File Access

- Reading: Savitch, Chapter 9

Objectives

- To learn binary file accessing
- To learn how to read/write object by object

Binary File Accessing

Binary file accessing involves the following steps

(1) Define a DataInputStream/DataOutputStream object for reading/writing

Example

```
DataInputStream in = new DataInputStream
    (new FileInputStream ("myBinaryFile"));

FileOutputStream outTemp = new
    FileOutputStream ("myBinaryFile");
DataOutputStream out = new
    DataOutputStream(outTemp);
```

(2) Use readChar(), readInt(), readFloat() ... to read from the binary file.

Use writeChar(), writeInt(), writeFloat(), writeChars() ... to write to the binary file.

Example

```
char c = in.readChar();
out.writeChars("What a wonderful world!");
```

(3) close the file

Example

```
class NewBorn {
    private float weight;
    private char gender;
    NewBorn(float _w, char _g)
        {weight = _w; gender = _g;}
    public float getWeight() {return weight;}
    public char getGender() {return gender;}
}
```

```
//BinaryFileTest.java
//The program writes each newborn's details
//into a binary file.
import java.io.*;

public class BinaryFileTest
{
    public static void main(String[] args) throws IOException  {
    final int N = 2;
    NewBorn nbList[ ] = new NewBorn[N];
    nbList[0] = new NewBorn(3.4F, 'F');
    nbList[1] = new NewBorn(4.2F, 'M');
```

```
FileOutputStream outTemp = new
        FileOutputStream ("myBinaryFile");
DataOutputStream out = new
        DataOutputStream(outTemp);

for (int i = 0; i < N; i++) {
        out.writeFloat(nbList[i].getWeight());
        out.writeChar(nbList[i].getGender());
}

out.close();
```

```
        DataInputStream in = new DataInputStream
                    (new FileInputStream ("myBinaryFile"));

        for (int i = 0; i < N; i++) {
              System.out.print(in.readFloat());
              System.out.println(in.readChar());
          }
    }
}
```

Object Serialization

- ObjectInputStream and ObjectOutputStream are two classes in *java.io*. They can be used to read/write objects.

Reading/Writing object by object involves the following steps

(1) The class, from which its object will be read/written, implements the interface Serializable.

Serializable is an interface defined in *java.io*.

Example

```
class NewBorn implements Serializable {
    ... ...
}
```

(2) Define

ObjectInputStream/ObjectOutputStream object for reading/writing.

Example

```
ObjectOutputStream out = new
    ObjectOutputStream(new
    FileOutputStream ("myObjectFile"));
```

(3) Use readObject()/writeObject() methods to read/write.

Example

```
NewBorn myChild = (NewBorn) in.readObject();
```

(4) close the file.

Note

When a field in an object refers to another object, the writeObject() method is invoked recursively to serialize that object (the field) as well.

Similarly, readObject() method recovers such an object recursively.

Example

```
//TriangleFile.java
//the program writes and reads Triangle objects.
import java.io.*;
class Point implements Serializable {
    double x, y;
    Point(double _x, double _y) {x = _x; y = _y;}
    public String toString() { return "(" + x + ", " + y + ") "; }
}
```

```
class Triangle implements Serializable {
    Point a, b, c;
    Triangle(Point _a, Point _b, Point _c) { a = _a; b = _b;
    c = _c;}
    public String toString() { return "<" + a.toString()+
    b.toString()+ c.toString() + ">";}
}
```

```
public class TriangleFile
{
    public static void main(String[] args)
            throws IOException, ClassNotFoundException {

    Triangle t1 = new Triangle(new Point(-1,0), new
    Point(1, 0), new Point(0, 1));

    ObjectOutputStream out = new
            ObjectOutputStream(new FileOutputStream
            ("myObjectFile"));
```

```
        out.writeObject(t1);
        out.close();

        ObjectInputStream in = new ObjectInputStream (new
        FileInputStream ("myObjectFile"));

        Triangle t2 = (Triangle) in.readObject();

        System.out.println(t2.toString());
    }
}
```