Object-Oriented Programming: Polymorphism

12

Objectives

In this chapter you'll:

- See how polymorphism makes programming more convenient and systems more extensible.
- Understand the relationships among objects in an inheritance hierarchy.
- Use C++11's **`overrides`** keyword when overriding a base-class virtual function in a derived class.
- Use C++11's **`default`** keyword to autogenerate a **`virtual`** destructor.
- Use C++11's **`final`** keyword to indicate that a base-class virtual function cannot be overridden.
- Create an inheritance hierarchy with both abstract and concrete classes.
- Determine an object's type at runtime using runtime type information (RTTI), **`dynamic_cast`**, **`typeid`** and **`type_info`**.
- Understand how C++ can implement **`virtual`** functions and dynamic binding.
- Use **`virtual`** destructors to ensure that all appropriate destructors run on an object.

Self-Review Exercises

12.1 Fill in the blanks in each of the following statements:

a) Treating a base-class object as a(n) ________ can cause errors.
ANS: derived-class object.
b) Polymorphism helps eliminate ________ logic.
ANS: `switch`.
c) If a class contains at least one pure `virtual` function, it's a(n) ________ class.
ANS: abstract.
d) Classes from which objects can be instantiated are called ________ classes.
ANS: concrete.
e) Operator ________ can be used to downcast base-class pointers safely.
ANS: `dynamic_cast`.
f) Operator `typeid` returns a reference to a(n) ________ object.
ANS: `type_info`.
g) ________ involves using a base-class pointer or reference to invoke `virtual` functions on base-class and derived-class objects.
ANS: Polymorphism.
h) Overridable functions are declared using keyword ________.
ANS: `virtual`.
i) Casting a base-class pointer to a derived-class pointer is called ________.
ANS: downcasting.

12.2 State whether each of the following is *true* or *false*. If *false*, explain why.

a) All `virtual` functions in an abstract base class must be declared as pure `virtual` functions.
ANS: False. An abstract base class can include virtual functions with implementations.
b) Referring to a derived-class object with a base-class handle is dangerous.
ANS: False. Referring to a base-class object with a derived-class handle is dangerous.
c) A class is made abstract by declaring that class `virtual`.
ANS: False. Classes are never declared `virtual`. Rather, a class is made abstract by including at least one pure virtual function in the class.
d) If a base class declares a pure `virtual` function, a derived class must implement that function to become a concrete class.
ANS: True.
e) Polymorphic programming can eliminate the need for `switch` logic.
ANS: True.

Exercises

NOTE: Solutions to the programming exercises are located in the `ch12solutions` *folder.*