Homework 1 – Intro to OO

Due: Sep. 28, 2004

1- Consider a university campus with the following community: Staff, Faculty and Students. Staff includes full-time and part-time employees. Faculty can be Adjunct or Tenure-track faculty. And students can be Undergraduate or Graduate. Graduate students can further be Masters or PhD, and Undergraduate students can be in their Junior or senior. Staff and Faculty are employees of the university, and, along with students, form the "people" in campus.
All employees and students of the university have their information stored in a *Directory*, that is used to view the information of different members of the campus. The search for an individual can be done by their name or ID. Once a member is located, all their information (except for confidential info, such as SSN) is displayed.

The university also has different units, such as the library and all the academic departments, such as CIS, Education, Science, etc., the Human Resources and Administration buildings.

Every member of the campus is affiliated to one unit, i.e., their office location or the department a student is enrolled in.

Your task is to model a university system based on the above information. You classes include: People, Units, Directory, Faculty, Staff, Students, Employee, Library, HR, Administration, Registrar, etc.. These are some of the issues you should address:

a) Which classes are abstract classes?
b) What type of abstractions will you use to model this problem? Identify where "is-a", "has-a", "part-of", etc.. are used. Illustrate inheritance through an inheritance tree.
c) Identify the attributes and methods of each class
d) When a student or employee is located in the directory, the information displayed always has their name and location, for example. Illustrate how one method called *describe* or *toString* can be invoked to display the info of all types of objects.
e) How are the constructors defined? What do they initialize?
f) How is a new employee added to the directory – i.e., what method is required and where should it be located?
g) What type(s) of inheritance is used in your solution? Does it allow subtyping? Give examples.
h) Would it be appropriate/efficient/recommended to have *Directory* as an interface or abstract class? Why? What about *People?*
i) Illustrate three different abstraction **levels** of your system.

2- Insert something at "insert here" in the following C++ code:

```
#include <iostream.h>

        // insert here
 main()
 {
       question2 q2;
        cout << "Hello, world\n";
      delete q2;
  }
```

so that it produces the output:

Initialize
Hello, world
Clean up

Do NOT modify *main()* in any way.

3- Given the following C# code:

```
class Animal {
        public virtual void WhoAreYou() { Console.WriteLine("Iam an animal");}
}

class Dog: Animal {
        public override void WhoAreYou(){Console.WriteLine("I am a dog"); }
}
```

Add a *Cat* and *Cow* subclasses to the *Animal* class. Each subclass should implement a method *Likes(string food*), which returns true if the animal likes the food that was passed as a parameter. Each animal should also have a method *Speak()*, which returns a string with the sound that is produced by this animal (e.g., "woof" for *Dog*). An animal farm should then be usable as follows:

```
foreach (Animal a in farm) // where fram is an array: Animal[] farm
   if (a.likes("fish")) a.Speak();   // this shoud return "woof" or "moo" or a message stating that the given animal does not like the food passed as parameter.
```

The array *farm* can be initialized as:

static Animal[] farm = { new Cow(), new Dog(), new Cat(), new Dog()};

You should also have a *name* method that will return the name of the animal. For example, in the *Dog* class, you would have:

public string Name { get {return “dog”;}) // using property

Since all dog, cow and cat classes inherit from Animal, they will all *override* the methods from *Animal*. Thus, all the methods in *animal* will be abstract, i.e., without any implementation.

4- Define a class “parent” so that the lines of code below are either okay, or give errors, as described by the comments:

```
class child: public parent
{
public:
        play()
        {       f(); // should be okay
                g(); // should be okay
                h(); // should give error
        }
};

void another(void)
{
        parent trap;
        trap.f(); // should be okay
        trap.g(); // should give error
        trap.h(); // should give error
}
```

How would a C++ code differ from Java in this? What about C#?