Lab 10 – Pair Programming & Inheritance

CPSC 1021 – Spring 2017

CLEMSON School of COMPUTING

Introduction

In this week's lab, you will use pair programming principles in developing a mini-project that uses inheritance.

Lab Objectives

- Implement classes using inheritance to meet specifications.
- Learn and apply pair programming principles in developing a software project

Submission Instructions

Tar and submit your Package.cpp, Package.h, ThreeDay.cpp, ThreeDay.h, OverNight.cpp, Overnight.h, Person.cpp, Person.h, and driver.cpp to Handin (http://handin.cs.clemson.edu) All code files must contain a header consistent with headers in previous labs. Please make sure it contains all team member names.

Lab Instructions

Part A: Formal Introduction to Pair Programming

There are several different methods that can be applied to developing software projects—for instance, you might have heard about the Waterfall Method. Agile Methodology is a recent and popular method of software development, which is described as "being an alternative to traditional project management….[that] helps teams respond to unpredictability." One method often used to support agile development is known as Extreme Programming, which emphasizes teamwork in creating a robust software development model that allows programmers to respond to changing requirements and circumstances. One practice of Extreme Programming is pair programming.

Pair programming, simply put, is two individuals sitting in front of *one* computer (with *one* keyboard and *one* mouse) where each partner takes turns coding and monitoring/planning. At first glance, this sounds like a waste of resources, however, research (and industry experience discussed later) finds several advantages.

In 1995, a researcher named Kraut studied 750 programmers and found that the most used communication skill while programming was discussing issues encountered with colleagues. As a second-semester programming student, can you relate? When stuck on a lab or programming project, you may typically ask your TA or instructor for guidance during office hours (or if permitted, collaborate with a classmate). The "standard" for programmers in industry who confront a problem that he or she cannot solve alone is to work with a colleague, which might involve someone critiquing code or brainstorming ideas on a whiteboard.

Pair programming thus creates an environment where you always have a colleague close by.

Research shows that together, pairs of programmers can:

- Solve problems they couldn't solve alone
- Help improve each other's skills
- Develop higher quality code (fewer bugs, better performance, etc.)
- Reduce time to completion
- Typically generate more than twice as many possible solutions to a problem as opposed to working independently
- Typically focus in on the best solution and implement it with better quality.

Note: Pair programming is not the same as two programmers working next to each other. If your attention is devoted to different displays, you are not using pair programming. For this lab, **do not develop any code without your partner present and paying attention**.

If you're skeptical of pair programming, you are urged to participate in this lab with an open mind and consider that one of the Fortune 100 companies that heavily recruits Clemson computer science, computer engineering, and CIS students is moving their entire software development processes to utilize agile and extreme programming practices. Yes, that involves pair programming! Thus, consider this exposure to pair programming as learning an additional skill to put in your software development toolbox.

Here are some specific tips and ground rules for pair programming for this lab in CPSC 1021:

- One person takes on the role of **driver** by actively programming using the keyboard and mouse.
- One person takes on the role of **navigator** by actively checking the driver's code for mistakes, thinking about higher-level concepts, and planning ahead.
- Pairs should switch roles frequently (as often as every 5-10 minutes). No more than 10 minutes should go by without a change in roles.
- All contributions, whether good or bad, should be considered joint (e.g., "ours" and not "mine" or "yours")
- Make sure your partner stays focused and on-task
- It's okay to have a healthy debate or disagreement. Find the middle ground.

Part B: Inheritance

Inheritance is one of the main features of object-oriented programming (in addition to encapsulation and polymorphism, which will be covered in future lectures and labs). Inheritance promotes code reuse by allowing you to create a base (or sometimes referred to as a parent or super) class with some general characteristics, and then let other classes known as child classes (or derived or inherited) to inherit the general characteristics from the base class. Each child class can then add more functions or member variables that are more specific, thus more unique to their own purposes.

The syntax for inheritance is:

```
class derived_class:access base_class {
  // body of new class
  // access refers to access specifier
}
```

The access specifier used in the example above controls the ways that variables and functions in the base class may be accessed via the derived class. Note, there are two types of access specifiers used in C++ object oriented programming and it's important to understand the distinction between them:

1. Base class access: determine access for inherited members

2. Member access: determine access for members in the defined class
    a. Private: members can only be accessed by members of its class
    b. Protected: members can be accessed only by members of its class, however, derived classes also have access to protected members
    c. Public: can be accessed by other parts of a program including a derived class

Part C: Your Project

You will write a small program to model how package delivery services work. A provider like UPS offers a variety of different shipping options with different costs associated with it. You and your partner will develop an inheritance hierarchy to represent various types of packages using the following constraints:

- Package is the base class.
- ThreeDay and OvernightPackage are derived classes of Package
- The base class "has an" instance of Person. Person provides variables that represent the name and address of a person.
- The base class should store the weight (in ounces) and cost per ounce (in dollars) to ship the package
- The base class constructor should initialize these data members and include reasonable checks on the values provided (such as the cost per ounce isn't negative)
- The base class should provide a function calculateCost that returns the cost associated with shipping the package.
- ThreeDay should inherit the functionality of the base class but include a data member that represents an additional fee per ounce (in dollars) that is also charged for this service. Thus, you will need to redefine calculateCost for the ThreeDay class
- OvernightPackage should also inherit from Package but contain members for a flat fee (in dollars) and an additional fee per ounce (in dollars) for this service. You will also have to redefine the member function calculateCost to take this additional cost into account.
- driver.cpp creates objects of each type of Package to tests your classes.

Planning Requirements:

You are strongly encouraged to take a few minutes with your pair programming partner to plan your strategy for completing this lab.

Below is a sample output for this lab.

Sample output:

```
Enter the Sender's Name: Donald Duck
Enter the Sender's Street Address: 123 Disney Ave
Enter the Sender's City: Walt
Enter the Sender's State: Disney
Enter the Sender's Zip Code: 54321

Enter the Recipient's Name: Daffy Duck
Enter the Recipient's Street Address: 321 Mickey Mouse Dr
Enter the Recipient's City: Walt
Enter the Recipient's State: Disney
Enter the Recipient's Zip Code: 54321

Enter the weight, in oz, of the package: 6.3
Enter the cost per oz, in dollars, of the package: .30
Enter the additional fee per ounce of the 3-day package: .10
Enter the additional fee per ounce of the overnight package: .20
Enter the flat fee for the overnight package: 5.00

SENDER ADDRESS:
Donald Duck
123 Disney Ave
Walt, Disney  54321

RECIPIENT ADDRESS:
Daffy Duck
321 Mickey Mouse Dr
Walt, Disney  54321

Cost calculation for Package: $1.89
Cost calculation for ThreeDay: $2.52
Cost calculation for OvernightPackage: $8.15
```