

## **48024 Object Oriented Design**

### **Self-Study Module: File I/O in Java**

This module covers the basics of file input and output in Java.

Objectives:

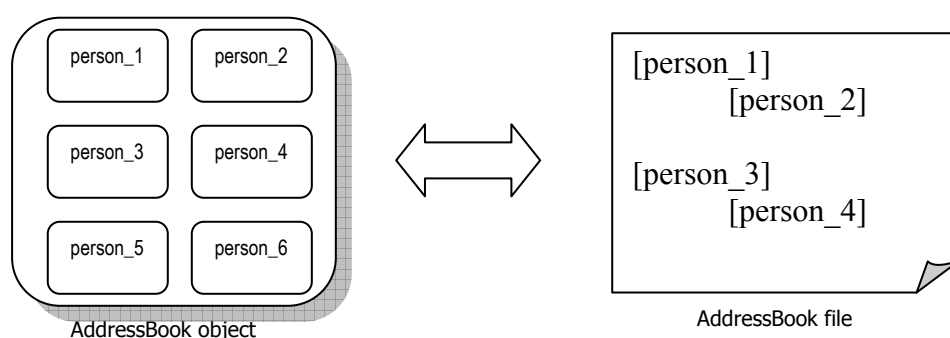
- To gain skill in the creation, reading and writing of files in Java.

References:

- Wu, Thomas. An Introduction to OOP with Java  
Chapter 11: File Input and Output
- AddressBookLab8 Project (Downloadable from UTSONline OOD Course Documents/ Self-Study Modules).

**Laboratory Exercise 1 - File Input and Output**

In this exercise you will use a project that you are very familiar with: Wu's AddressBook project which keeps a collection of Person objects. This time, we want to save all the information contained in the AddressBook object into a file, so later on the system could be able to reproduce the same AddressBook object just by reading data from the file.



A very first approach with Java has been developed by Wu in his book. If you study the `AddressBookStorage` class, you'll learn about the Java Input/Output `ObjectStream` classes which let you manage Java objects as streams of information, and so, save such objects to a binary file.

The second approach, the one you are going to implement during this laboratory, saves the information of the `AddressBook` into a text-file. To do so, follow the suggested steps:

Read section 11.3 High-level File I/O of Wu. There you can find the example of how to write data into a text file using Java `File`, `FileOutputStream` and `PrintWriter` objects. The example piece of code is:

```
//setup file and stream
File outFile = new File("c:\ABFile.data");
FileOutputStream outFileStream = new FileOutputStream(outFile);
PrintWriter outputStream = new PrintWriter(outFileStream);

//write values of primitive data types to the stream
outputStream.println("data");

//output done, so close the stream
outputStream.close();
```

Using this code as a guide, write an `exportAddressBook(String outFileName)` method which saves the data from the `AddressBook` object to a text file with the name given in `outFileName`. Remember to use a line-format you can identify once you are reading the file to construct the `AddressBook` object again, i.e. choose what

## 48024 Object Oriented Design

### Self-Study Module: File I/O in Java

text, what information you are going to write in each line of the text file, for example, you can set that every line contains the information of a Person object separated by a spacing character. Typical spacing characters include commas (,) and tabs (\t).

In that same section of Wu's book, read how data from a file is retrieved using File, FileReader and BufferedReader objects. The example code is:

```
//setup file and stream
File inFile = new File("c:\ABFile.data");
FileReader fileReader = new FileReader(inFile);
BufferedReader bufReader = new BufferedReader(fileReader);
String str;

//get info
str = bufReader.readLine();

//input done, so close the stream
bufReader.close();
```

Now it is your turn to write an `importAddressBook(String inFileName)` method that should return an `AddressBook` object formed from the data read from the text-file corresponding to `inFileName`.

#### A little further...

If you want to improve your application, try to use the Java `FileDialog` object to obtain the name of a file from the user. That is nicer than asking the user to type a string with the path of the file.

#### Review Questions

1. Did you handle any kind of Java exceptions in this program? If so, which methods throw those exceptions? Is it possible not to handle them? Explain your answer.
2. What other classes does Java offer which handle Input and Output file operations?