Networking Serialization

CSCI 201

Principles of Software Development

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu

USC Viterbi
School of Engineering

Outline

- Network Serialization
- Program

Serialization

- Recall that serialization allows us to write objects to an output stream and read objects from an input stream
- We have already seen serialization with file I/O
- This is shown on the following slides

Serialization with File I/O

```
1   import java.io.Serializable;
2
3   public class Employee implements Serializable {
4     public static final long serialVersionUID = 1;
5     private String fname, lname;
6     private transient String password;
7     public Employee(String fname, String lname, String password) {
8       this.fname = fname;
9       this.lname = lname;
10      this.password = password;
11    }
12    public void printEmployee() {
13      System.out.println(fname + " " + lname + ": " + password);
14    }
15  }
```

Serialization with File I/O

```
1   import java.io.*; // to save space
2   public class EmployeeMain {
3     public static void main(String [] args) {
4       Employee emp = new Employee("donald", "trump", "billionaire");
5       ObjectOutputStream oos = null;
6       ObjectInputStream ois = null;
7       try {
8         oos = new ObjectOutputStream(new FileOutputStream("out.txt"));
9         oos.writeObject(emp);
10        oos.flush();
11        ois = new ObjectInputStream(new FileInputStream("out.txt"));
12        Employee emp1 = (Employee)ois.readObject();
13        emp.printEmployee();
14        emp1.printEmployee();
15      } catch (FileNotFoundException fnfe) {
16        System.out.println("fnfe: " + fnfe.getMessage());
17      } catch (IOException ioe) {
18        System.out.println("ioe: " + ioe.getMessage());
19      } catch (ClassNotFoundException cnfe) {
20        System.out.println("cnfe: " + cnfe.getMessage());
21      } finally {
22        try {
23          if (oos != null)
24            oos.close();
25          if (ois != null)
26            ois.close();
27        }
28        catch (IOException ioe) {
29          System.out.println(“ioe: “ + ioe.getMessage());
30        }
31      } // ends finally
32    }
33  }
```

Network Serialization

- With network communication, we are also able to transmit objects back and forth between two different programs over a `Socket`
- This works almost exactly the same as serialization with file I/O, but instead of a `FileInputStream` and `FileOutputStream`, we will use the input stream and output stream associated with the `Socket`

Serialization with Server Networking

```
1   import java.io.*; // shortened for space
2   import java.net.*; // shortened for space
3   public class NetworkSerializationServer {
4     private ServerSocket ss = null;
5     private Socket s = null;
6     public NetworkSerializationServer() {
7       ObjectInputStream ois = null;
8       ObjectOutputStream oos = null;
9       try {
10        ss = new ServerSocket(6789);
11        s = ss.accept();
12        ois = new ObjectInputStream(s.getInputStream());
13        Employee emp1 = (Employee)ois.readObject();
14        oos = new ObjectOutputStream(s.getOutputStream());
15        oos.writeObject(emp1);
16        oos.flush();
17      } catch (IOException ioe) {
18        System.out.println("ioe: " + ioe.getMessage());
19      } catch (ClassNotFoundException cnfe) {
20        System.out.println("cnfe: " + cnfe.getMessage());
21      } finally {
22        try {
23          if (ois != null)
24            ois.close();
25          if (oos != null)
26            oos.close();
27          if (s != null)
28            s.close();
29          if (ss != null)
30            ss.close();
31        } catch (IOException ioe) {
32          System.out.println("IOE closing streams: " + ioe.getMessage());
33        }
34      } // ends finally
35    }
36    public static void main(String [] args) {
37      new NetworkSerializationServer();
38    }
39  }
```

Serialization with Client Networking

```
1   import java.io.*; // shortened for space
2   import java.net.Socket;
3   public class NetworkSerializationClient {
4     private Socket s = null;
5     public NetworkSerializationClient() {
6       ObjectOutputStream oos = null;
7       ObjectInputStream ois = null;
8       Employee emp = new Employee("donald", "trump", "billionaire");
9       try {
10        s = new Socket("localhost", 6789);
11        ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());
12        oos.writeObject(emp);
13        oos.flush();
14        ObjectInputStream ois = new ObjectInputStream(s.getInputStream());
15        Employee emp1 = (Employee)ois.readObject();
16        oos.close();
17        ois.close();
18        System.out.println("Employee sent");
19        emp.printEmployee();
20        System.out.println("Employee received");
21        emp1.printEmployee();
22      } catch (IOException ioe) {
23        System.out.println("ioe: " + ioe.getMessage());
24      } catch (ClassNotFoundException cnfe) {
25        System.out.println("cnfe: " + cnfe.getMessage());
26      } finally {
27        try {
28          if (ois != null)
29            ois.close();
30          if (oos != null)
31            oos.close();
32          if (s != null)
33            s.close();
34        } catch (IOException ioe) {
35          System.out.println(ioe.getMessage());
36        }
37      } // ends finally
38    }
39    public static void main(String [] args) {
40      new NetworkSerializationClient();
41    }
42  }
```

Outline

- Network Serialization
- Program

Program

- Modify the chat client and chat server program to pass serialized objects instead of strings. The serialized object should contain the IP address, port, and String that is being sent. The output should then contain the IP and port.

```
C:>java ChatClient localhost 6789
hello
192.168.1.2:56450: how are you?
fine, and you?
192.168.1.2:56450: good, thanks

C:>java ChatClient localhost 6789
192.168.1.3:56451: hello
how are you?
192.168.1.3:56451: fine, and you?
good, thanks
```