CSE 655, Assignment #4
Due: June 1, '12.
20 points.

Note: This assignment may be turned in to Neelam in hardcopy (give it to Neelam in class, or in his office (DL 579), or slip it under the door of DL 579), or submitted electronically using the `submit` command:

```
submit c655ab hw4 mySolns
```

where `mySolns` is the name of the *plain text* (not .doc, .pdf etc.) file containing your solution to the problem. If you do this, you must submit by 11:59 pm on Friday, June 1.

The purpose of this assignment is to see the use of OO polymorphism/dynamic dispatch in the implementation of interpreters. Consider a CORE-like language with the following syntax for statements:

⟨*stmt*⟩ ::= ⟨*stmts*⟩ | ⟨*loop*⟩ | ⟨*if*⟩ | ⟨*assign*⟩ | ⟨*in*⟩ | ⟨*out*⟩
⟨*stmts*⟩ ::= ⟨*stmt*⟩ ⟨*stmt*⟩
⟨*loop*⟩ ::= **while** ⟨*comp*⟩ { ⟨*stmt*⟩ };
⟨*if*⟩ ::= **if** ⟨*comp*⟩ **then** ⟨*stmt*⟩ **end**; | **if** ⟨*comp*⟩ **then** ⟨*stmt*⟩ **else** ⟨*stmt*⟩ **end**;

The productions for ⟨*assign*⟩, ⟨*in*⟩, and ⟨*out*⟩ are omitted; assume standard (CORE-like) syntax for those. Note also that there is no ⟨*stmt-seq*⟩ here; instead ⟨*stmts*⟩ is one alternative for ⟨*stmt*⟩ and this gives us the option of using more than one statement in place of a single statement so we don't need ⟨*stmt-seq*⟩; ⟨*stmt*⟩ will be used in place of ⟨*stmt-seq*⟩ in the production for ⟨*prog*⟩ as well.

Now for the problem: we want to implement an *execute()* method that can be used to execute such statements; we also have to implement all the related methods (such as *execute()* method for loops). The key point is that the *execute()* method of the `stmt` class should not explicitly contain a multi-way selection to call the appropriate execute method based on the type of the statement being executed but that the dynamic dispatch mechanism of the underlying OO language is used to achieve that.

To get you started, here is a possible class (in *C++* syntax) for ⟨*stmt*⟩:

```
class stmt {
  public:
    virtual void execute( );
};
```

(In *Java* (and in *Eiffel* and in *Smalltalk*) all functions are *virtual* by default but that is a language detail.)

What you have to do is complete the `stmt` class (if it needs any completion), and write down the classes corresponding to the different types of statements. Do not worry about the parse and print methods.

Make sure that you write down not only the header portions of the classes but also the definitions of the member functions. Your code does not have to be completely legal *C++* (or *Java* if you prefer) but it must be readable and must be conceptually correct for you to get full credit.