```
#include <iostream>  // std::cout

class Circle : public Shape
{
    int d_radius;

public:
    Circle(int radius) : d_radius(radius) { }

    void draw() const  // concrete implementation of abstract draw function
    {
        std::cout << "Circle(radius = " << d_radius << ")\n";
    }
};

class Rectangle : public Shape
{
    int d_length;
    int d_width;

public:
    Rectangle(int length, int width) : d_length(length), d_width(width) { }

    void draw() const  // concrete implementation of abstract draw function
    {
        std::cout << "Rectangle(length = " << d_length << ", "
                                "width = " << d_width  << ")\n";
    }
};
```

Notice that a `Circle` is constructed with a single integer argument, i.e., `radius`, and a `Rectangle` is constructed with two integers, i.e., `length` and `width`.

We now implement a function that takes an arbitrary shape, via a **`const`** *lvalue* reference to its abstract base class, and prints it:

```
void print(const Shape& shape)
{
    shape.draw();
}

void testShape()
{
    print(Circle(1));        // OK, prints: Circle(radius = 1)
    print(Rectangle(2, 3));  // OK, prints: Rectangle(length = 2, width = 3)
    print(Shape());          // Error, Shape is an abstract class.
}
```