Assignment on Frameworks – RandomAccessFile and HashMap

In this project, you will develop an application that modifies an existing one. I provide the following files for you to use as a starting point:

ObjectRandomAccessFile.java – a modified version of the class for enabling RandomAccessFiles to handle objects which Jia presented in his textbook. This class is complete; you do not need to modify it, but your code should be able to use it without having problems.

Student.java – a class that represents students. This class needs some modifications.

Driver.java – a class that creates objects of the Student class, writes them to an object of ObjectRandomAccessFile, reads them back from the file, updates some Student objects and writes them back to the file. This class needs some modifications.

Modifications

Student.java

This class contains four data members of type String. The String class poses a problem for random access files. The reason for this is that String objects have variable lengths, as opposed to primitive data types like int, long and double which always have lengths of 4, 8 and 8 bytes, respectively. Variable data lengths cause a problem because all of the records in a random access file must have the same length. If String objects representing last name, first name, etc. have different lengths, their records will have different lengths, and Java will not be able to correctly compute the location of each record in a file.

Your first task is to ensure that each occurrence of a particular String object has a consistent length. Thus, each occurrence of lastName would have a consistent length, for instance 24 characters, as would each occurrence of firstName , perhaps 16 characters, majorCode 3 characters and email, 32. The best way to do is to create a method that adjusts the lengths of Strings by right padding them if they are shorter than their required length or truncating them if they are longer than it.

Driver.java

Once you modifiy Student.java to have consistent String lengths, it should work with ObjectRandomAccessFile. However, not all is not well; duplicated code clutters the driver making it excessively long and difficult to follow. You need to refactor the code to eliminate the duplication. Create methods wherever possible to accomplish this.

The Driver has another problem: in order to update an object in the file, a user must provide the updateRecord method with the record number of the object. It seems unreasonable to expect users to know record numbers. It makes more sense to have them enter a more commonly available value like a student ID. To make this possible, Driver needs a mechanism for mapping Student IDs to record numbers. Java can help with this because it provides a HashMap class that stores keys and their related values, like an index in a book. Driver can be modified to add entries to a HashMap after it instantiates a Student object. Driver can look up a studentID, the key, in the HashMap and find the corresponding recordNumber, the value. Then it can pass the recordNumber to updateRecord. This is an extremely simplified version of how databases operate.

See the RandomAccessFile example for help with this.

If you are wondering whether we could have used an ArrayList of int arrays to create the relationship between studentID and recordNumber because both are ints, the answer is “yes”, but we’ll take this as an opportunity to practice using Maps.

In addition to making these changes, you will have an opportunity to demonstrate your knowledge of relevant concepts. In a separate document, please answer the questions on the next page. Please include the questions with your answers.

1. Explain what the RandomAccessFile seek method does.

2. In ObjectRandomAccessFile , the read and write methods, which are invoked in readObject and writeObject, respectively, are not explicitly qualified by an object name. For example, they are simply used as

   read(byteArray, 0, objectSize); and

   write(byteArray, 0, objectSize);

   To what object do read and write belong?

3. What data source is being "read" in read(byteArray, 0, objectSize);?

4. What is the destination to which data is copied in read(byteArray, 0, objectSize);?

5. Why do the read and write methods invoked by readObject and writeObject in ObjectRandomAccessFile operate on byte arrays, not objects?

6. In the statement read(byteArray, 0, 16), what does the 0 represent?

7. In the statement read(byteArray, 0, 16), what does the 16 represent?

8. ObjectRandomAccessFile does not define the read and write methods. How is it able to use them, or put another way, where do they come from?

9. What does the writeObject method of ObjectOutputStream do? Say more than it writes objects.

10. What does the readObject method of ObjectInputStream do? Say more than it reads objects.

11. Why do we need to derive ObjectRandomAccessFile from RandomAccessFile? Why not just use RandomAccessFile?

12. What effect does marking some of the data members in Student as "transient" have?

13. The Serializble interface does not require classes which implement it to define any methods. So what purpose does it serve?

14. Explain what it means to serialize an object?

15. Given the statement: Map <String, Book> isbnKeyedMap = new HashMap();

    String is the class to which the _____ belongs, and Book is the class to which the _____ belongs.

16. What does a Map.Entry represent?

17. What method would you use to retrieve the key object from a record in a Map.Entry object?

18. What method would you use to retrieve the value object from a record in a Map.Entry object?

19. What does the entrySet method do?

20. What method would you use to insert a record into a Map?

21. When a program reads a file, what condition causes the EOFException to be thrown?

22. What method would you use to convert a ByteArrayOutputStream object to a byte array?

23. The loadMaps method in Driver.java does explicitly return any Map objects to the method that called it. How does the calling method receive the Map objects that loadMaps processes?

24. Write an "if" statement to test the condition empObj is an instance of the Employee class.

25. Why can't we store primitive data types – char, int, long, double, etc. – in Java collection strucatures like ArrayLists and Maps?