Programming Project 2

Due Tuesday, March 19 at 11:59pm

The purpose of this project is give you practice using a Object heirarchy in designing a program. You should not use any loops in your code. The key point of this project is to create a good object-oriented design. In particular, you will be graded on how well you use method overriding, `this`, `super`, and `instanceof`.

Programming Style (20% of the project grade)

Here are some guidelines for the project.

- You should place a comment at the top of the file that contains your name and a short description of the purpose of the class.
- You should place a *short* comment before (directly above) each method describing the method. The comment should be one sentence and describe *what* the method does, not *how* it does it. Do not simply copy the descriptions below for your comments.
- You should place a short comment directly above each field (if you have any) indicating the purpose of the variable.
- *The comments at the top of the class, above each method, and above each field are to be in JavaDoc format.*
- Any other complicated code such as code the has lots of if statements or variables should contain *short* comments to help the reader. The comments can either be above the code fragment or to the right, aligned in a column.
- Remember to use good style: everything should be indented nicely, variables should have good names, there should be a blank line between each method.

Here are some new guidelines for this project:

- You may add additional methods and fields to the classes below beyond the ones given.
- You should not have an object store the same value in multiple fields. (For example, if the parent class has a field that stores a value, the class that extends it should not have a field that stores the identical value.)

Don't overdo the inheritance. Your class hierarchy should be relatively simple and intuitive.

To submit your project, click on the *Project 2* link, and you should find a *Submit* link. Attach your Java files and your testing report. Remember to attach the files that end in `.java`, not `.class` or `.java~`.

Programming Correctness (60% of the project grade)

You will program a function calculator. Each function can be a function of one variable (ex: *3x^2 - 6*) or zero variables (ex: *57^4 - 10*). The variable will always be "x". You can use the calculator to give the value of the function at any point or to compute the derivative of the function.

Every function needs to have the following methods:

1. `double value(double x)`: gives the value of the function where `x` is the input to the function. Note: this needs to work correctly even if the function does not have any variables. For example: the result of `value(10)` on the function *x^2-3x* is 70.0.
2. `double value()`: gives the value of a function that does not contain a variable. For variables, this method should throw a `UnsupportedOperationException`. For example, the result of `value()` on the function *45 + 6 - 10* is 41.0.
3. `Function derivative()`: returns the function that is the derivative of this function.

The following are the different types (classes) of functions that you need to implement:

1. `Variable`: represents the variable *x*. The constructor should take no input. The `toString` method should just give *"x"* and the `equals` method should return `true` (if this object is compared to another `Variable`).
2. `Number`: represents a number as a `double`. The constructor should take a single `double` value. The `toString` method should give a `String` representation of the number, and the `equals` should compare the number values.
3. `BinaryOp`: represents a binary operator (+, -, *, /) and two function operands. The constructor should take three values: a `char` that represents the operator and the left and right operands. The `BinaryOp` class should have the getter methods `getOperator`, `getLeftOperand`. and `getRightOperand`. The `equals` method should return `true` if both `BinaryOp` instances have the same operator and both operands are equal. The `toString` representation should be *`left-operand`* `op` *`right-operand`* where `op` is one of +, -, *, /. (Note the single space between each operand and the operator.) The left operand should be placed inside parentheses if it is a `BinaryOp`. The right-operand should be places in parentheses if it is a `BinaryOp` that has a different operator.
4. `Polynomial`: represents a function raised to a power. The constructor should take two values: a function that is the operand and a `double` that is the power. The `Polynomial` class should have a `getPower` and `getOperand` methods. For the `toString` method, the string should use the `^` character. For example, `x^5`. If the operand is a `BinaryOp`, the string representation should place it inside parentheses. . Two polynomials are equal if their powers and their operands are equal.
5. `Log`: Represents the natural logarithm function. The constructor should take a single value, the function that is the operand of the logarithm function, and the class should have a `getOperand` method. The string representation should be *"Log[`operand`]"*. Two `Log` functions are equal if their parameters are equal.
6. `Exp`: Represents the natural exponential function. The constructor should take a single value, the function that is the operand of the exponential function, and the class should have a `getOperand` method. The string representation should be *"Exp[`operand`]"*. Two `Exp` functions are equal if their parameters are equal.
7. `Sin`: Represents the sine function. The constructor should take a single value, the function that is the operand of the sine function, and the class should have a `getOperand` method. The string representation should be *"Sin[`operand`]"*. Two `Sin` functions are equal if their parameters are equal.
8. `Cos`: Represents the cosine function. The constructor should take a single value, the function that is the operand of the cosine function, and the class should have a `getOperand` method. The string representation should be *"Cos[`operand`]"*. Two `Cos` functions are equal if their parameters are equal.

Hint: This project is not as hard as it may first seem. Design a good hierarchy for the above classes. Try adding each class, in the order given, into your hierarchy. If you have a good hierarchy, you will not need to rewrite any classes that you have completed when you add a new class to the hierarchy.

Testing Report (20% of the project grade)

As before, your report will be graded on the thoroughness of your testing, not the correctness of the results. **Important:** Your tests must match your code. That is, you must give the true outputs of your program. If your program gives an incorrect result or an error, you should try to fix the problem and re-run the tests. If you are unable to fix the problem before the homework deadline, submit the tests with the incorrect results and/or error messages. You should also include a statement highlighting the error. If you do not submit a test report, you will get a 0 on this section, but if you are found to have faked your results, you will get a negative score. In other words, you will be docked more points than this section is worth.