# PEER-LED TEAM LEARNING
# COMPUTER SCIENCE

# MEETING 5 – STUDENT VERSION
# DEFINING CLASSES II

BARBARA G. RYDER AND PRADIP HARI

**Exercise 1: Using reference variables and fields**

We will be playing a game with Money objects that consist of dollars and cents parts. Each student will represent a variable (you can use your first name as the variable name.) The peer leader will bring cards each marked with one of the following kinds of statements:

a. <var> = new Money(<dollar_amt>,<cents_amt>)
b. <var> = null
c. <var1> = <var2> (for this the student gets to choose which other student's variable to use on the righthandside
d. Increment the dollars in your object by 100 ☺
e. Decrement the dollars in your object by 100 ☹
f. Add 50 cents to your object.
g. Swap the amounts in your object's dollars and cents.
h. Etc.

Each student takes turns picking a card and simulating its actions. S/he also has to draw the corresponding Java code on the growing code list on the board.

The peer leader will draw each new object and each of the variables on the board. As the cards are drawn from the deck, each player changes the drawing on the board to reflect the meaning of the card. Notice that some actions may be errors, such as trying to increment a null object; if you draw this sort of action, explain why it doesn't make sense and therefore is an error. What will happen if a running Java program encounters this action?

As an additional activity, each student will add their own cards to the game and reply it. Think about ways to make the operations allow more interesting actions on the Money objects.

**Exercise 2: Using reference variables and fields (with sharing)**
For this exercise we will consider how to simulate a student's class schedule using object-oriented programming. Each class will have an associated time and a roster of at most 5 students. Each student will take 3 classes (we are only considering the math and CS parts of her schedule). Assume a set of Student and Course objects has already been created; we will refer to this using common names as reference variables (e.g., sally will be referring to a particular Student object, already created).

As a group, define the objects in each class and the fields necessary to model this real-life Situation.

Again we will have cards each of which contains a command such as:

a. Add class CS111
b. Drop a class

In turn, you each will pick a card and then perform the action on it. Hint: the commands on some cards may involve operations on more than one class.

First, your peer leader will select someone to draw a card and then draw the action on the board. After doing this for a few cards, we can make sure everyone understands how the simulation works.

Then, you will take turns each drawing a card and performing its action. During the exercise, the peer leader will draw all of the Roster objects on the board, with their corresponding students. Each student will be able to copy the board drawing and add to it for her own Student object's classes, to see how these objects become interrelated.

At random times, the peer leader will announce changes to the class times of certain classes. When this happens, you need to check if your Student object will be affected by this time change.

After a while, the peer leader will have 2 students who are taking the same class, draw the Course and Roster objects for their shared class on the board. Next, the peer leader will choose 2 students who do not share a class and have them each draw one of their unshared Rosters on the board.

**Exercise 3. 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 4: A logic game for fun**

(UWisc7) Given a 3x3 chess board we can play a logic game. Assume the knights (B for blue knight, R for red knight) are placed on the chessboard thusly:

```
B  --  B
--  --  --
R  --  R
```

Can you move the knights so that the B knights and R knights are swapped? Recall that knights move in an L-shaped pattern on the board and that 2 knights cannot occupy the same space at any time.

If you solve the puzzle, then what is the minimal number of moves to do so?

**Cite this module as**: Ryder, B.G., Hari, P. (2012). Peer-Led Team Learning Computer Science: Meeting 5 - Student Version; Defining Classes II. Online at http://www.pltlis.org.
Originally published in *Progressions: The Peer-Led Team Learning Project Newsletter*, Volume 9, Number 1, Fall 2007.