

Pointers, References and Inheritance Solutions

Pointers and Inheritance

- Write a simple inheritance hierarchy with a base class "Shape" and a derived class "Circle"
- Write a program which
 - Creates a Circle object and a Shape* to it
 - Creates a Shape object and a Circle* to it
- Explain your results
 - Making a Shape* to a Circle compiles
 - Making a Circle* to a Shape does not compile
 - This is because a Circle object always contains a Shape object, but a Shape object may not necessarily be part of a Circle object

Pointers and Inheritance

- Write a similar program, but this time use references instead of pointers
- Explain your results
 - Binding a Shape& to a Circle compiles
 - Binding a Circle& to a Shape does not compile
 - This is because a Circle object always contains a Shape object, but a Shape object may not necessarily be part of a Circle object

Member Function Calls

- Add some member functions to your classes
 - A member function in Shape which is reimplemented in Circle
 - A member function which is unique to Circle
- Write a program which creates a Circle object and a Shape* which points to the Circle object
- Call the Circle member functions through this pointer

Member Function Calls

- Explain your results
 - The Shape version of the first member function is called
 - The call to the Circle-only member function does not compile
 - This is because the compiler sees the pointer as pointing to a Shape object, even though it is actually pointing to a Circle object