```
1  // Example illustrating Subclassing
2  // ECE2036
3  // George F. Riley, Georgia Tech, Fall 2012
4
5  #include <iostream>
6
7  using namespace std;  // We will discuss namespaces later
8
9  class TwoInt { // Simple class with two integer variables
10 public:
11   TwoInt(int a0, int b0);
12   void Print() const;   // Print the variables
13   int  First() const;   // Return the first int
14   int  Second() const;  // Return the second int
15   int  Sum() const;     // Get the sum
16   // Note the use of "private" here
17 private:
18   int a;
19   int b;
20 };
21
22 // Define a class FourInt that INHERITS from TwoInt
23 // This means the FourInt class has EVERYTHING that TwoInt has,
24 // plus any additional things we might add
25
26 class FourInt : public TwoInt {
27   // Inherits from TwoInt the members and functins
28 public:
29   FourInt(int a0, int b0, int c0, int d0);
30   void Print() const; // Print the variables
31   int  Third() const;
32   int  Fourth() const;
33   double Average() const ;
34
35 private:
36   int c;
37   int d;
38 };
39
40 // Methods for TwoInt
41 TwoInt::TwoInt(int a0, int b0)
42   : a(a0), b(b0) // Initialize with correct constructors
43 { // Nothing else needed
44 }
45
46 void TwoInt::Print() const   // Print the variables
47 {
48   cout << "a " << a << ", b " << b << endl;
49 }
50
51 int  TwoInt::First() const   // Return the first int
52 {
53   return a;
54 }
55
56 int  TwoInt::Second() const  // Return the second int
```

Program subclassing.cc

```
57  {
58    return b;
59  }
60
61  int  TwoInt::Sum() const  // Return the sum
62  {
63    return a+b;
64  }
65
66  // Methods for FourInt
67  // Define the FourInt constructor, with four integers
68  FourInt::FourInt(int a0, int b0, int c0, int d0)
69    : TwoInt(a0, b0), c(c0), d(d0)
70  { // Nothing else needed
71  }
72
73  void FourInt::Print() const // Print the variables
74  {
75    // There are two ways to do this; first is to refer to superclass (TwoInt)
76    // functions get get member variables. This is NOT the preferred method.
77    cout << "a " << First() << ", b " << Second()
78         << "c " << c << ", d " << d << endl;
79
80    // Below is a better method.
81    TwoInt::Print();  // Let TwoInt "Print Itself"
82    cout << "c " << c << ", d " << d << endl;
83  }
84
85  int  FourInt::Third() const     // Get the third int
86  {
87    return c;
88  }
89
90  int  FourInt::Fourth() const    // Get the fourth int
91  {
92    return d;
93  }
94
95  double FourInt::Average() const // Compute average
96  {
97    return (Sum() + c + d) / 4.0;
98  }
99
100 void Sub1Ref(const TwoInt& ti)
101 { // Sub1Ref expects a "TwoInt" reference as a paramenter.
102   // We can pass any object of type TwoInt or any subclass of TwoInt
103   // We can call any TwoInt function,
104   cout << "Hello from Sub1Ref()" << endl;
105   ti.Print();
106   // But cannot call FourInt functions
107   // ti.Average();  // Won't compile
108 }
109
110 void Sub1Ptr(TwoInt* ti)
111 { // Sub1Ptr expects a "TwoInt" pointer as a paramenter.
112   // We can pass any object of type TwoInt or any subclass of TwoInt
```

Program subclassing.cc (continued)

```
113     // We can call any TwoInt function,
114     cout << "Hello from Sub1Ptr()" << endl;
115     ti->Print(); // Note different syntax from Sub1Ref() above
116     // But cannot call FourInt functions
117     // ti.Average();  // Won't compile
118   }
119
120   void Sub1Value(TwoInt ti)
121   { // Sub1Value expects a TwoInt BY VALUE.  Although this appears similar
122     // to the two examples above, it is quite a bit different.  This will
123     // become apparent when we discuss virtual functions.
124     cout << "Hello from Sub1Value()" << endl;
125     ti.Print();
126     // But cannot call FourInt functions
127     // ti.Average();  // Won't compile
128   };
129
130   int main()
131   {
132     TwoInt ti1(1, 2);
133     TwoInt ti2(2, 4);
134     FourInt fi1(10, 11, 12, 13);
135     FourInt fi2(fi1);
136     // We can call the Sub1 variants passing "TwoInt" objects as parameters
137     // either by reference, by pointer, or by value
138     cout << "Calling Sub1Ref() passing TwoInt" << endl;
139     Sub1Ref(ti1);
140     cout << "Calling Sub1Ptr() passing TwoInt" << endl;
141     Sub1Ptr(&ti2);
142     cout << "Calling Sub1Value() passing TwoInt" << endl;
143     Sub1Value(ti1);
144
145     // Note we can pass any subclass of TwoInt to the Sub1's
146     cout << "Calling Sub1Ref() passing FourInt" << endl;
147     Sub1Ref(fi1);
148     cout << "Calling Sub1Ptr() passing FourInt" << endl;
149     Sub1Ptr(&fi2);
150     // What happens when we pass a FourInt to Sub1Value?
151     cout << "Calling Sub1Value() passing FourInt" << endl;
152     Sub1Value(fi1);
153   }
154
155
156
157
```

Program subclassing.cc (continued)