Software Design Lecture Notes
Spring 2004

For next Monday:

1) Start Homework 7.

2) Work on a project proposal (due next Thursday)

Quiz 4 Solution

```
def list_others(turtles):
    for t in turtles:
        t.others = []
        for other in turtles:
            if other is self: continue
            t.others.append(other)
```

Program development plan

When creating a new class, I usually start with

1) an __init__ method

   usually the parameters are optional,
   and get assigned to attributes with the same name

2) a __str__ method

   when you say "print obj", Python executes "obj.__str__()"

   Sometimes I also have a print method that displays the
   object in more detail (used for debugging).

3) test code that instantiates the class and prints the instance.

This is what is known in the biz as a "minimal testable unit"

Usually the test code is wrapped in:

```
if __name__ == '__main__'
```

Then I can import the file and get the class definition
without executing the test code.

As I write each new method, I add code to test it.

Object invariants

An "object invariant" is something that should be true for
all objects in a class.

Examples:

1) an object might be required to have certain attributes

2) some attributes might be required to fall in a certain
   range (think of a Time or Date object)

3) there might be relationships among the attributes that
   should be true (example: r.area should be r.width * r.height)

One way or another, it is up to the programmer to establish and preserve invariants.

1) invariants are usually established in __init__

2) all other methods should preserve the invariant, which
   means that IF it's true at the beginning of the method,
   it MUST be true at the end

   There may be places in between where the invariant is
   violated. (these places should be documented)

Inheritance

When you define a new class, you can extend an existing class:

```
class TurtleWorld(World):
    ...

class Turtle(Animal):
    ...
```

The new class is the "child"; the old class is the "parent".

When you invoke a method on an object, Python

1) finds the class of the object

2) checks whether the class defines the method in question

   if so, it invokes it

   otherwise it

3) finds the parent of the class, and goes to step 2

What this means is:

1) Turtle "inherits" all the methods defined in Animal

2) you can define additional methods that pertain to
   Turtles, but not Animals

3) you can override the definition of an Animal method
   and replace it with a more specific Turtle method.

Usually a member of a child class is required to satisfy all the invariants of the parent class, and more.

One way to verbalize the relationship between a child class and a parent class is

"A Turtle is an Animal, except for the following additional methods, attributes and invariants."

But the shorthand is to call this an IS-A relationship.