

Content Teaching Academy – Computer Science

POGIL Activity: Object Oriented Programming (50m)

Why?

Object Oriented Programming is widely used programming paradigm that is an essential component of modern computer science.

Learning Objectives

- Understand and be able to explain the concepts of classes and objects in OOP.
- Interpret and write code for simple classes in Java.

Resources

- Reading (attached)

Vocabulary

class, object (syn: class instance), attribute (syn: data member), method (syn: procedure member)

Plan

1. Review the attached reading, individually. (10)
2. Assign key roles to members of your team (2).
3. Answer the Key Questions individually, and then evaluate your answers as a team. (6)
4. Do the Exercise as a team, and then check your answers with the instructor. (8)
5. Do the Problems as a team. (14)
6. Present part of your solution to the whole class. (8)
7. Complete the assessment as a team and then turn in your deliverables. (2)

Key Questions

1. Which statement is correct? *A class is a set of objects*, or *An object is a set of classes*
2. Why does each object have unique attribute values, while all objects within a class share the same methods?

Exercise

Examine the following calculator class. Then determine what will be output by each of the following code examples.

1.

```
Calculator ca = new Calculator(); // create a Calculator instance
ca.enter(1.5);
ca.add();
ca.enter(2.0);
ca.add();
ca.display();
```

2. `Calculator cb = new Calculator(); // create another Calculator instance`
`cb.enter(5.0);`
`cb.enter(5.5);`
`cb.add();`
`cb.add();`
`cb.store();`
`cb.enter(6.0);`
`cb.add();`
`cb.recall();`
`cb.subtract();`
`cb.enter(10.0);`
`cb.add();`
`cb.display();`

----- Check with the instructor before proceeding. -----

Problems (Deliverable)

1. Add these methods to the Calculator:
 1. `clear()`: clear the accumulator
 2. `clearEntry()`: clear the entry value
 3. `clearMemory()`: clear the memory
2. Add methods for performing multiplication and division to the Calculator class.

In the division method, include a test for division by zero that will display an error message if entry is zero.

3. Add a square root method to the Calculator class.

Include an error test that will display an error message if the entry value is negative

Assessment (Deliverable)

1. What part of this activity was most difficult for you?
2. How could your team improve their performance on future activities?

```

/*
 * Calculator class
 *
 * @author R.Grove
 * @version 19Jun2013
 */
public class Calculator {

    // attributes
    private double accumulator = 0.0;
    private double memory = 0.0;
    private double entry = 0.0;

    // methods
    public void enter(double value) {
        entry = value;
    }
    public void add() {
        accumulator = accumulator + entry;
    }
    public void subtract() {
        accumulator = accumulator - entry;
    }
    public void store() {
        memory = accumulator;
    }
    public void recall() {
        entry = memory;
    }
    public void display() {
        System.out.println(accumulator);
    }
}

```

Concepts of Object Oriented Programming:

The essence of Object Oriented Programming is the creation of software models (abstractions) of application-domain entities. For example, in a Library management application, we might create models of the entities Book, Member, Check-Out, Book-Hold. Note that the entities include things, people, and events (transactions). Each entity would be defined (modeled) In software as a **class**. So, our Library application would have four classes. (Technically, classes are also called *abstract data types*.)

When it becomes necessary to create records of individual things, people, or events during execution of the application, each of those individuals is represented by an **object** of the appropriate class (objects are also called *class instances*). For example, there would be many Book objects, Patron objects, Check-Out objects, etc. So, there is only one class for each type of entity, but many objects of that class.

The data items that are unique to each model are called its **attributes**. For example, Book might have attributes Title, Publication-Date, and Call-Number, while Patron might have attributes Name, Member-Id, and Status. Each object has its own unique copies of the attributes, called its attribute values.

The procedures that operate on attributes of a class are called its **methods**. A class can define as many methods as are necessary to perform its role in the application. Methods can be responsible for storing data, retrieving data, performing calculations, or performing input/output. Since methods are common to the class, there is only one copy of the methods that all objects share.

Example:

Patron Class	Patron Object 1	Patron Object 2	Patron Object 3
Attributes: <ul style="list-style-type: none">• name• member-Id• status Methods: <ul style="list-style-type: none">• getName• getStatus• setName(newName)• setStatus(newStatus)	<ul style="list-style-type: none">• Sue Jones• 123456• Active	<ul style="list-style-type: none">• James Wyatt• 444444• Active	<ul style="list-style-type: none">• Gail Smith• 565656• Inactive

POGIL Team Roles:

Assigning individual roles to team members during classroom activities helps to keep all team members engaged, and gives them each a share of the responsibility. Typical team roles include:

- Facilitator: Guides team through the assignment by following the plan
- Recorder: Records all products of teamwork; submits deliverables
- Observer: Watches what the team is doing; Keeps the team on track to finish on time; Notes problems and
- Checker: Reviews all work for quality, completeness, and correctness
- Reporter: Reports teamwork results to the entire group when necessary

Team roles can be modified as appropriate, or combined to fit the number of team members. Team roles should be identified at the start of each activity.