Sharing Work

**Problem:** Redundant code.
(Any time you copy-and-paste code, you are likely doing something wrong.)

**Solution:** Create a *parent* class with shared code
- Then, create *subclasses* of the *parent* class

---

### Defining a Subclass

```python
class Shape:
    """A shape located at x,y""
    def __init__(self, x, y): ...  
    def draw(self): ...

class Circle(Shape):
    """An instance is a circle.""
    def __init__(self, x, y, radius): ...  
    def draw(self): ...

class Rectangle(Shape):
    """An instance is a rectangle.""
    def __init__(self, x, y, ht, len): ...  
    def draw(self): ...
```

---

### Extending Classes

```python
class <name>(<superclass>):
    """Class specification""
    class variables
    initializer (__init__)
    methods
```

---

### Name Resolution Revisited

- To look up attribute/method name
  1. Look first in instance (object folder)
  2. Then look in the class (folder)
- Subclasses add two more rules:
  3. Look in the superclass
  4. Repeat 3. until reach object

**Often called the Bottom-Up Rule**

```python
c1 = Circle(1,2,4.0)
r = c1.radius  
c1.draw()
```
Q1: Name Resolution and Inheritance

```python
class A:
    def f(self):
        return self.g()
    def g(self):
        return 10

class B(A):
    def g(self):
        return 14
def h(self):
    return 18
```

• Execute the following:
  ```python
  >>> a = A()
  >>> b = B()
  ```

• What is value of a.f()?
  A: 10  
  B: 14  
  C: 5  
  D: ERROR  
  E: I don’t know

Q2: Name Resolution and Inheritance

```python
class A:
    def f(self):
        return self.g()
    def g(self):
        return 10

class B(A):
    def g(self):
        return 14
def h(self):
    return 18
```

• Execute the following:
  ```python
  >>> a = A()
  >>> b = B()
  ```

• What is value of b.f()?
  A: 10  
  B: 14  
  C: 5  
  D: ERROR  
  E: I don’t know

Q3: Name Resolution and Inheritance

```python
class A:
    x = 3  # Class Variable
    y = 5  # Class Variable
    def f(self):
        return self.g()
    def g(self):
        return 10

class B(A):
    y = 4  # Class Variable
    z = 42  # Class Variable
    def g(self):
        return 14
def h(self):
    return 18
```

• Execute the following:
  ```python
  >>> a = A()
  >>> b = B()
  ```

• What is value of b.x?
  A: 4  
  B: 3  
  C: 42  
  D: ERROR  
  E: I don’t know

Q4: Name Resolution and Inheritance

```python
class A:
    x = 3  # Class Variable
    y = 5  # Class Variable
    def f(self):
        return self.g()
    def g(self):
        return 10

class B(A):
    y = 4  # Class Variable
    z = 42  # Class Variable
    def g(self):
        return 14
def h(self):
    return 18
```

• Execute the following:
  ```python
  >>> a = A()
  >>> b = B()
  ```

• What is value of a.z?
  A: 4  
  B: 3  
  C: 42  
  D: ERROR  
  E: I don’t know

The `is` Function

```
== compares equality
is compares identity
```

```python
c1 = Circle(1, 1, 25)  c2 = Circle(1, 1, 25)  c3 = c2
c1 == c2?
c1 is c2?
c2 == c3?
c2 is c3?
```

Q5: `isinstance` and Subclasses

```python
>>> shape1 = Rectangle(0,0,10,10)
```

```python
>>> isinstance(shape1, Square)
```