CPS212 - COMPUTATIONAL STRUCTURES AND ALGORITHMS
Lab 2 - Due: Wednesday, Feb 4th, at the start of class

Objective: To introduce you to working with C++ inheritance and polymorphism.

Introduction

As you know from the syllabus, one of the goals of this course is to help you learn to use C++. We will do this largely through the labs, using a "learn as you go" approach. In this lab, you will be taking Java code (created in CPS112), and re-implementing it in C++.

In your final lab in CPS112, you developed a Java class hierarchy that can be used by a pet kennel to keep track of pets that have been boarded by their owners. For this lab, you will re-create this hierarchy, but using C++. By way of review, the requirements for the CPS112 lab are attached, along with a UML diagram showing the hierarchy. You will probably want to refer to the work you did for this lab last spring. In addition, during the time this lab is meeting, the professor's version of the code from last spring will be accessible online through our CPS212 GoogleGroup. (Since Prof. Bjork plans to assign this lab in CPS112 again this spring, please do not make any printed copies!)

Resources:
Inheritance in C++: http://www.cs.bu.edu/teaching/cpp/inheritance/intro/
Polymorphism in C++: http://www.cs.bu.edu/teaching/cpp/polymorphism/intro/

Differences Between C++ and Java Versions of the Pet hierarchy

Because this version is being done in C++, there will be some differences:

- Create two files: a header file, pet.h, which will contain the declarations for all eight classes, and an implementation file, pet.cpp, which will contain the implementations for all eight classes. To access the declarations, pet.cpp should contain the following at the start:
#include "pet.h"

As was the case last spring, the professor will provide a test driver (lab2.cpp), which will reside in a separate file that you will be given - this time in C++ source form. (Also, the professor will provide date.h and date.cpp – include these files as part of your C++ project)

- In the Java version, the printBill() method wrote the bills to an output file specified by the test driver. In the C++ version, it will write the bills to standard output (cout). (Thus, in the Java version, printBill() took the file as a parameter, while in the C++ version it will not take any parameters.)

- Use names of fields that end with an underscore – e.g. name_, etc.

• In Java, the standard character string class is called String; in C++ it is called string (note the case distinction.) Further, to use this class, the following should appear at the start of your header file. (Note that, in this case if you use the name string.h you will get a very different header!)

#include <string>

• In Java, access to the various classes needed for text output is obtained by importing java.io.*. In C++, you will use the iostream library (and, in particular, the standard output stream cout). To use this library, the following should appear at the start of your implementation file:

#include <iostream>

• In the Java version, you used Math.ceil to handle the requirement that for dogs, the daily rate is $7 for the first 20 pounds plus $1 for each 10 pounds or fraction thereof above 20. . In C++, you will use the ceil function. To use this function, the following should appear at the start of your implementation file. (Note: because math.h is a C header, its name ends in ".h")

#include <math.h>

• In the Java version, dates were represented using the class java.util.Date, and the DateFormat class was used to format them. For the C++ version, you will use a Date class supplied by the professor. For this reason, the following should appear near the start of your header file, and the file date.h and date.cpp furnished by the professor will need to be in your directory, since Date is not a standard library class. (That's what the use of " " instead of < > in #include indicates - the compiler should look for the header file in your current working directory, rather than in the standard library directory.)

#include "date.h"

When outputting dates, it will not be necessary to do anything special to format them - e.g. if arrivalDate_ is defined as a field in the current object, then the following code will output the arrival date appropriately formatted to cout:

```
cout << arrivalDate_;
```

You can use the subtraction operator to directly calculate the number of days elapsed between two Date objects - e.g. if arrivalDate_ and departureDate_ are defined as fields in the current object, then the following code will calculate the charges for the pet (welcome to the world of operator overloading in C++!):

```
getDailyRate() * (departureDate_ - arrivalDate_)
```

Note: The includes for string and date.h appear in the header file, pet.h, because the class declarations require fields of classes string and Date. When pet.cpp includes pet.h, it will

include the headers for these classes as well. The includes for iostream and math need only appear in the implementation file, pet.cpp, because they are only used by the implementation code.

IMPORTANT: You are expected to use appropriate whitespace and commenting - including file, class, and method prologue comments - in all the code you turn in.

Detailed Directions

- See detailed directions from Lab1 about the proper naming standard and the proper way to turn in your lab files. (using CPS212drop)

- In this lab session, you will be using Eclipse – see the previous lab's explanation about setting up a C++ project and so forth.

- Create a file *pet.h* and a file *pet.cpp* which will contain the class definitions and class implementations. Use the Pets description and UML diagram below (and the Java program from CPS112) as reference.

- Include date.cpp, date.h and lab2.cpp in your project folder.. (There are two ways to place already created files in a project – either import or "cut and paste". Perhaps the easiest is to just "cut and paste" – here's how it works: create the necessary header/source files for your project and then simply cut and paste the lines from the supplied files.)

- Once both files (pet.cpp and pet.h) are complete, you can use the proper Eclipse command to compile your code and the date/test driver furnished by the professor, and then link them to produce an executable. Do not, under any circumstances, make changes to the source code of the test driver!

- You can now test run your program by using the run command: Check the output of your program carefully! Your output should match the sample output which will be available in the lab. Be sure to fix errors of both correctness and neatness. Show the output produced by the program to the professor – so that he can be impressed by your progress.

Place a copy of the final version of your source files pet.h and pet.cpp, plus your correct output and executable into a properly named folder, and then drop them into the class dropbox.

APPENDIX A: CS112 REQUIREMENTS FOR JAVA PETS HIERARCHY

Pets are classified as rodents, birds, reptiles, cats or dogs. The particular information that the kennel stores about the pet depends on what kind of pet it is.

• All pets have a name, an owner, an arrivalDate, and a departureDate. The name and owner are Strings, and the arrivalDate and departurreDate are Dates.

• Rodents, birds, and reptiles are considered small pets. In addition to the information listed above, they also have a species (a String) - which will be something like hamster or iguana .

• Cats and dogs are considered medium pets. In addition to the information listed above, they also have their weight recorded (a float), together with an indication as to whether or not their rabies shot is up to date (a boolean - true if a shot is needed.) (If the rabies shot is not up to date, the kennel's vet will give the animal a shot and the vet will send the owner a separate bill.)

• For dogs, their breed (e.g. beagle, collie, bichon) is also recorded as a String.

The kennel calculates its charges as number of days the pet stays in the kennel times daily rate. The daily rate is as follows: for rodents, the rate is $1; for birds $2; and for reptiles $3. For cats the daily rate is $5 if the cat weighs 15 pounds or less and $7 if it weighs more. For dogs, the daily rate is $7 for the first 20 pounds plus $1 for each 10 pounds or fraction thereof above 20. (Example: a 45 lb. dog would be billed $7 + $3 = $10 per day).

Each class of Pet (Rodent, Bird, Reptile, Cat, or Dog) has an appropriate constructor that allows all relevant information to be specified at the time the object representing the pet is constructed. (E.g. when an object representing a dog is constructed, the name, owner, arrival date, departure date, weight, need for a rabies shot, and breed must be specified.)

In addition, the following methods are applicable to all pets. Some can be implemented just once (in class Pet) and some will need to be implemented two or more times at different points in the hierarchy.

• Accessor for information common to all pets: getName().
• Accessor that returns a String describing the pet: getDescription()
    • For a small pet (rodent, bird, or reptile) the result returned is the word little followed by the species of the pet.
    • For a cat, the result returned is the phrase fat cat if the cat weighs more than 15 pounds, else just cat .
    • For a dog, the result returned is the dog s breed.
    Examples: little hamster , little parakeet , little iguana , cat , fat cat , beagle

• Accessor that returns the daily rate for boarding the pet (as an int): getDailyRate(). The daily rate is determined as described above.
• Accessor that prints a bill for the pet to a PrintWriter: printBill(PrintWriter). The bill should look like the following, where the elements in angle brackets are replaced by the appropriate information for the particular pet:

To my owner <owner>: I stayed at the Skunk Hollow Pet Hotel* from <arrival> to <departure> I cost you $<charge calculated from days stayed and daily rate>

Your loving <description>, <name>

In the case of a medium pet (cat or dog), the above should be followed immediately by one or the other of the following two lines, as appropriate:

P.S. My rabies shot was up to date.
P.S. I got a rabies shot - ouch. The vet will send you a separate bill.

*Feel free to use a different name for the kennel if you wish.

Example:

To my owner, J. Arbuckle
I stayed at the Skunk Hollow Pet Hotel from 01/01/02 to 01/31/02
I cost you: $210

Your loving fat cat, Garfield

P.S. I got a rabies shot - ouch.
The vet will send you a separate bill.

APPENDIX B: UML DIAGRAM FOR PETS CLASS HIERARCHY