Lab Exercise #4 – Serialization

Computer Science 2334

Due by: Friday, February 24, 2012, 11:30 am

Note on Team Work and Lab Submission:

As with most other labs in this course, you are to work on this lab in a team with at least one other person. However, because there will be no paper copy submitted for this lab, you should ensure that all team members include their names on an electronic cover sheet that is submitted along with the completed Eclipse archive (zip) file.

Objectives:

1. To learn how to use serialization to write and read objects to and from files.
2. To learn how to use the `writeObject()` and `readObject()` methods of the **ObjectOutputStream** and **ObjectInputStream** classes.
3. To learn how to use the **FileOutputStream** and **FiletInputStream** classes to deal with files as streams.
4. To demonstrate this knowledge by completing a series of exercises.

Instructions:

This lab exercise requires a laptop with an Internet connection. Once you have completed the exercises in this document, your team will submit it for grading through D2L.

Make sure you read this lab description and look at all of the source code posted on the class website for this lab exercise before you begin working.

For this lab the input and output filenames should be provided as command line arguments only.

Assignment:

Serialization is an important feature of Java; one that could be used in a future project. Carefully inspect how it works in this lab and the documentation comments included in the code.

1. Download the `Lab4-eclipse.zip` project archive from the class website. Import the project into your Eclipse workspace using the slides from Lab 2. You will submit the modified project archive when you are finished.

2. **ObjectOutputStream** and **ObjectInputStream** can be used to write and read objects to and from streams. Combined with **FileOutputStream** and **FiletInputStream**, we can use these classes to write and read objects to and from binary files. Which interface must be implemented by the **Candidate** class whose objects we want to write and read? Answer this question by adding a comment in the class comment block for **Candidate.**

3. Add the interface you chose to the declaration of the **Candidate** class. The declaration should have the following form

   `public class Candidate implements` *interface*

   where *interface* is the name of the interface you determined from Step 2.

4. Note that when you add the code suggested above to **Candidate**, Eclipse will give you a warning. Resolve this warning by having Eclipse generate a serial version ID number for you.

5. Repeat steps 2 – 4 for the **Contest** class.

6. Add a method with the following signature to the **Candidate** class that writes a **Candidate** object (in other words, an entry called `candidate`) to a file, whose name is passed in as an argument to the method.

```
public static void writeCandidate (String filename, Candidate candidate)
```

The code for this method will be similar to the following:

```
FileOutputStream fileOutputStream = new FileOutputStream(filename);
ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);
objectOutputStream.writeObject(candidate);
objectOutputStream.close();
```

Here you will need to deal with possible exceptions. For this lab, it is fine to simply throw them, as Eclipse suggests. We will learn later in the course how to deal with them properly.

7. Add a method with the following signature to the **Candidate** class that reads in a **Candidate** object from the file.

```
public static Candidate readCandidate(String filename)
```

The code for this method will be similar to the following:

```
FileInputStream fileInputStream = new FileInputStream(filename);
ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);
Candidate candidate = (Candidate) objectInputStream.readObject();
objectInputStream.close();
return candidate;
```

Again you will need to deal with possible exceptions and again it is fine to simply throw them, as Eclipse suggests, for this lab.

8. Add code to `main` in the **Lab4Driver** class that uses the methods `writeCandidate()` and `readCandidate()` to write and read a **Candidate** object to and from a binary file. The code should follow the algorithm given below. Once you have written this code, test your program to ensure that it writes and reads the binary file.

   a. Create a list of three **Contest** objects called `contests`. See Lab #3 for how to do this.
   b. Create a **Candidate** object called `candidate`, that uses `contests`.
   c. Write `candidate` to a file.
   d. Set `candidate` to `null`.
   e. Print `candidate`, which should be `null`, to the console using `System.out.println()`.
   f. Read in the **Candidate** object from a file and assign it to `candidate`.
   g. Print `candidate` to the console using `System.out.println()`.

9. Add a new method to the `Lab4Driver` class that has a signature similar to that given below. This method will write an entire list of **Candidate** objects, called `candidates`, to an output file using **ObjectOutputStream**.

```
public static void writeCandidates(String filename, List<Candidate> candidates)
```

Model the body of this method on the body of the `writeCandidate()` method above noting that the method call to **ObjectOutputStream** should be similar to the following:

```
objectOutputStream.writeObject(candidates);
```

10. Add a new method to the **Lab4Driver** class that has the signature given below. This method will read a complete list of **Candidate** entries (i.e., `candidates`) from an input file using **ObjectInputStream**.

```
public static List<Candidate> readCandidates(String filename)
```

The method call to **ObjectInputStream** should be similar to the following:

```
List<Candidate> candidates = (List<Candidate>) objectInputStream.readObject();
```

11. Add code to the main method of the Lab4Driver class that uses the methods `writeCandidates()` and `readCandidates()` to write and read the list of **Candidate** entries (i.e., `candidates`) to and from a binary file. The code should follow the algorithm given below. Once you have written this code, test your program to ensure that it writes and reads the list of items.

a. Create a list of six **Candidate** objects called `candidates`. See Lab #3 for how to do this. Each of these candidates should have three (different) contest results assigned to it.
b. Write out `candidates` to a file.
c. Erase all of the elements in `candidates`.
d. Print `candidates`, which should be empty, to the console using `System.out.println()`.
e. Read in the list from the file used in step b and assign it to `candidates`.
f. Print `candidates` to the console using `System.out.println()`.

12. Ensure that there are no warnings generated for your code. **Do not suppress warnings.** Fix your code so that warnings are not necessary. (If you can't figure out how to fix your code to avoid the cast warning on reading the **List** into `candidates`, you may leave in that warning.)

13. Create an electronic cover page (plain text is best but PDF is also acceptable). Include this cover page in the `doc` directory of your Eclipse project.

14. Submit the **project archive** following the steps given in the **Submission Instructions** by **11:30 am, February 24, 2012** through D2L (`http://learn.ou.edu`). Note that there is no paper copy to turn in for this lab.