**OOP2 Problem Set 6A Persistence. Units 16 and 17.**

Writing Objects to File: Unit 16

1. Copy the classes Bicycle and BicycleFrame3 to your current folder, compile

them, and run BicycleFrame3. Enter details of a few Bicycles, save them to

file, close the system, then start it up again, select ‘open’ then ‘display’. You

should now see the details of your bicycles. If you improved BicycleFrame2

as part of problem sheet5B, copy your better versions of the add() and

display() methods over the versions in BicycleFrame3, and repeat the exercise.

2. What is the name of the data file being created by BicycleFrame3? Open it

using Notepad, and see whether you can make sense of the contents. Is it a

binary file or a text file? Try deleting something within it, saving and closing,

then re-run the ‘open’ option within BicycleFrame3 without saving anything.

What happens?

3. What happens if you comment out the words ‘implements Serializable’ in the

header of Bicycle? Try it and see. You will need to recompile Bicycle, then

re-run BicycleFrame3 and try to save items, before you will find out.

4. What is a stream in Java? What are the two categories of streams? What

stream objects are being used by BicycleFrame3, other than the constructor?

5. Investigate the documentation files for the class ObjectOutputStream to see

what methods are available for it. Which two of these are being used by

BicycleFrame3?

9. In the ‘save’ method of BicycleFrame3, what exception is being thrown?

Where is it being caught?

10. Rewrite the sample program NameAgeDialog1 from Unit 1, which asks the

user to enter a number: include a try-catch block which catches

NumberFormatExceptions, so that the program doesn’t crash if the user enters

a word instead of a number.

11. In the ‘open’ method of BicycleFrame3, why is the typecast necessary? (refer

to the documentation for the ObjectInputStream class and the readObject()

method).

12. In the ‘open’ method of BicycleFrame3, ‘comment-out’ the section that counts

the number of valid bicycles in the array, then re-run. What happens when

you try to open an existing file, which should have valid records in it?

**OPTIONAL**

High-Level File I/0 with DataOutputStream and DataInputStream (Unit 17)

13. Copy the program DataInAndOut.java. Right now, this program opens a

DataOutputStream to a file called data and writes out an int, a boolean, and a

double. Run this program and examine the contents of the file data. Do you

recognize the output? Why or why not?

**OPTIONAL**

14. Now uncomment the rest of the code in this file by taking out the line /\* and

also \*/. The code you uncommented, opens a DataInputStream to the file data.

It reads an int, a boolean, and a double and writes them to the console using

System.out.println(). Is the output what you expected? How does it compare to

the contents of the data file?

**OPTIONAL**

15. NameAgeDialog3 is a sample program from Unit 1 that asks you to enter

some information then display it in a message dialog): add a section at the end

that writes all the information into a binary file using a DataOutputStream.

Then write another program that opens that file, reads back the data, and

displays it in a message dialog.

Mixing Stream types: reading back data that was written as text strings

**OPTIONAL**

16. Copy the class TextInAndOut. This program writes the same data to file as is

written by DataInAndOut, but as text strings, and then reads the data back in

using a DataInputStream and prints the values to the console. Try to run this

program. What is the problem? Notice that an exception is thrown. Fix the

program so that it reads the same int, boolean, and double that were written to

the text file. Hint: Use a Scanner or a BufferedReader instead of a

DataInputStream and convert input lines to the appropriate data types. Have a

look at Ch12TestBufferedReader and Ch12TestScanner to see how to do this.

FileChooser and Filters (Unit 17)

**OPTIONAL**

17. Copy the program ChooserExercise. Modify it to use a JFileChooser

object instead of asking the user to type in the file name. The dialog window

should open on the current directory, that is, the one in which

ChooserExercise.java is located. Check to see if the user has opened a file

or cancelled. If the user has selected a file, display the file name. If the user

has cancelled, print "dialog cancelled".

18. (**Optional**) Add an OurFilter file filter to your

JFileChooser so that only files ending in .txt will be displayed. Before you

run this, put some .txt files into the current working folder (create them

6. Define the term **exception**.

7. Give *any* example you like of an exception that can occur during the execution

of a program.

8. File handling is a well-known situation where the potential for run-time

exceptions is increased. Give *any* two examples of situations that can arise

during file handling that could trigger a run-time exception.