Inheritance

Jaime Canizales

City University of New York

jaime.canizales@hunter.cuny.edu

August 15, 2024

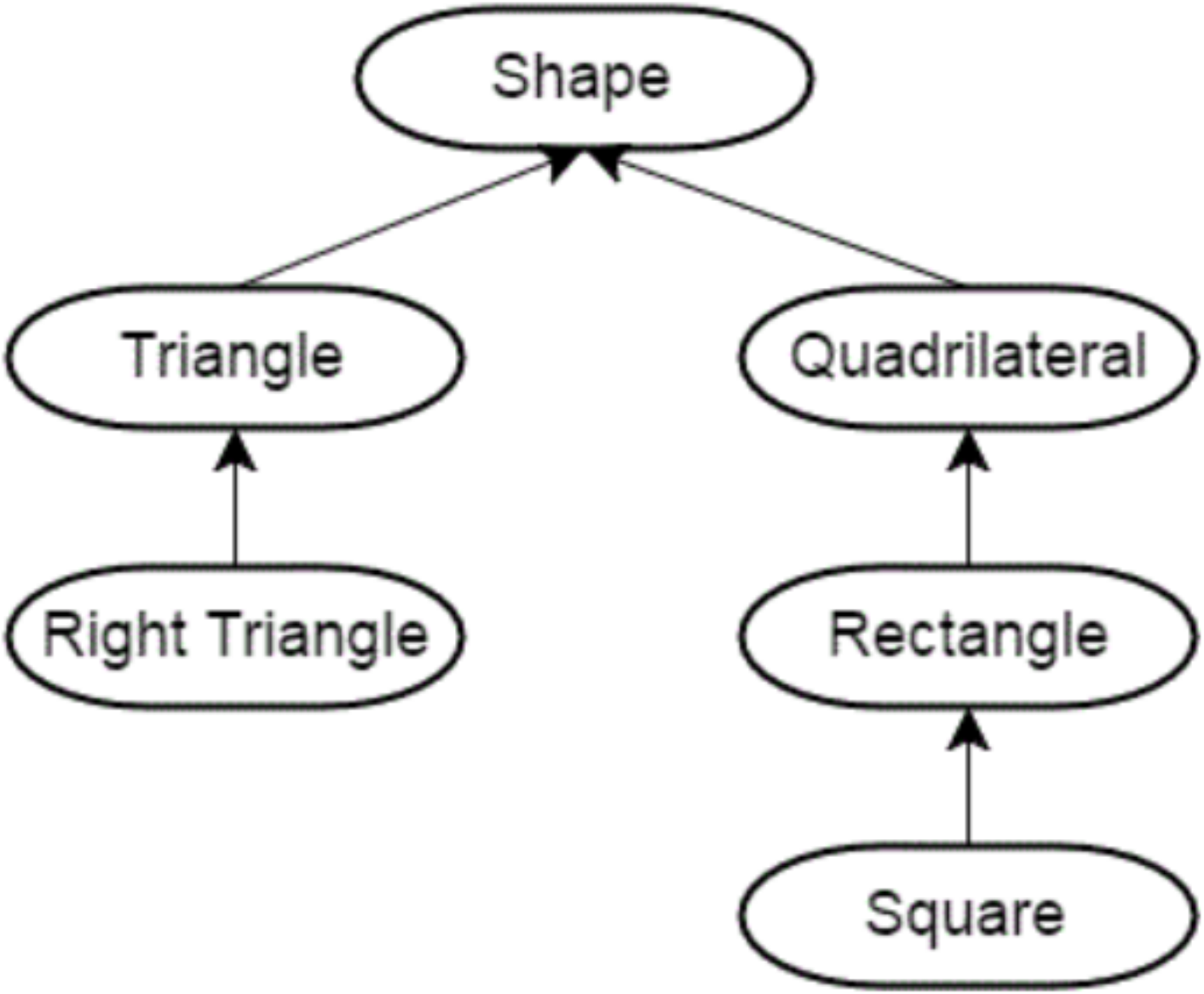
Overview

What is Inheritance?

Problem Statement

Inheritance is a mechanism in C++ which allows for you to pass properties (attributes and methods), from one class to another. The class whose properties and methods are inherited is known as the base class. And the class that inherits the properties from the parent class is the derived class.

Simple Example



Simple Example Cont.

- Here we can see that both triangle and square are shapes. Therfore they can inherit properties from the shape class. In this image we say that triangle and square are the derived classes and shape is the base class.
- Right triangle is the derived class of triangle(base class), and triangle is the derived class of shape(base class). This means the right triangle class inherits properties from both shape and triangle.

Why use Inheritance?

- Allows us to write more concise code by reusing code from a pre-existing class.
- Can help in the understanding and implementation of polymorphism.

What is Polymorphism?

- Polymorphism describes the concept that you can access objects of different types through the same interface.
- This is computed at runtime in C++

Why use Polymorphism?

- For example, you have classes: BaseClass, DerivedClass1 and DerviedClass2, and they all have a method doSomething(). The user presses a key "A", "S", or "D", and, depending on his input, an instance of the appropriate class is created, and its doSomething() method is called.

Constructors and Inheritance

- When you create an instance of the derived class, the constructor for the base class must be called to initialize the data associated with the base class.
- You can set this up by adding the constructor you want to call for the base class in the member intialization list for the derived class.
- If you skip this step, the compiler will try to call the default constructor for the base class. If there is no default constructor you will get a compiler error.

Important Keywords

- **virtual:** Allows for a function in the base class to be overwritten by functions of the same name in the derived class.
- **override:** Ensure that a derived class overwrites a virtual function in the base class.

Types of Inheritance

- Public inheritance will inherit all the public properties of the base class and make them public members of the derived class. Everyone has access to these members. Syntax: **class derived: public base**
- Protected inheritance will inherit all the public properties of the base class and make them protected members of the derived class. Protected means only the current class and classes derived from the current class will have access to protected members. Syntax: **class derived: protected base**
- Private inheritance will inherit all the public properties of the base class and make them private members of the derived class. Only the current class will have access to these members. Syntax: **class derived: private base**
- If you do not choose an inheritance type, C++ defaults to private inheritance (just like members default to private access if you do not specify otherwise).

Technical things to keep in mind

- If you implement a destructor, make sure base class is virtual.
- Virtual functions and polymorphism only works for pointers and references.

Software Architecture of Code Example