

# Virtual Functions Part Two Solutions

- What effect does the `override` keyword have?
  - The `override` keyword signifies that a member function overrides a member function which is inherited from the parent class
  - If it does not, it is a compiler error
- What do you expect to happen when the code on the next page is compiled?
  - Compiler error - `void draw(int radius)` does not override
- Compile the code and verify your answer

- What effect does the `final` keyword have?
  - A member function which is marked as `final` cannot be overridden in derived classes

```
class Drawable {  
public:  
    virtual void draw() { cout << "Drawable::draw()\n"; }  
};
```

```
class Circle : public Drawable {  
public:  
    void draw() override { cout << "Circle::draw()\n"; }  
    void draw(int radius) override { cout << "Circle::draw() with radius\n"; }  
};
```

- What is meant by a "pure virtual member function"?
  - A pure virtual member function has “=0” instead of a function body
  - It is undefined in the class, but can be overridden in derived classes

- Why is it useful?
  - It allows the base class to provide an interface to the class hierarchy, without having to provide an unnecessary function body
- What effect does making a member function pure virtual have?
  - If a class has a pure virtual member function, we cannot create instances of it
  - Every child of this class must implement the member function

- What is an "abstract base class"?
  - An abstract base class contains a pure virtual member function
- What is unusual about an abstract base class?
  - We cannot create instances of it
  - However, we can create pointers and references to an abstract base class, and bind them to instances of its derived classes
- What is an abstract base class used for?
  - Similar to an “interface” in other programming languages

- In the program you wrote in the previous lesson, change `Drawable::draw()` to be a pure virtual function
- Compile and run your program again
- What happens if you try to create an instance of `Drawable`?
  - **Compiler error - cannot create a variable of abstract type**



- Create a function which takes a Drawable instance by value
- What happens? Does this help programmers?
  - Compiler error - cannot declare a parameter of abstract type
  - Protects against “object slicing” in which only the base class part of a variable is passed to the function
  - Enforces use of reference or pointer with base classes which have virtual functions