// Example illustrating Subclassing with virtual functions
// ECE3090
// George F. Riley, Georgia Tech, Spring 2009

#include <iostream>

using namespace std; // We will discuss namespaces later

class TwoInt { // Simple class with two integer variables
public:
    TwoInt(int a0, int b0);
    // Note the use of the "virtual" keyword below
    virtual void Print() const; // Print the variables
    int First() const; // Return the first int
    int Second() const; // Return the second int
    virtual int Sum() const; // Get the sum
    virtual double Average() const;
    // Hello is NOT virtual
    void Hello() const;
    // Note the use of "private" here
private:
    int a;
    int b;
};

// Define a class FourInt that INHERITS from TwoInt
// This means the FourInt class has EVERYTHING that TwoInt has,
// plus any additional things we might add

class FourInt : public TwoInt {
    // Inherits from TwoInt the members and functions
public:
    FourInt(int a0, int b0, int c0, int d0);
    virtual void Print() const; // Print the variables
    int Third() const;
    int Fourth() const;
    virtual int Sum() const; // Get the sum
    virtual double Average() const;
    // Hello is NOT virtual
    void Hello() const;
private:
    int c;
    int d;
};

// Methods for TwoInt
TwoInt::TwoInt(int a0, int b0):
    a(a0), b(b0) // Initialize with correct constructors
{ // Nothing else needed
}

void TwoInt::Print() const // Print the variables
{
    cout << "a " << a << " b " << b << endl;
}

Program virtual-functions.cc
int TwoInt::First() const  // Return the first int
{
    return a;
}

int TwoInt::Second() const  // Return the second int
{
    return b;
}

int TwoInt::Sum() const  // Return the sum
{
    cout << "Hello from TwoInt::Sum()" << endl;
    return a+b;
}

double TwoInt::Average() const  // Return the average
{
    cout << "Hello from TwoInt::Average()" << endl;
    return Sum() / 2.0;
}

void TwoInt::Hello() const
{
    cout << "Hello from TwoInt::Hello" << endl;
}

// Methods for FourInt
// Define the FourInt constructor, with four integers
FourInt::FourInt(int a0, int b0, int c0, int d0)
    : TwoInt(a0, b0), c(c0), d(d0)
{
    // Nothing else needed
}

void FourInt::Print() const  // Print the variables
{
    TwoInt::Print();  // Let TwoInt "Print Itself"
    cout << "c " << c << " d " << d << endl;
}

int FourInt::Third() const  // Get the third int
{
    return c;
}

int FourInt::Fourth() const  // Get the fourth int
{
    return d;
}

int FourInt::Sum() const  // Return the sum
{
    cout << "Hello from FourInt::Sum()" << endl;
    return TwoInt::Sum() + c + d;
}
double FourInt::Average() const // Compute average
{
    return Sum() / 4.0;
}

void FourInt::Hello() const
{
    cout << "Hello from FourInt::Hello" << endl;
}

void Sub1Ref(const TwoInt& ti)
{// Sub1Ref expects a "TwoInt" reference as a parameter.
    // We can pass any object of type TwoInt or any subclass of TwoInt
    // We can call any TwoInt function,
    cout << "Hello from Sub1Ref()" << endl;
    ti.Print();
    cout << "Sub1Ref() calling sum" << endl;
    int s = ti.Sum();
    cout << "Sub1Ref() calling average" << endl;
    double avg = ti.Average();
    cout << "Sum is " << s << " average " << avg << endl;
    ti.Hello(); // What gets called here?
}

void Sub1Ptr(TwoInt* ti)
{// Sub1Ptr expects a "TwoInt" pointer as a parameter.
    // We can pass any object of type TwoInt or any subclass of TwoInt
    // We can call any TwoInt function,
    cout << "Hello from Sub1Ptr()" << endl;
    ti->Print(); // Note different syntax from Sub1Ref() above
    cout << "Sub1Ptr() calling sum" << endl;
    int s = ti->Sum();
    cout << "Sub1Ptr() calling average" << endl;
    double avg = ti->Average();
    cout << "Sum is " << s << " average " << avg << endl;
    ti->Hello(); // What gets called here?
}

void Sub1Value(TwoInt ti)
{// Sub1Value expects a TwoInt BY VALUE. Although this appears similar
    // to the two examples above, it is quite a bit different. This will
    // become apparent when we discuss virtual functions.
    cout << "Hello from Sub1Value()" << endl;
    ti.Print();
    cout << "Sub1Value() calling sum" << endl;
    int s = ti.Sum();
    cout << "Sub1Value() calling average" << endl;
    double avg = ti.Average();
    cout << "Sum is " << s << " average " << avg << endl;
    ti.Hello(); // What gets called here?
}

int main()
{
    TwoInt ti1(1, 2);
    TwoInt ti2(2, 4);
    Program virtual-functions.cc (continued)
FourInt fi1(10, 11, 12, 13);
FourInt fi2(fi1);

// We can call the Sub1 variants passing "TwoInt" objects as parameters
// either by reference, by pointer, or by value
cout << "Calling Sub1Ref() passing TwoInt" << endl;
Sub1Ref(ti1);
cout << "Calling Sub1Ptr() passing TwoInt" << endl;
Sub1Ptr(&ti2);
cout << "Calling Sub1Value() passing TwoInt" << endl;
Sub1Value(ti1);

// Note we can pass any subclass of TwoInt to the Sub1's
cout << "Calling Sub1Ref() passing FourInt" << endl;
Sub1Ref(fi1);
cout << "Calling Sub1Ptr() passing FourInt" << endl;
Sub1Ptr(&fi2);
// What happens when we pass a FourInt to Sub1Value?
cout << "Calling Sub1Value() passing FourInt" << endl;
Sub1Value(fi1);

cout << "Calling Hello on TwoInt and FourInt" << endl;
ti1.Hello();
fi1.Hello();
}

Program virtual-functions.cc (continued)