connecting a PrintWriter to a text file (Appending)

```
File f = new File(“student.txt");

FileWriter fw = new FileWriter(file, true);

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter out = new PrintWriter(bw, true);
```

Writing to a character stream

- Use print() and println() methods

```
System.out.print(“ID”);
System.out.print("\t");
System.out.println(“Name”);

        OR

String line = “ID” + "\t" + “Name”;
System.out.println(line);
```

Writing to a character stream

- Hands on Session
  - Refer "MyCharacterWriting"

- Don't forget to flush

```
out.flush()
out.close()
```

Binary Streams - Reading

- File
  - Once again, you use the File class to represent the file itself

```
File(String path)
File(URI uri)
File(File parent, String child)
File(String parent, String child)
```

Binary Streams - Reading

- **FileInputStream**
  - FileInputStream is what connects the input stream to a file

    FileInputStream File (File file)

    FileInputStream(String path)

Binary Streams - Reading

- **BufferedInputStream**
  - This class adds buffering to the basic FileInputStream, which improves the stream's efficiency and gives it a moist and chewy texture

BufferedInputStream(InputStream in)

Binary Streams - Reading

- **DataInputStream**
  - This class is the one you actually work with to read data from the stream.
  - The other Stream classes read a byte at a time.
  - This class knows how to read basic data types, including primitive types and strings.

DataInputStream(InputStream in)

Creating a DataInputStream

- Create a file object

```
File file = new File(“Students.dat");
```

- Create FileInputStream

```
FileInputStream fs = new FileInputStream(file);
```

- Create BufferedInputStream

```
BufferedInputStream bs = new BufferedInputStream(fs);
```

- Create DataInputStream

```
DataInputStream in = new DataInputStream(bs);
```

Reading from a DataInputStream

- Use the various read methods of the DataInputStream class to read the fields one at a time
- To do that, you have to know the exact sequence in which data values appear in the file

```
String val = in.readUTF();
int num = in.readInt();   etc
```

Reading from a DataInputStream

```
boolean eof = false;
while (!eof){
try{
        String name = in.readUTF();
        int id = in.readInt();
        // do something with the data here
}
catch (EOFException e)
{
        eof = true;
}
catch (IOException e)
{…………..}
```

Reading from a DataInputStream

- Hands on session
  - Refer "MyBinaryReading"

- Don't forget to

```
in.close();
```

Binary Streams - Writing

- **FileOutputStream**
  - Connects to a File object and creates an output stream that can write to the file.
  - This output stream is limited in its capabilities, however, in that it can write only raw bytes to the file.
  - In other words, it doesn't know how to write values such as ints, doubles, or strings.

```
FileOutputStream(File file)
FileOutputStream(File file, boolean append)
FileOutputStream(String path)
FileOutputStream(String path, boolean append)
```

Binary Streams - Writing

- **BufferedOutputStream**
  - This class connects to a FileOutputStream and adds output buffering

BufferedIOutputStream(Output Stream out)

Binary Streams - Writing

- **DataOutputStream**
  - This class adds the ability to write primitive data types and strings to a stream

DataOutputStream(Output Stream out)

Creating a DataOutputStream (Replacing)

- Create the file object

  File file = new File(name);

- Create FileOutputStream

  FileOutputStream fos = new FileOutputStream(file);

- Create BufferedOutputStream

  BufferedOutputStream bos = new BufferedOutputStream(fos);

- Create DataOutputStream

  DataOutputStream out = new DataOutputStream(bos);

Creating a DataOutputStream (Appending)

```
File file = new File(name);

FileOutputStream fos = new FileOutputStream(file, true);

BufferedOutputStream bos = new BufferedOutputStream(fos);

DataOutputStream out = new DataOutputStream(bos);
```

Writing to a binary stream

- Can use various write methods to write different data types to the file

```
out.writeUTF(Stingval);
out.writeInt(Intval);
out.writeDouble(Doubleval);
```

Writing to a binary stream

- Hands on session
  - Refer “MyBinaryWriting”

- Don’t forget to

```
out.flush();
out.close();
```

Homework

- Write a java program that will perform following operations on a .txt file
  - User input - File name (with path)
  - Program must display
    - <filename>
      - No of lines : <amount>
      - No of words : <amount>
      - No of characters : <amount>

Homework

- Study about
  - The java.nio.file package and its related package, java.nio.file.attribute, which provide comprehensive support for file I/O and for accessing the default file system.
  - https://docs.oracle.com/javase/tutorial/essential/io/fileio.html

Summary

- *Streams*
  - *Character Streams*
  - *Binary Streams*
- *Character Streams*
  - *Reading*
  - *Writing*
- *Binary Streams*
  - *Reading*
  - *Writing*

References

- https://docs.oracle.com/javase/tutorial/essential/io/index.html

- How To Program (Early Objects)
  - By H .Deitel and P. Deitel
- Headfirst Java
  - By Kathy Sierra and Bert Bates

Questions ???

Thank You