CURTIN UNIVERSITY (CRICOS number: 00301J)
Faculty of Engineering and Science
Department of Computing
Data Structures and Algorithms

Practical 2

Aims

- To read/write DSAMining Orders as binary data files
- To serialize OrePile

Before the Practical:

- Read this practical sheet fully before starting.

Activity 1: Binary data files

To do this activity you will need to add (at least) two methods to the ShipmentOrder class. You can call them save() and load(). At this stage, both of these methods need to prompt the user for the name of a file.

First, you are going to write a method to save a single ShipmentOrder to a binary file.

Use the lecture notes and the Java documentation to guide you. When you get to writing the Ore object, you cannot just write this to file as it is an object reference, not the data.

There are several ways to handle this problem. For example, your save() method could simply call the getters in Ore class and write the data. This is messy. Instead we will add a write method in Ore class. This method will need to import the file stream object, not the file name as the file is already open! How will you handle IOExceptions in this method?

Now you are going to write a method to read the ShipmentOrder into memory from the file that save() wrote to.

Use the lecture notes and the Java documentation to guide you. When you get to reading the Ore object data, you will have to create a new object.

Question: How would you deal with orderID when reading multiple ShipmentOrders from a file?

Activity 2: Testing

Write a test harness that creates a ShipmentOrder object, writes it to file, reads it back from the file and outputs the contents to the screen. You must test both valid and invalid scenarios!

You should only need to modify the test harness you wrote for last week's prac.

Activity 3: Serialization

In a later worksheet, you will need to implement a save and a load method for the Sheds.

To practice for this, you will create a separate program to do part of the job - reading/writing a single OrePile. This will involve a class with a main(), loadPile() and savePile() methods, and any other methods required to make your algorithm modular. Call your program OrePileIO

As you have seen from activity 1, it can get messy writing a save/load method for each class, so now you will do it the easy way - with object serialization.

Your program needs to create an OrePile object, write it to a file, then read it back from this file and output the contents to the screen.

Use the lecture notes to guide you. You will need to mark OrePile as serializable. How should you handle the Ore object this time?

Submission Deliverable:

Your modified Ore, OrePile and ShipmentOrder classes, and your OrePileIO program are due at the beginning of your next tutorial.

SUBMIT ELECTRONICALLY VIA BLACKBOARD, under the *Assessments* section.

If you finish early, use the rest of the practical to start the next worksheet, because that will be due later on.