```
 1: // $Id: inheritance2.cpp,v 1.10 2015-05-14 17:45:38-07 - - $
 2:
 3: //
 4: // Example using objects, with a base object and two derived objects.
 5: // Similar to inheritance2, but uses gcc demangler.
 6: //
 7:
 8: #include <iostream>
 9: #include <memory>
10: #include <typeinfo>
11: #include <vector>
12: using namespace std;
13:
14: #define LOG cout << __func__ << "[" << __LINE__ << "]: "
15:
16: #include <cxxabi.h>
17: template <typename type>
18: string demangle_typeid (const type& object) {
19:    const char* name = typeid(object).name();
20:    int status = 0;
21:    using deleter = void (*) (void*);
22:    unique_ptr<char,deleter> result {
23:       abi::__cxa_demangle (name, nullptr, nullptr, &status),
24:       std::free,
25:    };
26:    return status == 0 ? result.get() : name;
27: }
28:
```

```
29:
30: //////////////////////////////////////////////////////////////////
31: // class object
32: //////////////////////////////////////////////////////////////////
33:
34: class object {
35:    private:
36:       object (const object&) = delete;
37:       object& operator= (const object&) = delete;
38:       object (object&&) = delete;
39:       object& operator= (object&&) = delete;
40:       static unsigned next_id;;
41:    protected:
42:       const unsigned id;
43:       object(); // abstract class, so only derived can used ctor.
44:    public:
45:       virtual ~object(); // must be virtual
46:       virtual void print (ostream&) const;
47: };
48:
49: ostream& operator<< (ostream& out, const object& obj) {
50:    obj.print (out);
51:    return out;
52: }
53:
54: unsigned object::next_id = 0;
55:
56: object::object(): id(++next_id) {
57:    LOG << "Create: " << *this << endl;
58: }
59:
60: object::~object() {
61:    LOG << "Delete: " << *this << endl;
62: }
63:
64: void object::print (ostream& out) const {
65:    out << "[" << static_cast<const void *const> (this) << "->"
66:        << demangle_typeid(*this) << "] id=" << id << ": ";
67: }
68:
```

```
 69:
 70: //////////////////////////////////////////////////////////////////
 71: // class square
 72: //////////////////////////////////////////////////////////////////
 73:
 74: class square: public object {
 75:    private:
 76:       size_t width;
 77:    public:
 78:       square (size_t width = 0);
 79:       virtual ~square();
 80:       virtual void print (ostream&) const;
 81: };
 82:
 83: square::square (size_t width): width(width) {
 84:    LOG << "Create: " << *this << endl;
 85: }
 86:
 87: square::~square() {
 88:    LOG << "Delete: " << *this << endl;
 89: }
 90:
 91: void square::print (ostream& out) const {
 92:    this->object::print (out);
 93:    out << "square: width=" << width;
 94: }
 95:
 96: //////////////////////////////////////////////////////////////////
 97: // class circle
 98: //////////////////////////////////////////////////////////////////
 99:
100: class circle: public object {
101:    private:
102:       size_t diameter;
103:    public:
104:       circle (size_t diameter = 0);
105:       virtual ~circle();
106:       virtual void print (ostream&) const;
107: };
108:
109: circle::circle (size_t diameter): diameter(diameter) {
110:    LOG << "Create: " << *this << endl;
111: }
112:
113: circle::~circle() {
114:    LOG << "Delete: " << *this << endl;
115: }
116:
117: void circle::print (ostream& out) const {
118:    this->object::print (out);
119:    out << "circle: " << "diameter=" << diameter;
120: }
121:
122:
```

```
123:
124: //////////////////////////////////////////////////////////////////
125: // main
126: //////////////////////////////////////////////////////////////////
127:
128: int main() {
129:    LOG << "sizeof (object) = " << sizeof (object) << endl;
130:    LOG << "sizeof (square) = " << sizeof (square) << endl;
131:    LOG << "sizeof (circle) = " << sizeof (circle) << endl;
132:
133:    vector<shared_ptr<object>> vec;
134:    // ERROR: v.push_back (new object());
135:    // ERROR: object o;
136:    vec.push_back (shared_ptr<object> (new circle (  )));
137:    vec.push_back (shared_ptr<object> (new circle (10)));
138:    vec.push_back (shared_ptr<object> (new square (  )));
139:    vec.push_back (shared_ptr<object> (new square ( 5)));
140:    vec.push_back (shared_ptr<object> (new square ( 8)));
141:    cout << endl;
142:
143:    for (const auto& ptr: vec) {
144:       LOG << "Object: " << *ptr << endl;
145:    }
146:    cout << endl;
147:
148:    LOG << "return 0" << endl;
149:    return 0;
150: }
151:
152: /*
153: //TEST// valgrind --leak-check=full --show-reachable=yes \
154: //TEST//       inheritance2 >inheritance2.out 2>&1
155: //TEST// mkpspdf inheritance2.ps inheritance2.cpp* inheritance2.out*
156: */
157:
```

```
1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting inheritance2.cpp
2: inheritance2.cpp:
3:      $Id: inheritance2.cpp,v 1.10 2015-05-14 17:45:38-07 - - $
4: g++ -g -O0 -Wall -Wextra -rdynamic -std=gnu++11 inheritance2.cpp
5:          -o inheritance2 -lglut -lGLU -lGL -lX11 -lrt -lm
6: rm -f inheritance2.o
7: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished inheritance2.cpp
```

```
    1: ==20979== Memcheck, a memory error detector
    2: ==20979== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al
.
    3: ==20979== Using Valgrind-3.9.0 and LibVEX; rerun with -h for copyright i
nfo
    4: ==20979== Command: inheritance2
    5: ==20979==
    6: main[129]: sizeof (object) = 16
    7: main[130]: sizeof (square) = 24
    8: main[131]: sizeof (circle) = 24
    9: object[57]: Create: [0x4e7d090->object] id=1:
   10: circle[110]: Create: [0x4e7d090->circle] id=1: circle: diameter=0
   11: object[57]: Create: [0x4e7d300->object] id=2:
   12: circle[110]: Create: [0x4e7d300->circle] id=2: circle: diameter=10
   13: object[57]: Create: [0x4e7d580->object] id=3:
   14: square[84]: Create: [0x4e7d580->square] id=3: square: width=0
   15: object[57]: Create: [0x4e7d820->object] id=4:
   16: square[84]: Create: [0x4e7d820->square] id=4: square: width=5
   17: object[57]: Create: [0x4e7da40->object] id=5:
   18: square[84]: Create: [0x4e7da40->square] id=5: square: width=8
   19:
   20: main[144]: Object: [0x4e7d090->circle] id=1: circle: diameter=0
   21: main[144]: Object: [0x4e7d300->circle] id=2: circle: diameter=10
   22: main[144]: Object: [0x4e7d580->square] id=3: square: width=0
   23: main[144]: Object: [0x4e7d820->square] id=4: square: width=5
   24: main[144]: Object: [0x4e7da40->square] id=5: square: width=8
   25:
   26: main[148]: return 0
   27: ~circle[114]: Delete: [0x4e7d090->circle] id=1: circle: diameter=0
   28: ~object[61]: Delete: [0x4e7d090->object] id=1:
   29: ~circle[114]: Delete: [0x4e7d300->circle] id=2: circle: diameter=10
   30: ~object[61]: Delete: [0x4e7d300->object] id=2:
   31: ~square[88]: Delete: [0x4e7d580->square] id=3: square: width=0
   32: ~object[61]: Delete: [0x4e7d580->object] id=3:
   33: ~square[88]: Delete: [0x4e7d820->square] id=4: square: width=5
   34: ~object[61]: Delete: [0x4e7d820->object] id=4:
   35: ~square[88]: Delete: [0x4e7da40->square] id=5: square: width=8
   36: ~object[61]: Delete: [0x4e7da40->object] id=5:
   37: ==20979==
   38: ==20979== HEAP SUMMARY:
   39: ==20979==     in use at exit: 0 bytes in 0 blocks
   40: ==20979==   total heap usage: 65 allocs, 65 frees, 1,464 bytes allocated
   41: ==20979==
   42: ==20979== All heap blocks were freed -- no leaks are possible
   43: ==20979==
   44: ==20979== For counts of detected and suppressed errors, rerun with: -v
   45: ==20979== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```