PEER-LED TEAM LEARNING
COMPUTER SCIENCE

MEETING 6 – STUDENT VERSION
DEFINING CLASSES III

BARBARA G. RYDER AND PRADIP HARI

Exercise 1. Designing classes with static fields and static methods

a. In this exercise we will be designing a class for AddressBookEntry objects. We will be emphasizing the new concepts of *static* methods and fields and the utility of using multiple constructors for a class.

- Name some fields for an AddressBookEntry. Show the Java code that defines this class.
- Can you think of a useful static field for this class? Show the Java code that defines these fields.
- What static methods would be useful in this class?
- Can you think of real-world uses you could make of a set of AddressBookEntry objects?

b. Now think of designing a Bookmark object for your favorite browser.

- What fields are natural for a Bookmark object?
- Do all Bookmark objects require the same fields? If not, then this is a situation needing multiple constructors in a class. Define 3 alternative constructors for this class. Give the function signature and the statements in the function body.
- Draw a flow chart of one of the constructors.
- In pairs, make up 2 calls to one of your constructors: one call is legal, the other is not. Now show your constructor function signatures and your 2 calls to the other team and have them tell you which call is legal.
- Still in pairs, produce Java code for one of the constructors.
- Can you suggest possible static fields for the Bookmark class. Given your static fields, describe how you would implement this functionality in the Bookmark class (i.e., what methods would have to be changed and what are the changes to be introduced?). Given the static fields you defined, can you think of a practical use for the information they contain?

Exercise 2. Writing the code for a class

In this exercise we will be writing the code for a class called Calculator, that represents the formula entered into a calculator (which consists of two operands and an operator). We will be defining the fields and completing the methods of this class.

We have given you a program called CalcGui that acts as a GUI (Graphical User Interface), by displaying a calculator and letting the user press its buttons. The GUI uses your Calculator class to perform the actual mathematical calculations. We can use CalcGui to test our Calculator class after we write each method.

The operations we will implement in Calculator include

- Entering a single digit of either operand (when the user presses a digit button). We have to properly update the operand to account for its new rightmost digit.
- Retrieving each of the two operands (for display by the GUI).
- Entering an operator: either +, -, *, or / (when the user presses an operator button).
- Computing the answer to the formula entered (when the user presses the = button).
- Clearing out the formula (when the user presses the Clear button).

```
public class Calculator
{
   /* add data fields here */

   /*
    * Method Clear
    * Clears out entire formula
    */
   void Clear()
   {
            /* fill in this method */
   }

   /*
    * Method EnterDigitOfFirstOperand
    * Takes a digit and appends it to the end of the first operand
    */
   void EnterDigitOfFirstOperand(int digit)
   {
            /* fill in this method */
   }

   /*
    * Method GetFirstOperand
    * Retrieves the first operand of the formula being entered
    */
   int GetFirstOperand()
   {
            /* remove the line below and fill in this method */
            return 0;
   }

   /*
    * Method EnterDigitOfSecondOperand
    * Takes a digit and appends it to the end of the second operand
    */
   void EnterDigitOfSecondOperand(int digit)
   {
            /* fill in this method */
   }

   /*
    * Method GetSecondOperand
    * Retrieves the second operand of the formula being entered
    */
   int GetSecondOperand()
   {
            /* remove the line below and fill in this method */
```

```
            return 0;
    }

    /*
     * Method EnterOperator
     * Takes a mathematical operator and stores it
     */
    void EnterOperator(char op)
    {
            /* fill in this method */
    }


    /*
     * Method ComputeAnswer
     * Evaluates the formula entered
     */
    int ComputeAnswer()
    {
            /* remove the line below and fill in this method */
            return 0;
    }

}
```

Exercise 3. More exercises in logic

a. There are two lengths of rope. Each one can burn in exactly one hour. They are not necessarily of the same length or width as each other. They also are not of uniform width (may be wider in middle than on the end), thus burning half of the rope is not necessarily 1/2 hour. By burning the ropes, how do you measure exactly 45 minutes worth of time?

b. What is the smallest number of coins that you can't make a dollar with? I.e., for what N does there not exist a set of N coins adding up to a dollar? It is possible to make a dollar with 1 current U.S. coin (a Susan B. Anthony), 2 coins (2 fifty cent pieces), 3 coins (2 quarters and a fifty cent piece), etc. It is not possible to make exactly a dollar with 101 coins.

c. You are a participant on "Let's Make a Deal." Monty Hall shows you three closed doors. He tells you that two of the closed doors have a goat behind them and that one of the doors has a new car behind it. You pick one door, but before you open it, Monty opens one of the two remaining doors and shows that it hides a goat. He then offers you a chance to switch doors with the remaining closed door. Is it to your advantage to do so?

Cite this module as: Ryder, B.G., Hari, P. (2012). Peer-Led Team Learning Computer Science: Meeting 6 - Student Version; Defining Classes III. Online at http://www.pltlis.org.
Originally published in *Progressions: The Peer-Led Team Learning Project Newsletter*, Volume 9, Number 1, Fall 2007.