SCIT-EIS-UOW
CSCI251/CSCI851 Advanced Programming
Autumn 2019

Laboratory Exercise 7 (Week 8)

Note that lab exercises marked with a * are effectively extension exercises.

1 Task One: Debugging +

1. There are two related pieces of debugging code, `Debug-A.cpp` and `Debug-B.cpp`. They are based on exercises from

   `Joyce Farrell, Object Oriented Programming Using C++, 3rd Edition, Thomson Learning.`

   Fix both so `Debug-A.cpp` can populate `Persons.txt` based on user input, and `Debug-B.cpp` can read data from `Persons.txt` and show their details and a count on the number of them. Notice that reading and writing are at the level of `Persons` objects using the insertion and extraction operators.

2. Buggy inheritance: Fix `Debug-C.cpp`. The run of this program, with input as shown (Blob etc.), should be:

```
Enter painting's title Blob
Enter artist Degas
Enter painting's title Blob
Enter artist Alice
Enter painting's title Blob
Enter artist Bob
Enter painting's title Blob
Enter artist Picasso

Blob by Degas value $25000
Blob by Alice value $400
Blob by Bob value $400
Blob by Picasso value $25000
```

3. Consider that we have a class `X` with prototypes for the manager functions as below. Let `Y` be an abstract data type we have defined. Why can some of the manager functions be considered to be providing implict casting? Make sure you identify which provide such casting as well as why.

```
X();
X(string);
X(int);
X(double, double);
X(Y);
~X();
```

2 Task Two: Relationships

1. For each of the following lists of related classes, describe which relationship(s) seems the most appropriate: association, aggregation, composition, inheritance (specialisation/generalisation), ... Where a multiplicity makes sense, what values would likely be appropriate?
    (a) Library, books.
    (b) Insects, legs, wings.
    (c) Pet shop, big dogs, little dogs.
    (d) Insects, ants, spiders.
    (e) Pencil case, pencils.
    (f) Robots, androids, humans.
2. Consider that we intend to model apples, making use of at least two classes, `Tree` and `Apple`.
    (a) Why might it make sense to have a private constructor for `Apple` that can only be accessed by an instance of a class derived from `Tree`?
    (b) Sketch code for the implied relationship.
3. The code fragment in `Three.cpp` contains three classes: `Subject`, `Student` and `Date`. Which of the relationships illustrates which concepts? Add code to flesh out the example.
4. Consider `A-Class.cpp`.
    (a) What concept does it illustrate?
    (b) How are the classes related?
    (c) What happens if one of the objects in main was deleted? How does this differ across those objects?
    (d) What happens if we add an additional company and add a contract between John and the new company?

3 * Task Three: Some short tasks

1. You should have used the stream manipulator `endl` a lot.
    (a) How do the stream manipulators `flush` and `ends` differ from `endl`?
    (b) Devise a test to illustrate the difference between using no output stream modifier, using `endl`, and using `flush`. Implement the test.
2. Can you have class X inheriting from class Y and class Y inheriting from class X at the same time? Write code to test this.