Objects and Connections

Rushikesh K Joshi

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Abstract Classes
Abstract Classes

A generic Component: Behavior undefined

class Component {
    public:
        virtual Pinset trigger (Pinset p)=0;
};
Properties of Abstract Classes

- Cannot instantiate this class, since it is abstract (not fully implemented)
- Notice the virtual function which is defined to be nil (i.e. 0), This makes it abstract!
- Two main ways to instantiate (but both are not permitted on class Component):
  - Component c;
  - Component *cp = new Component()
class Pinset {
    public:
    bool *pin;
    int count;
    Pinset(int k) {
        count = k;
        pin = new bool [k];
    }
};
class AND : public Component {
  public:
    virtual Pinset trigger(Pinset p);
};

Pinset AND::trigger(Pinset in) {
  Pinset out(1);
  if(in.pin[0]) out.pin[0]=in.pin[1];
  else out.pin[0]=false;
  return out;
};
class NOT : public Component {
public:
    virtual Pinset trigger(Pinset p);
};

Pinset NOT::trigger(Pinset in) {
    Pinset out(1);
    if(in.pin[0]) out.pin[0]=false;
    else out.pin[0]=true;

    return out;
};
A Composite Class which uses many objects of other classes

class NAND: public Component {
public:
    virtual Pinset trigger(Pinset p);
};

Pinset NAND::trigger(Pinset in) {
    AND g1;
    NOT g2;
    Pinset out1(1);
    Pinset out2(1);
    out1 = g1.trigger(in);
    out2 = g2.trigger(out1);
    return out2;
};
The Benefit of Abstract Classes: Generic Code

```cpp
int main (int argc, char* argv[]) {
    Component *cct;
    cout << "select circuit: XOR (0), NOR (1), NAND (2)";
    int choice;
    cin >> choice;
    switch (choice) {
        case 0: cct = new XOR(); break;
        case 1: cct = new NOR(); break;
        case 2: cct = new NAND(); break;
    }
    cout << "give two inputs from set (0 0, 1 1, 1 0, 0 1)";
    Pinset in(2), out(1) ;
    cin>>in.pin[0]; cin>>in.pin[1];
    out =cct->trigger(in) ;
    cout << out.pin[0] « endl;
}
```