Lab Exercise 3 – Serialization

Computer Science 2334

Due by: Friday, 24 February 2017, 11:59 pm

This lab is an individual exercise. Students must complete this assignment on their own.

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 **FileInputStream** 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 Mimir and 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.

Assignment:

Serialization is an important feature of Java; one that could (will) be used in future projects. Carefully inspect how it works and the documentation comments included in the code.

1. Download the `Lab3-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 to D2L and the individual files to Mimir when you are finished.

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

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

```
public class Book implements interface
```

where *interface* is the name of the interface you determined from Step 2. You will also need to add the corresponding import statement for this interface.

4. Note that when you add the code suggested above to **Book**, 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 **BookList** class.

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

```
public static void writeBook(String filename, Book book)
```

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

```
FileOutputStream fileOutputStream = new FileOutputStream(filename);
ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);
objectOutputStream.writeObject(book);
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 **Book** class that reads in a **Book** object from the file.

```
public static Book readBook(String filename)
```

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

```
FileInputStream fileInputStream = new FileInputStream(filename);
ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);
Book book = (Book) objectInputStream.readObject();
objectInputStream.close();
return book;
```

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 **Lab3Driver** class that uses the methods `writeBook()` and `readBook()` to write and read a **Book** 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.

Algorithm:

a. Use `readBookFile()` to read the book data into `bookList` from the text file specified by the user .
b. Get the first book from `bookList` using `get()` and assign it to a variable called `book0`.
c. Write `book0` to a file using `writeBook()`. Hardcode the filename "`BookFile`" for this method call.
d. Set `book0` to `null`.
e. Print `book0` to the console using `System.out.println("First book: " + book0)`. At this point, `book0` should be `null`.
f. Read in the **Book** object from the file using `readBook()` and assign it to `book0`. Again, hardcode the filename "`BookFile`" for this method call.
g. Print `book0` to the console using `System.out.println("First book: " + book0)`. At this point, `book0` should have a non-`null` value.

9. Add a new method to the **BookList** class that has a signature similar to that given below. This method will write the entire **BookList** object called `bookList` to an output file using **ObjectOutputStream**.

```
public static void writeBookList(String filename, BookList bookList)
```

Model the body of this method on the body of the `writeBook()` method above. Note that there should be no loops in the body of this method. Instead it should have method call to **ObjectOutputStream** similar to the following:

```
objectOutputStream.writeObject(BookList);
```

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

```
public static BookList readBookList(String filename)
```

Model the body of this method on the body of the `readBook()` method above. Note that there should be no loops in the body of this method. Instead it should have method call to **ObjectInputStream** similar to the following:

```
BookList bookList = (BookList) objectInputStream.readObject();
```

11. After the code added to `main` previously (in Step 8), add code to **Lab3Driver** that uses the methods `writeBookList()` and `readBookList()` to write and read the list of **Book** entries (i.e., the `BookList`) 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.

Algorithm:

a. Write out `bookList` to a file using `writeBookList()`. Hardcode the filename "`BookListFile`" for this method call.

b. Set `bookList` to null.

c. Print `bookList` to the console using `System.out.println("Book list:\n" + bookList)`. At this point, `bookList` should be `null`.

d. Read in the **BookList** object from the file and assign it to `bookList`. Again, hardcode the filename "`BookListFile`" for this method call.

e. Print `bookList` to the console using `System.out.println("Book list:\n" + bookList)`. At this point, `bookList` should have a non-`null` value.

12. Ensure that there are no warnings generated for your code. **Do not suppress warnings.** Fix your code so that warnings are not necessary.

13. Upload your code to Mimir and ensure that it passes all tests. Your final upload to Mimir must come by **Friday, 24 February 2017 at 11:59 pm**.

14. Submit the **project archive** by **Friday, 24 February 2017 at 11:59 pm** through D2L (http://learn.ou.edu).