Lab Exercise # 4

Serialization

Computer Science 2334

Due by: Friday, February 27, 2009, 2:00 pm

Members: _____

_____

_____

_____

_____

_____

Learning Objectives:

- To learn how to use Serialization to write and read objects to and from files.
- 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 group will submit it for grading. All group members should legibly write their names at the top of this lab handout.

Make sure you read this handout 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.

Serialization is an important feature of Java, one that could be used in a future project. You must be able to read and write data to a file using Serialization.

For this lab exercise you will modify a project similar to Lab Exercise # 3. Carefully inspect how it works and the documentation comments included in the code.

1. Download the Eclipse 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 when you are finished. But, before you start modifying these files, first answer Question # 2.

2. **`ObjectInputStream`** and **`ObjectOutputStream`** can be used to read and write objects from and to streams. Combined with **`FileReader`** and **`FileWriter`**, we can use these classes to read and write objects from and to binary files. Which interface must be implemented by the **`ContactInfo`** class whose objects we want to read and write?

3. Add the interface you chose for the above question to the declaration of the **ContactInfo** class. For example, the declaration should read:

```
public class ContactInfo implements < insert interface here > {
```

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

```
public static void writeContactInfo( String filename, ContactInfo contact )
```

The code for this method will be similar to:

```
FileOutputStream fos     = new FileOutputStream( filename );
ObjectOutputStream oos   = new ObjectOutputStream( fos );
oos.writeObject( contact );
oos.close();
```

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

```
public static ContactInfo readContactInfo( String filename )
```

The code for this method will be similar to:

```
FileInputStream fis   = new FileInputStream( filename );
ObjectInputStream ois = new ObjectInputStream( fis );
ContactInfo   contact = ( ContactInfo ) ois.readObject();
ois.close();
return contact;
```

6. Add code to the main method of the **Lab4Driver** class that uses the methods **writeContactInfo()** and **readContactInfo()** to write and read a **ContactInfo** object to/from a binary file. The code should follow the algorithm given below. Once you have written this code, test your program to ensure it writes and reads the binary file.

a) Write out the **ContactInfo** object to a file.
b) Read in the **ContactInfo** object from a file.
c) Print the **ContactInfo** object to the console using **System.out.println()**.

7. Add a new method to the **Lab4Driver** class that has the signature given below. This method will write an entire list of **ContactInfo** objects, called an **addressBook,** to an output file using **ObjectOutputStream**.

```
public static void writeAddressBook ( String filename, List addressBook )
```

The method call to **ObjectOutputStream** should be similar to:

```
oos.writeObject( addressBook );
```

8. Add a new method to the **Lab4Driver** class that has the signature given below. This method will read a complete list of **ContactInfo** entries (i.e. an entire **addressBook)** from an input file using **ObjectInputStream**.

```
public static List readAddressBook ( String filename )
```

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

```
List addressBook = ( List ) ois.readObject();
```

9. Add code to the main method of the **Lab4Driver** class that uses the methods **writeAddressBook()** and **readAddressBook()** to write and read the list of **ContactInfo** entries (i.e. an entire **addressBook)** to/from a binary file. The code should follow the algorithm given below. Once you have written this code, test your program to ensure it writes and reads the list of items.

a) Create a list of **ContactInfo** objects (called an **addressBook**). See Lab # 3 for how to do this.
b) Write out the **addressBook** entries to a file.
c) Erase all the elements in the list.
d) Print the list, which should be empty, to the console using **System.out.println()**.
e) Read in the **addressBook** entries from the file used in step # 1.
f) Print the contents of the **addressBook** entries to the console using **System.out.println()**.

10. Submit the **project archive** following the steps given in the **Submission Instructions.**

11. Turn in this lab handout to your lab instructor. Please submit just **one copy for your group**.

12. **Do not suppress warnings**.