Virtual Destructor Solutions

Virtual Destructor

- Write a program which binds a child object to a pointer to its base class
    - Both classes have a non-virtual destructor function
    - The destructor body prints out the name of its class
- At the end of the program, call delete on the pointer
- What output do you expect to see?
- Compile and run the program
- Explain your results
    - The Shape destructor executes, but the Circle destructor does not
    - The destructor is not virtual, so the static type is used to resolve the call
    - The static type of the variable is Shape*, so only the Shape destructor is called

Virtual Destructor

- Change your program so that the destructor is declared virtual in the base class
- What output do you expect to see?
- Compile and run the program and compare the results
- Explain any differences from the previous output
  - The Circle destructor executes, followed by the Shape destructor
  - The destructor is virtual, so the dynamic type of the object will be used
  - The dynamic type of the variable is Circle*, so the Circle destructor will be called, then the Shape destructor

Synthesized Destructor

- Why is this important?
  - If we do not implement a destructor, the compiler will synthesize one
  - This synthesized destructor is not virtual
  - If we destroy a derived class through a pointer or reference to the base class, only the base class destructor executes
  - The child part of the object is not destroyed
  - This can cause resource leaks or other undefined behaviour

Virtual Destructor Recommendations

- In general, what should we do when writing a class which has virtual member functions?
  - Implement a virtual destructor for it