COMP 3522 Lab #4: Inheritance and polymorphism

Christopher Thompson, Jeffrey Yim
jyim3@bcit.ca

Due Friday 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. Clone your repo using github classroom: https://classroom.github.com/a/KjJD7kxn
2. Fill out your **name** and **student number** at the top of **inheritance.cpp**
3. Ensure you commit and push your work frequently. You will not earn full marks if you don't

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. Place the new animal classes in their own separate .cpp/.hpp files
3. 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.
   (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 member 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.

4. 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!

5. 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!

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

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

8. 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!)

9. Ensure you use the virtual keyword so we can employ polymorphism.
10. You may need to create some (inline?) accessors and mutators.
11. When you override a member function use the override keyword in the derived class' version of the function header.
12. Ensure you are not duplicating code.
13. Create three animal pointers in the main method. 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.

    ```
    cout << "---Create all animals---" << endl;

    Animal *animalPtr = //new animal object with dynamic memory

    Animal *birdPtr = //new bird object with dynamic memory

    Animal *caninePtr = //new canine object with dynamic memory
    ```

14. Demonstrate how the move method works. Demonstrate how the sleep and eat methods work. Can you demonstrate how the hunt method works with your animal pointer that points to a canine? Why not? How can you overcome this (Hint: casting).
15. Use your overloaded insertion operator to print out the information of all the animals you created before they moved, after they moved, and after they've unsuccessfully and successfully hunted
16. Remember to delete dynamic memory
17. That's it! Make sure your code is pushed to Github classroom

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 and ensure the output shows the animals' initial information, moving, and hunting.
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. Prevent memory leaks.

4 Sample output

```
---Create all animals---

animal constructor

animal constructor

bird constructor

animal constructor

canine constructor
```

```
---Print all animals---
animal:[age=10, id=0, alive=1, location=(2,8,0)]
bird:[age=15, id=1, alive=1, location=(6,3,7)]
canine:[age=15, id=2, alive=1, location=(8,1,0)]

---Make all animals move and print---
Animal 0 moved to (1, 1, 0)
Bird 1 moved to (1, 2, 0)
Canine 2 moved to (1, 2, 0)
animal:[age=10, id=0, alive=1, location=(1,1,0)]
bird:[age=15, id=1, alive=1, location=(1,2,0)]
canine:[age=15, id=2, alive=1, location=(1,2,0)]

---Make all animals sleep---
Animal is sleeping
Bird is sleeping
Canine is sleeping

---Make all animals eat---
Animal is eating something
Bird is eating a worm
canine is eating grass

---Make canine hunt bird---
Canine 2 successfully hunted bird

---Print all animals at end---
animal:[age=10, id=0, alive=1, location=(1,1,0)]
bird:[age=15, id=1, alive=0, location=(1,2,0)]
canine:[age=15, id=2, alive=1, location=(1,2,0)]

---Delete dynamic memory---
animal destructor
bird destructor
animal destructor
canine destructor
animal destructor
```

FAQ

How do I initialize member variables in the parent class using member initialization in the child class?

Call the parent's constructor in the derived class constructor's member initialization area, and pass any shared parameters.

```
Child::Child(int param1, long param2, long param3, long param4) :
Parent(param1, param2, param3, param4) {}
```

Can I store x and y with something other than a pair? Like a tuple?

Yes, you can store the x,y,z values in a tuple or even your own custom struct type

Why isn't my Bird, or Canine's overloaded insertion operator called? My "cout <<" code seems to always call Animal?

Hint: Double check the object that's being passed when you call "cout <<". Are you really passing a Bird or Canine? Do you have to perform some pointer casting and dereferencing to get a Bird or Canine object?

Can I write more functions and data members than the ones you've suggested?

Yes definitely! You're free to add as many functions and data members as you need