APRIL 2005 UCC becomes British Columbia's Newest University THOMPSON RIVERS UNIVERSITY

THE UNIVERSITY COLLEGE
OF THE CARIBOO

Computing Science

COMP 113 & 152 - Computer Programming I (3,1,1)
Fall 2004

Lab 8

Due: *Beginning of your lab session during the week of November 22 to26, 2004.*
Note: There are no labs during the week of Nov. 8 to 12.

Introduction

A class definition is extremely useful in encapsulating the definition of a practical/real-world kind of object. As the definition is prepared, keep the following aspects in mind,

- the specific characteristics that each object of the class will have are the *instance variables* of the class
- the actions each object is capable of are the *methods* of the class
- aspects visible by a program declaring an object of the class should be **public**
- aspects that are used only within an object of the class should be **private**

Self-Review Questions

See the end of this lab for a set of suggested review questions.

Lab Preparation

Review, and keep handy, the **PetRecord** class definition on pg 313 (Display 5.20). Review the accompanying explanation and application program on pg 316 (Display 5.21). Use both as guides in completing the **Temperature** class.

Problem

Complete problem #7, pg 338. Please read the problem statement very carefully and complete the task exactly as outlined. If you are unclear about any aspect of the problem, ask your instructor.

Additional problem suggestions:

- there are only two instance variables: the *temperature value* and the *scale* ('C' or 'F'), with the scale secure against other character values
- have only a single accessor to return the temperature, called: **getTemperature()**
- the conversion formulas are: C = 5 * (F – 32) / 9 and F = (9 * C / 5) + 32
  with the appropriate formula used to recalculate the *temperature value* when the **setTemperatureAndScale()** and **setScale()** methods are called
- the comparison methods should utilise the nature of the conversion to overlook if two Temperature objects have different scales,
  - if the scales are different (a C compared to a F, or F compared to a C), a temporary conversion must occur in the current object so it's temperature is temporarily in the same scale as the other object,
  - the **lessThan()** method that compares if the current object's temperature is less than the other object
  - the **greaterThan()** method that compares if the current object's temp. is greater than the other object
  - the **equals()** method that compares if the current object's temperature is the same as the other object
- include a display method (called **show(), writeOutput(),** or **toString()** that returns a **String** containing a formatted display of the object's *temperature value* and *scale*. By using this method, the String can be displayed in the **main()** to either the CLI or GUI.

Conclusion and Testing the *Temperature class* (writing an application program)

You will need to make a separate (application) program to test the methods you just have written (you will also need to store all source code files in the same directory).

Your testing (application) program should do the following,

- create 3 Temperature objects with different data; use 3 different constructors, and some of the mutator methods to give the objects data
- show contents of each temperature object before and after mutator methods are used, ensuring changes are made
- call at least 2 of the accessor methods, 2 of the mutator methods, one of the comparison methods and the equals() method; capture a selection of program output to show that these methods are working correctly

Submit print-outs resulting from the execution of the application program, clearly showing the results described above, together with the source code of the **Temperature** class and your testing (application) program.

[Optional] Challenge Problem

Create your own input/output utility class, called MyIO.java, containing static methods to allow the user to input various types of data from the GUI. For example, consider such methods readInt(), readDouble(), readChar(), readString(), and other methods that can be used in your future programs to simplify the data input.

Essentially, these methods are similar to the SavitchIn class, except that input data uses JOptionPane.showInputDialog() and the appropriate Integer.parseInt(), Double.parseDouble(), or .charAt() methods on the user's input.

Self-Review Questions

Solutions to these review questions are found at the end of the relevant chapter. Make an honest attempt at answering each before checking the solution.

- page 277, questions: #4, #5
- page 277, questions: #10 *(** gotcha! **)*
- page 282, questions: #14, #16
- page 304, questions: #26, #28, #30 *(** overloading! **)*
- page 322, questions: #36, #37 *(** constructors **)*