CS180

File I/O and Networking

Why File I/O?

- Computer/program/JVM not always running
- Sometimes we want to save information onto disk
- Then, we can recall the data at any time
- Remember to close the files you use!

Simple Input Example

- Reading in a set of 3D points to 3 arrays

File format

n

x_1 y_1 z_1

x_2 y_2 z_2

…

x_n y_n z_n

```
Scanner s = null;
try {
   s = new Scanner(new File("input.txt"));
} catch (FileNotFoundException e) {
}
if (s != null) {
   int n = s.nextInt();
   x = new float[n];
   y = new float[n];
   z = new float[n];
   for (int k=0; k<n; k++) {
      x[k] = s.nextFloat();
      y[k] = s.nextFloat();
      z[k] = s.nextFloat();
      System.out.printf("(%f, %f, %f)\n", x, y, z);
   }
   s.close();
}
```

Simple Input Example

- Reading in a set of 3D points to 3 arrays

File format

n

x_1 y_1 z_1

x_2 y_2 z_2

…

x_n y_n z_n

```
Scanner s = null;
try {
   s = new Scanner(new File("input.txt"));
} catch (FileNotFoundException e) {
}
if (s != null) {
   int n = s.nextInt();
   x = new float[n];
   y = new float[n];
   z = new float[n];
   for (int k=0; k<n; k++) {
      x[k] = s.nextFloat();
      y[k] = s.nextFloat();
      z[k] = s.nextFloat();
      System.out.printf("(%f, %f, %f)\n", x, y, z);
   }
   s.close();
}
```

Why declare outside try/catch?

Simple Input Example

- Reading in a set of 3D points to 3 arrays

Design question: better approach than using 3 arrays?

File format

n

x_1 y_1 z_1

x_2 y_2 z_2

…

x_n y_n z_n

```
Scanner s = null;
try {
   s = new Scanner(new File("input.txt"));
} catch (FileNotFoundException e) {
}
if (s != null) {
   int n = s.nextInt();
   x = new float[n];
   y = new float[n];
   z = new float[n];
   for (int k=0; k<n; k++) {
      x[k] = s.nextFloat();
      y[k] = s.nextFloat();
      z[k] = s.nextFloat();
      System.out.printf("(%f, %f, %f)\n", x, y, z);
   }
   s.close();
}
```

Why declare outside try/catch?

Simple Output Example

- Writing out a set of 3D points (which are stored in 3 arrays)

File format

n

x_1 y_1 z_1

x_2 y_2 z_2

…

x_n y_n z_n

```
PrintWriter pw = null;
try {
   pw = new PrintWriter(new FileOutputStream(new File(
     "output.txt")));
} catch (FileNotFoundException e) {
}

if (pw != null) {
   pw.println("" + n);
   for (int k=0; k<n; k++) {
     String s = x[k] + " " + y[k] + " " + z[k];
     pw.println(s);
   }
   pw.close();
}
```

Simple Output Example

- Writing out a set of 3D points (which are stored in 3 arrays)

File format

n

x_1 y_1 z_1

x_2 y_2 z_2

…

x_n y_n z_n

```
PrintWriter pw = null;
try {
   pw = new PrintWriter(new FileOutputStream(new File(
     "output.txt")));
} catch (FileNotFoundException e) {
}

if (pw != null) {
   pw.println("" + n);
   for (int k=0; k<n; k++) {
     String s = x[k] + " " + y[k] + " " + z[k];
     pw.println(s);
   }
   pw.close();
}
```

Where is the call to flush?

Reading and Writing Objects to File

- Previous example only writes primitives!
- Can write objects in one of two ways
  1. Each object eventually can be decomposed to a bunch of primitives and pointers, so write these to file
  2. Use Serializable interface
- Then use ObjectIntputStream and ObjectOutputStream

Networking

- Networks allow us to communicate between multiple computers (LAN, internet, etc.)
- Networking in Java is very similar to File I/O
  - Both are streams
  - Socket is the equivalent of a File in Java networking
- Network basics
  - IP address is the computer ID
  - Sockets/ports receive send data to/from the computer
  - Some ports are reserved (80 for http, 20 for ftp)
    - Safe to use ports > 4000

Client-Server Model

- Server
  - Always listening for connections from clients
  - In Java…
    - Create a ServerSocket object for a given port
    - Wait for connections with accept() method on ServerSocket
      - Returns a Socket object
      - Can attach input/output streams to Socket, then get streams with getInputStream(), getOutputStream()
- Client connects to the server to communicate
  - Create Socket with server's IP address and port number

Server Receiving 3D Points

- Waiting for a client to provide 3D points

Data format

n

x_1 y_1 z_1

x_2 y_2 z_2

…

x_n y_n z_n

```
ObjectIntputStream in;
try {
   int port = 180;  // number tastefully selected
   ServerSocket socket = new ServerSocket(port);
   Socket socket = serverSocket.accept();
   in = new ObjectInputStream(socket.getInputStream());

   int n = in.readInt();
   x = new float[n];
   y = new float[n];
   z = new float[n];
   for (int k=0; k<n; k++) {
      x[k] = in.readFloat();
      y[k] = in.readFloat();
      z[k] = in.readFloat();
   }
} catch (IOException e) {
} finally {
   try {
       in.close();
   } catch (IOException e) {
   }
}
```

Server Receiving 3D Points

- Waiting for a client to provide 3D points

```
ObjectIntputStream in;
try {
   int port = 180;  // number tastefully selected
   ServerSocket socket = new ServerSocket(port);
   Socket socket = serverSocket.accept();
   in = new ObjectInputStream(socket.getInputStream());

   int n = in.readInt();
   x = new float[n];
   y = new float[n];
   z = new float[n];
   for (int k=0; k<n; k++) {
      x[k] = in.readFloat();
      y[k] = in.readFloat();
      z[k] = in.readFloat();
   }
} catch (IOException e) {
} finally {
   try {
      in.close();
   } catch (IOException e) {
   }
}
```

Accept() blocks until a connection is made

Data format

n

x_1 y_1 z_1

x_2 y_2 z_2

…

x_n y_n z_n

Client Sending 3D Points

- Make connection to server

Data format

n

x_1 y_1 z_1

x_2 y_2 z_2

…

x_n y_n z_n

```
ObjectOutputStream out = null;
Socket socket = null;
try {
   String ip = "1.2.3.4";
   int port = 180;  // number tastefully selected
   socket = new Socket(ip, port);
   out =
     new ObjectOutputStream(socket.getOutputStream());

   for (int k=0; k<n; k++) {
     out.writeFloat(x[k]);
     out.writeFloat(y[k]);
     out.writeFloat(z[k]);
   }
} catch (IOException e) {
} finally {
   try {
     if (out != null)
       out.close();
     if (socket != null)
       socket.close();
   } catch (IOException e) {
   }
}
```

Client Sending 3D Points

- Make connection to server

Use ObjectOutputStream to pair with server

```
ObjectOutputStream out = null;
Socket socket = null;
try {
   String ip = "1.2.3.4";
   int port = 180;  // number tastefully selected
   socket = new Socket(ip, port);
   out =
     new ObjectOutputStream(socket.getOutputStream());

   for (int k=0; k<n; k++) {
     out.writeFloat(x[k]);
     out.writeFloat(y[k]);
     out.writeFloat(z[k]);
   }
} catch (IOException e) {
} finally {
   try {
     if (out != null)
       out.close();
     if (socket != null)
       socket.close();
   } catch (IOException e) {
   }
}
```

Output streams don't send data until a flush (which is called by a close)

Terminate connection to server if desired.

Data format

n

x_1 y_1 z_1

x_2 y_2 z_2

…

x_n y_n z_n

Servers on the Web

- Servers on the web need to communicate with multiple clients
  - Does not make sense if first client hogs the entire server
  - Thus, for each client wishing to connect to the server, need to start **new thread** to serve the client
  - Same applies for any server which needs to serve more than one client at a time

Server Supporting Multiple Clients

```
try {
   int port = 5678;
   ServerSocket socket =
      new ServerSocket(port);
   while (true) {
      Socket socket =
         serverSocket.accept();
      (new ClientConnectThread(
         socket)).start();
   }
} catch (IOException e) {
}
```

```
public class ClientConnectThread extends Thread {
   Socket clientSocket;
   int[] x, y, z;

   public ClientConnectThread(clientSocket) {
      this.clientSocket = clientSocket;
   }

   public void run() {
      ObjectInputStream in = null;
      try {
         in = new ObjectInputStream(
            socket.getInputStream());

         int n = in.readInt();
         x = new float[n];
         y = new float[n];
         z = new float[n];
         for (int k=0; k<n; k++) {
            x[k] = in.readFloat();
            y[k] = in.readFloat();
            z[k] = in.readFloat();
         }
      } catch (IOException e) {
      } finally {
         try {
            if (in != null)
               in.close();
            clientSocket.close();
         } catch (IOException e) {
         }
      }
      // do stuff to x, y, z…
   }
}
```

Server Supporting Multiple Clients

Loop continuously waits for next client to connect.

```
try {
   int port = 5678;
   ServerSocket socket =
      new ServerSocket(port);
   while (true) {
      Socket socket =
         serverSocket.accept();
      (new ClientConnectThread(
         socket)).start();
   }
} catch (IOException e) {
}
```

```
public class ClientConnectThread extends Thread {
   Socket clientSocket;
   int[] x, y, z;

   public ClientConnectThread(clientSocket) {
      this.clientSocket = clientSocket;
   }

   public void run() {
      ObjectInputStream in = null;
      try {
         in = new ObjectInputStream(
            socket.getInputStream());

         int n = in.readInt();
         x = new float[n];
         y = new float[n];
         z = new float[n];
         for (int k=0; k<n; k++) {
            x[k] = in.readFloat();
            y[k] = in.readFloat();
            z[k] = in.readFloat();
         }
      } catch (IOException e) {
      } finally {
         try {
            if (in != null)
               in.close();
            clientSocket.close();
         } catch (IOException e) {
         }
      }
      // do stuff to x, y, z…
   }
}
```

Server Supporting Multiple Clients

Loop continuously waits for next client to connect.

```
try {
   int port = 5678;
   ServerSocket socket =
      new ServerSocket(port);
   while (true) {
      Socket socket =
         serverSocket.accept();
      (new ClientConnectThread(
         socket)).start();
   }
} catch (IOException e) {
}
```

Does the code on the client side have to change?

```
public class ClientConnectThread extends Thread {
   Socket clientSocket;
   int[] x, y, z;

   public ClientConnectThread(clientSocket) {
      this.clientSocket = clientSocket;
   }

   public void run() {
      ObjectInputStream in = null;
      try {
         in = new ObjectInputStream(
            socket.getInputStream());

         int n = in.readInt();
         x = new float[n];
         y = new float[n];
         z = new float[n];
         for (int k=0; k<n; k++) {
            x[k] = in.readFloat();
            y[k] = in.readFloat();
            z[k] = in.readFloat();
         }
      } catch (IOException e) {
      } finally {
         try {
            if (in != null)
               in.close();
            clientSocket.close();
         } catch (IOException e) {
         }
      }
      // do stuff to x, y, z…
   }
}
```

Quiz

- What is wrong with the following code which writes integers to a file? What is the result?

```
PrintWriter pw = null;
try {
  pw = new PrintWriter(new FileOutputStream(new File(
    “output.txt”)));
} catch (FileNotFoundException e) {
}

int[] arr = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
for (int k=0; k<arr.length; k++) {
  String s = “” + arr[k];
  pw.println(s);
}
```