Activity 11: Designing Classes

Previously we explored how classes define attributes and methods. Static variables and methods apply to the whole class, whereas non-static variables and methods apply to specific objects.

Content Learning Objectives

After completing this activity, students should be able to:

- Discuss benefits of POGIL for student learning.
- Explain the purpose of constructor, accessor, and mutator methods.
- Implement the equals and toString methods for a given class design.
- Design a new class (UML diagram) based on a general description.

Process Skill Goals

During the activity, students should make progress toward:

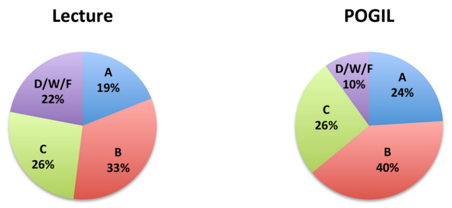
- Identifying attributes and data types that model a real-world object. (Problem Solving)

Model 1 POGIL Research

Process-Oriented Guided Inquiry Learning (see pogil.org) is a student-centered, group-learning instructional strategy and philosophy developed through research on how students learn best. The following two figures are from peer-reviewed articles published in education journals.

Grade Distributions in General Chemistry

Data (n = 905) from small (~24 students) sections of three instructors using lecture approach (1990-94) prior to implementation of POGIL pedagogy (1994-98).

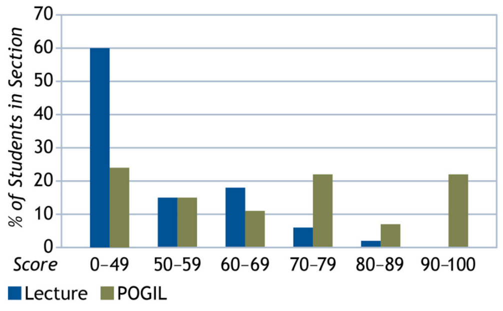


Farrell, J.J., Moog, R.S., & Spencer, J.N. (1999). A Guided Inquiry Chemistry Course. *Journal of Chemical Education*, *76*, 570–574.

Performance on Organic Chemistry 2 Unannounced First Day Pre-Quiz

All students passed Organic Chemistry 1 at this institution during the previous semester

All sections of Organic Chemistry 1 had more than 150 students.



Ruder, S.M., & Hunnicutt, S.S. (2008). POGIL in Chemistry Courses at a Large Urban University: A Case Study. In R.S. Moog, & J.N. Spencer (Eds.), *Process-Oriented Guided Inquiry Learning: ACS Symposium Series 994* (pp. 133–147). Washington, D.C.: American Chemical Society.

Questions (10 min)

Start time: ________

1. How large were the classes at each of the universities shown above?

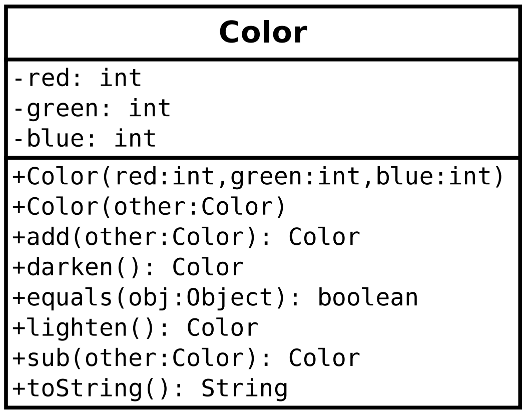
2. What are the measures of performance shown in each of the figures?

3. What does the figure on the left suggest about POGIL's impact on student success?

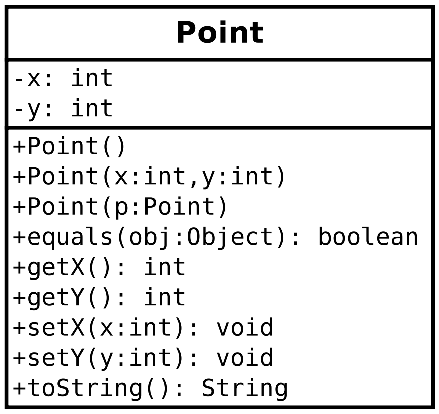
4. What does the figure on the right suggest about students' retention of knowledge?

Model 2 Common Methods

Classes are often used to represent abstract data types, such as `Color` or `Point`. They are also used to represent objects in the real world, such as `CreditCard` (see Model 3) or `Person`.



```
Color
-red: int
-green: int
-blue: int
+Color(red:int,green:int,blue:int)
+Color(other:Color)
+add(other:Color): Color
+darken(): Color
+equals(obj:Object): boolean
+lighten(): Color
+sub(other:Color): Color
+toString(): String
```



```
Point
-x: int
-y: int
+Point()
+Point(x:int,y:int)
+Point(p:Point)
+equals(obj:Object): boolean
+getX(): int
+getY(): int
+setX(x:int): void
+setY(y:int): void
+toString(): String
```

Classes generally include the following kinds of methods (in addition to others):

- **constructor** methods that initialize new objects
- **accessor** methods (getters) that return attributes
- **mutator** methods (setters) that modify attributes
- **object** methods such as `equals` and `toString`

Questions (20 min) Start time: ________

5. Identify the constructors for the `Color` class. What is the difference between them? What arguments do they take?

6. What kind of constructor does the `Point` class have that the `Color` class does not? What is the purpose of such a constructor?

7. Identify an accessor method in the `Point` class.

   a) Which instance variable does it get?

   b) What arguments does the method take?

   c) What does the method return?

8. Identify a mutator method in the `Point` class.

a) Which instance variable does it set?

b) What arguments does the method take?

c) What does the method return?

9. The `Color` class does not have accessors or mutators, but it provides methods that return lighter or darker `Color` objects. Why do you think the class was designed this way?

For the following questions, consider the source code of `equals` and `toString` from the `Point` class.

```
public boolean equals(Object obj) {
    if (obj instanceof Point) {
        Point p = (Point) obj;
        return this.x == p.x && this.y == p.y;
    }
    return false;
}

public String toString() {
    return String.format("(%d, %d)", this.x, this.y);
}
```

---

10. What is the value of each expression below? (Don't just guess; read the source code above.)

```
Point p1 = new Point();  Point p2 = new Point(0, 0);  Point p3 = new Point(3, 3);
```

a) `p1 == p1`

b) `p1.equals(p1)`

c) `p1 == p2`

d) `p1.equals(p2)`

e) `p1.equals(p3)`

f) `p2.toString()`

g) `p2.equals("(3, 3)")`

h) `p3.equals("(3, 3)")`

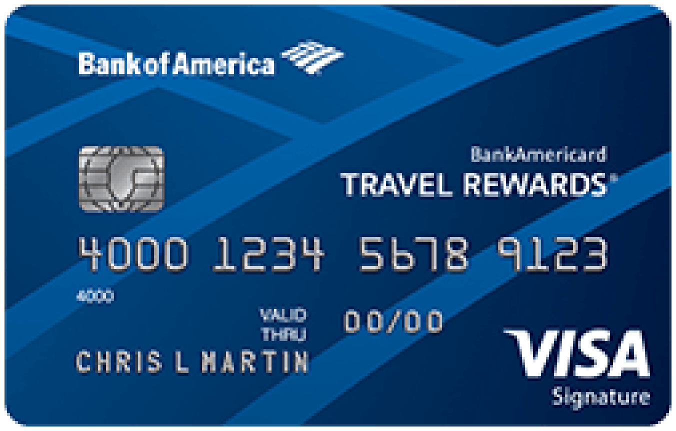
11. What is the purpose of the `equals` and `toString` methods?

12. What is the purpose of the `if`-statement in the `equals` method?

13. How could you modify the `equals` method to cause both #10d and #10h to return `true`?

Model 3 Credit Card

In this section, you will design a new class that represents an individual's credit card.



Questions (15 min) **Start time:** ________

14. List two or more attributes that would be necessary for the `CreditCard` class. For each attribute, indicate what data type would be most appropriate.

15. When constructing (or updating) a `CreditCard` object, what values would you need to check? What are the valid ranges of values for each attribute?

16. List two accessor methods would be appropriate for the `CreditCard` class. Include arguments and return values, using the same format as a UML diagram.

17. List two mutator methods would be appropriate for the `CreditCard` class. Include arguments and return values, using the same format as a UML diagram.

18. Describe how you would implement the `equals` and `toString` methods of the `CreditCard` class (i.e., what you would define them to return based on which attributes).