

Lab 4, CSC 203 – Inheritance

Due: 11/03 @10:00PM

In this lab, you will explore some aspects of inheritance in Java. In the first part, you will refactor a provided solution, which implements a simple calculator to gain experience with inheritance. In the second part, you will explore the notion of "equality" in Java.

Objectives

- To explore the use of inheritance
- Abstract class and method
- To practice recognizing when classes share data or similar methods and how to use inheritance to make these relationships explicit

Given Files

Retrieve the files provided for this lab from Canvas

Refactoring using Inheritance

In the calculator directory is a solution for a simple calculator (with support for variables). For this part of the lab, you will modify this solution to refactor common functionality out of some of the classes into a common class. The main method is in Calculator class. Run the lab and see the output.

Many of the expressions are what we would typically call binary expressions. In particular, AddExpression, DivideExpression, MultiplyExpression, and SubtractExpression each have very similar behavior. To see this clearly, compare AddExpression.java and SubtractExpression.java. In fact, let's compare them using diff to see just how similar they are.

```
diff AddExpression.java SubtractExpression.java
```

To refactor this common behavior, follow the steps below.

- Create a new class named BinaryExpression. This class should inherit from Expression.
- Each of the classes listed above must be modified to inherit from BinaryExpression.
- Now lift the common data out of these subclasses into the common parent class BinaryExpression. Make this data private.
- With the common data in the parent, the common functionality will also be lifted. Move the evaluate and toString methods into the BinaryExpression class.
- Add an additional argument to the BinaryExpression constructor. Modify the subclasses to pass a String, representing the operator, to the parent class. Update BinaryExpression's toString to use this operator string.
- Modify the code in evaluate in the BinaryExpression class to evaluate each operand (i.e., lft and rht) and to pass the results to a new, protected method named _applyOperator. This new method will not be defined in BinaryExpression, but must be declared (**what sort of method is this?**) in each subclass, implement this new method as is appropriate for the subclass's operator.

Equality

Examine the provided files in the equality directory.

For this part, you must implement equals and hashCode for the appropriate objects, in order that all test cases to pass.

For the CourseSection class, you must implement equals and hashCode in long form (i.e., you must program all computations explicitly. Based on the attributed type use similar functionality given in the slides). Be sure to handle null values (for object and every attribute).

For the Student class, you must implement equals and hashCode using the appropriate methods of java.util.Objects.

Submission

Submit java files in Canvas and demo before it is due.

Rubric:

	<u>Points</u>
BinaryExpression	10
_applyOperator	10
Add	5
Subtract	5
Multiply	5
Divide	5
Hash Student (using Objects hash)	10
Hash Code (long form)	10
TestCases	10
Our TestCases	30