

Virtual Functions Exercises

- Explain what is meant by static and dynamic binding when calling a member function on a class instance
- What two conditions must apply for dynamic binding to occur?

- Give an example of a situation where dynamic binding is useful
- Describe some of the advantages of dynamic binding

- What is a virtual member function?
- What output would you expect from the code sample on the next page?
- What output would you expect if `Parent::print()` is made virtual?
- Convert the code sample to a working program and compare your answers with the output

```
class Parent {  
    public:  
        void print() { cout << "I'm a parent\n"; }  
};  
class Child : public Parent {  
    public:  
        void print() { cout << "I'm a child\n"; }  
};
```

```
Child child;  
Parent& p;  
child.print();
```

- Using the Drawable and Circle classes from the code on the next page, write a program which creates a vector of pointers to Drawable instances.
- It then stores a Circle instance in this vector, iterates through the vector and calls the draw() member of the vector's elements
- Remove the virtual keyword from Drawable::draw(). Compile and run your program again. What difference does this make?

```
class Drawable {  
public:  
    virtual void draw() { cout << "I'm a Drawable shape!\n"; }  
};
```

```
class Circle : public Drawable {  
public:  
    void draw() { cout << "I'm a Circle!\n"; }  
};
```

- Add this class to your program and append an instance of it to the vector

```
class Triangle : public Drawable {  
public:  
    void draw() { cout << "I'm a Triangle!\n"; }  
};
```

- What other changes do you need to make to the program before you can compile and run it?