COMP 3522 Lab #4: Inheritance and polymorphism

Christopher Thompson, Jeffrey Yim
cthompson98@bcit.ca, jyim3@bcit.ca

Due Oct 11 2019, 11:59pm

Welcome!

Welcome back! For your fourth lab, you will implement a simple inheritance hierarchy in C++ that uses polymorphism via virtual functions.

1 Set up your lab

Start by creating a new project:

1. Create a new project in CLion. Let's call it Lab4. Choose C++ Executable for the project type and ensure you select C++14 as the Language Standard.
2. Examine the project files that are created. Note there is a file called main.cpp that contains a main method stub. Remember that CLion uses cmake, which is a build tool similar to ant. cmake uses a file called CMakeLists.txt. **Did you remember to set the correct compiler flags?**
3. Add this project to version control and GitHub. From the VCS menu in CLion select Import into Version Control | Create Git Repository... to add the project to a repo.
4. Add the project to GitHub by returning to the VCS menu and selecting Import into Version Control | Share Project on GitHub. Call it Lab4, make sure it's private, and add a first commit comment. It's fine to add all the files for your first commit.
5. Visit GitHub and ensure your repository appears. Add me as a collaborator. In GitHub, I am known as jeffbcit. You'll recognize my avatar.

2 Implement a simple inheritance hierarchy

Remember to create a separate source (.cpp) and header (.hpp) file for each class, plus an additional source (.cpp) file that contains the main method.

1. This week, you must put the main method inside a source file called inheritance.cpp.
2. Declare and define an animal base class:
   (a) animal stores an age (int), a unique long ID (I suggest you use a static variable similar to the included staticVariableExample.zip project), a boolean that is true if it is alive, and a location (a pair of double).
   (b) animal requires a default constructor and a 3 parameter constructor. Both constructors should set the unique ID automatically. They should also set the alive boolean to true. The three-parameter constructor should accept an int for the age and two doubles for the set of coordinates. The default should set these three values to 0.
   (c) animal requires a virtual move method which accepts two doubles that represent two coordinates, and 'moves' the animal to the set of coordinates.

(d) animal requires a copy constructor and a virtual destructor. The destructor should be virtual, and it needs an empty implementation.

(e) animal requires a virtual sleep method and a virtual eat method. Both methods return void. Both methods should print an appropriate message to cout.

(f) animal requires a setAlive function which accepts a boolean. This is a void function that changes the alive data member to the value of the boolean that is passed in.

(g) Overload the insertion operator for the animal.

3. Declare and define a bird class that is derived from animal:

   (a) bird stores an extra value (a double) that represents its height coordinate. Land-lubbers have two coordinates. Something that flies stores 3. How you do this is your decision. Should ALL animals store 3 coordinates? If so, how can you make the height default to zero for other animals?

   (b) bird requires a default constructor and a 4 parameter constructor. The four-parameter constructor should accept an int for the age and three doubles for the set of coordinates. The default should set these four values to 0.

   (c) bird requires a move method which accepts three doubles that represent three coordinates, and 'moves' the bird to the set of coordinates. How will you do this? Consider: if a bird has a 3-param move function, but the animal base class version accepts 2 parameters, we're not really overriding the original virtual 2-param version, and we won't be able to ask birds to move if they are being referenced by an animal pointer. Do you need to modify the animal class, in view of this information?

   (d) bird requires a copy constructor and a destructor.

   (e) bird must override the sleep method and the eat method. Both methods return void. Both methods should print an appropriate message to cout.

   (f) Overload the insertion operator for the bird. Can you call the insertion operator for the base class from the derived class' insertion operator? Investigate!

4. Declare and define a canine class that is derived from animal:

   (a) canine requires a default constructor and a 3 parameter constructor. The three-parameter constructor should accept an int for the age and two doubles for the set of coordinates. The default should set these three values to 0.

   (b) canine requires a move method which accepts two doubles that represent two coordinates, and 'moves' the canine to the set of coordinates.

   (c) canine requires a copy constructor and a destructor.

   (d) canine must override the sleep method and the eat method. Both methods return void. Both methods should print an appropriate message to cout.

   (e) canine requires a hunt method. The hunt method returns void. This method takes in an animal pointer as a parameter. The canine will set the other animal's alive boolean to false if both animals are in the same position. Two animals are in the same position if the x, y, and z values are within 1 of each other. This method should print an appropriate message to cout depending on if the hunt was successful or not.

   (f) Overload the insertion operator for the canine. Can you call the insertion operator for the base class from the derived class' insertion operator? Investigate!

5. Ensure all member variables are private or protected. Do not use global or public members to store instance data.

6. Ensure all constructors and destructors write one line of output to cout describing the version that is being called.

7. Ensure you do not duplicate member variables, i.e., if an age variable exists in the base class, do we need to declare one in a derived class? (Hint: the answer is no!)

8. Ensure you use the virtual keyword so we can employ polymorphism.
9. You may need to create some (inline?) accessors and mutators.
10. When you override a member function use the override keyword in the derived class' version of the function header.
11. Ensure you are not duplicating code.
12. In the main method, create three animal pointers. For the first, dynamically allocate a new animal. For the second, dynamically allocate a new bird. For the third, dynamically allocate a canine. Demonstrate the output from each of the constructors.
13. Demonstrate how the move method works. Demonstrate how the sleep and eat methods work. Can you demonstrate how the hunt method words with your animal pointer that points to a canine? Why not? How can you overcome this (Hint: casting).
14. After you submit your Lab to GitHub, send me a private message on Slack telling me you're finished, along with your student number, gitName, and collaboration url. ie: "Jeff I uploaded my work to gitHub. My student number is A00XXXXXX and my gitName is YYYYY, my git collaboration url is: ZZZZZZZZZZZZZZZ"
15. That's it. Invite me as a collaborator if you haven't yet, and let's mark it!

3 Grading

This lab will be marked out of 10. For full marks this week, you must:

1. (2 points) Commit and push to GitHub after each non-trivial change to your code
2. (3 points) Successfully implement the requirements exactly as described in this document
3. (3 points) Successfully test your code. Do you require unit tests?
4. (2 points) Write code that is consistently commented and formatted correctly using good variable names, efficient design choices, atomic functions, constants instead of magic numbers, etc.