Programming Challenge 31

01) How does your textbook define inheritance?

According to our textbook inheritance is “the process by which a new class—known as a derived class—is created from another class, called the base class. A derived class automatically has all the member variables and all the ordinary member functions that the base class has, and can have additional member functions and additional member variables” (600, 3rd edition).

02) Besides the terms derived class and base class, what are the additional terms used to describe the relationship between derived classes and base classes?

Base classes are also called “parent classes” while derived classes are also called “child classes.” Base classes that have many derived classes of derived classes are also called “ancestor classes.” According to our textbook: “When discussing derived classes, it is common to use terminology derived from family relationship. A base class is often called a parent class. A derived class is then called a child class. This makes the language of inheritance very smooth. For example, we can say that a child class inherits member variables and member functions from its parent class. This analogy is often carried one step further. A class that is a parent of a parent of a parent of another class (or some other number of “parent of” iterations) is often called an ancestor class. If class A is an ancestor of class B, then class B is often called a descendant of class A” (606, 3rd edition).

03) Are derived classes allowed to redefine member functions inherited from their base class?

Yes, derived classes are allowed to redefine member functions inherited from their base class. This is one of the primary reasons to use inheritance. According to our textbook: “A derived class inherits all the member functions (and member variables) that belong to the base class. However, if a derived class requires a different implementation for an inherited member function, the function may be redefined in the derived class. When a member function is redefined, you must list its declaration in the definition of the derived class, even though the declaration is the same as in the base class. If you do not wish to redefine a member function that is inherited from the base class, do not list it in the definition of the derived class” (618, 3rd edition).

04) Can a derived class invoke the constructor of its base class? If so, where in the derived class implementation will the base class constructor be invoked?

Yes, a derived class can invoke the constructor of its base class. According to our textbook: “A derived class does not inherit the constructors of its base class. However, when defining a constructor for the derived class, you can and should include a call to a constructor of the base class (within the initialization section of the constructor definition). If you do not include a call to a constructor of the base class, then the default (zero-argument) constructor of the base class will automatically be called when the derived class constructor is called” (612, 3rd edition).

11) What type or types do the following two objects have?

```
FoodItem donut;
MagicItem ring;
```

The object “donut” has the types FoodItem and its base class Item. The object “ring” has the types MagicItem and its base class Item. According to our textbook: “An object of a class type can be used anyplace that an object of any of its ancestor classes can be used. If class Child is derived from class Ancestor and class Grandchild is derived from class Child, then an object of class Grandchild can be used anyplace an object of class Child can be used, and the object of class Grandchild can also be used anyplace that an object of class Ancestor can be used” (611, 3rd edition).