CS1020: Data Structures and Algorithms I

Tutorial 2 – OO Design, Inheritance

(Week 4, starting 30 January 2017)

1. OO Design

Design a simple module registration system for SoC to manage students joining and leaving modules. List two **classes** that you think are necessary. Specify the **data** and **functionalities** for each class. Provide at least one **constructor**, one **accessor** and one **mutator** for each class. Please refer to the lecture notes for these concepts if you are not familiar with them. This is a *design question, implementation is not required*.

The following webpage provides a brief introduction to UML, a design language suited for the OO Model. The UML Class Diagram will prove useful for you to express your answer in a drawing that many others understand:

https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/

Do the data and functionality you have specified translate into instance members or class members?

2. Advanced OO

You are part of a mobile development team that is building a multiplayer catching/tagging game.

Each Player has a **name**, and is at a particular **location**. Some players have the ability to *hide* their location as long as the player's ability is active. Some of such "hiders" have a second special ability, *stealth* mode. In *stealth* mode, they will not appear at all, as long as the player's stealth ability is active.

Your junior has written this code to describe the different types of players generally. However, her code cannot be compiled, as there are 3 errors within these 3 classes. (Only 1 of these errors affect compilation)

[Code on page 2...]

'private/public' - access modifier / scope

Tip: Paste in vim and compile in sunfire! Once compilation errors are rectified, instantiate objects and test.

```
class Player {
     private String _location; // e.g. COM2-02-20
     private String _name; // e.g. ictm
     public Player(String locn, String name) {
          _location = locn; _name = name;
     }
     public String getName() { return _name; }
     public void ping() {
          System.out.println(_name + " is at " + _location);
     }
}

class Hider extends Player {
     protected String _location;
     protected String _name;
     protected boolean _isHiding;
     public Hider(String locn, String name) { Player(locn, name); }
     public void ping() {
          if(_isHiding) System.out.println(getName() + " is hiding!");
          else super.ping();
     }
     public void hide() { _isHiding = true; }
     public void expose() { _isHiding = false; }
}

class Stealther extends Hider {
     private boolean _isInvisible;
     public Stealther(String locn, String name) { super(locn, name); }
     public void ping() { if(!_isInvisible) ping(); }
     public void activateStealth() { _isInvisible = true; }
     public void expose() { _isHiding = false; _isInvisible = false; }
}
```

these 2 should be protected attr.s;
no longer need to be declared in subclass Hider

**variables cannot be overridden

*constructors cannot be inherited

change Player to super

Hider has _location _name attr.s but it cannot access,
it thus have to call super method to access them.

change to super.ping()

(a) What does the `protected` keyword mean? In this example, how is it useful?

method / attr. can only be accessed by its subclass. encapsulation / prevent unwanted data access

(b) Within a method in the Hider class, why can `getName()` be invoked?

inherited from its superclass Player

(c) How is overriding demonstrated here, and how is it useful?

ping() method in Stealther class overrides ping() in Hider class. (also, expose() method)

(d) Identify and rectify the 3 errors.

constructors of chain:
if C extends B extends A,
when we call C(),
A() will be called first, followed by B(), then C().

- Practise consist

Revise a concept
Test understanding
Code to confirm