Activity #20: Introduction to Classes
Recorder's Report

Manager: Reader:

Recorder: Driver:

Date: Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with ) below.

a) Model 1, Question #2

b) Model 2, Question #6

c) Model 3, Question #12 (a)

Activity #20: Introduction to Classes

In this activity, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to classes in C++.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain the benefits of abstraction in programming
- Explain the syntax for defining classes in C++
- Explain the benefits of encapsulation in programming
- Explain the use of mutator and accessor methods

Process Skill Goals

During the activity, students should make progress toward:

- Construct a simple class with data fields and methods

Model 1 C++ Code Snippets

```
/* Data for multiple rectangles */
const int MAX_RECTANGLES = 100;
int rectX[MAX_RECTANGLES];
int rectY[MAX_RECTANGLES];
int rectWidth[MAX_RECTANGLES];
int rectHeight[MAX_RECTANGLES];

```

```
/* Functions to use with rectangles */
int getArea(int length, int width);
int getPerimeter(int length, int width);
void move(int &x, int &y, int dx, int dy);
void draw(int x, int y, int wd, int ht);

```

Refer to Model 1 above as your group develops consensus answers to the questions below.

Questions (10 min) Start time:

1. The C++ code snippets above come from a program designed to define and manipulate rectangles. Answer the following questions related to this code.

a) Give C++ code to define a rectangle at the point $(1, 1)$ with a width of 3 and a height of 2 stored at index 0 in the arrays above.

b) Write an appropriate function call for each task described below.

(a) Move your rectangle up 2 and over 1.

(b) Compute the area of your rectangle.

(c) Draw your rectangle

c) What would you have to change in order to make the function calls above reference another rectangle stored at index 1?

2. A complete version of the program can be found in the file `activity20a.cpp`. Add  to this code to complete the following tasks.

- Create a second rectangle at point $(2, 5)$ with width 10 and height 8.
- Move your second rectangle so that its point is at $(0, 0)$.
- Determine which of the two rectangles has a larger perimeter (using C++ code).
- Draw your second rectangle.

Describe the most frustrating part of completing the tasks above.

3. In programming, the term *abstraction* is used to describe representing complex things simply. For example, we all know how to turn on a light switch even if we don't understand how it actually works. Decide if each of the following statements regarding this model is true or false.

a) We can define new rectangles without knowing the details of how rectangles are stored.

b) We can compute the area or perimeter of a rectangle without understanding the details.

c) We can draw or move a rectangle without understanding the details.

d) If we needed to expand our program to define and manipulate circles, we could do so without renaming functions or confusing circle and rectangle code.

Model 2 A Different Approach

```
class Rectangle {
  public:
    int x;          // (x,y) coords
    int y;
    int width;      // width (dx)
    int height;     // height (dy)
    int getArea();
    int getPerimeter();
    void move(int dx, int dy);
    void draw();
};

```

```
/* Data for multiple rectangles */
const int MAX_RECTANGLES = 100;
Rectangle myRects[MAX_RECTANGLES];

/* create rectangle at point (1,1) */
myRects[0].x = 1;
myRects[0].y = 1;
myRects[0].width = 2;
myRects[0].height = 3;

/* move this rectangle up 2 and over 1 */
myRects[0].move(1,2);

```

Refer to Model 2 above as your group develops consensus answers to the questions below.

Questions (10 min) Start time:

4. The code for this model can be found in `activity01b.cpp`. Use it to help you complete the following table, filling in the equivalent code in each model.

Model 1	Model 2

	`// define a new rectangle`
	`myRects[0].x = 1;`
	`myRects[0].y = 1;`
	`myRects[0].width = 2;`
	`myRects[0].height = 3;`

`// draw the 3rd rectangle`
`draw(rectX[2],rectY[2],`
`  rectWidth[2],rectHeight[2]);`	

	`// compare rectangle areas`
	`if (`
	`  myRect[0].getArea() > myRect[1].getArea()`
	`) {`
	`  cout << "First rectangle has more area";`
	`}`

`/* error moving x from rectangle 0`
`  and y form rectangle 1 */`
`move( rectX[0], rectY[1], 3, 5);`	

5. Now decide if each of the following statements is true or false for model 2.

a) We can define new rectangles without knowing the details of how rectangles are stored.

b) We can compute the area or perimeter of a rectangle without understanding the details.

c) We can draw or move a rectangle without understanding the details.

d) If we needed to expand our program to define and manipulate circles, we could do so without renaming functions or confusing circle and rectangle code.

6. Describe at least open advantage of the approach in model 2 over that seen in model 1.

7. Suppose we wish to ensure that other programmers never set rectangle widths and heights to be negative. Can this be done in either model? Explain.

Model 3 Revising Our Approach

```
1  class Rectangle {
2    public:
3      bool init(int xVal, int yVal, int wVal, int hVal);
4      int getArea();
5      int getPerimeter();
6      void move(int dx, int dy);
7      void draw();
8    private:
9      int x;        // (x,y) coords of bottom left corner
10     int y;
11     int width;    // width (dx) and height (dy)
12     int height;
13 };
14
```

Refer to Model 3 above as your group develops consensus answers to the questions below.

Questions (30 min) Start time:

8. Formulate a hypothesis about what each of the lines from the model mentioned below is for.

a) Line 2:

b) Line 8:

c) Line 3:

9. The full code for this model can be found in `activity01c.cpp`. Make the following changes to this code (resetting after each change) and describe what happens.

a) On line 24, add the code `myRects[0].x = 5;` and compile.

b) Change line 26 to use the function call `myRects[0].init(1,1,-5,3)`

10. We've previously seen that global variables should be avoided. Similarly, when defininig classes, we should protect the variables in the class by making them private. This practice is called *encapsulation*. Give at least one reason why encapsulation is a desirable feature in a program.

11. Functions that belong to a class are called *member functions*. How are member functions defined?

12. We've previously asked about expanding our program to define and manipulate circles. Assuming that a circle is defined by a center (x, y) point and a non-negative radius, complete the following.

a) Give a definition of a class for circles. Try to use abstraction and encapsulation as much as possible.

b) Add code to the `main` program to define a circle at the point $(-2, 1)$ with radius 3.

c) Add member functions to compute the area (πr^2) and circumference ($2\pi r$) of a circle.

d) Write code to compare the areas of the circle you defined and the rectangle defined in the program and indicate which is bigger.