**Why use inheritance?**

- We want to program to an *interface* (an abstraction, a concept)
  - The interface may be concretely implemented in different ways, consider stream hierarchy
    ```cpp
    void readStuff(istream& input){...}
    // call function
    ifstream input("data.txt");
    readStuff(input);
    readStuff(cin);
    ```
- What about new kinds of streams, ok to use?
  - Open/closed principle of code development
    - Code should be open to extension, closed to modification
    - Why is this (usually) a good idea?

**Two examples**

- Consider the expression example (expression.h.cpp)
  - What do we need to do to add a Multiplication class?
    -
  - What code must be modified vs. extended?
- Consider the RSG assignment
  - Expansion of a grammar element results in printing
    - Terminal, how to expand?
    - Nonterminal?
    - Production?
    - Definition?
  - Focus on one thing at a time, what about adding a new class called Grammar, the whole thing?

**What is an SDmap? A Definition?**

- Maps keys (strings) to values (definition pointers)
  - See the code in mapcount.cpp
    ```cpp
    while (input >> w) // read string
    {
      Definition * d = map->get(w); // look it up
      if (d == 0)
      {
        map->insert(w,new Definition()); // not found, store
      }
      else
      {
        d->incr(); // found, bump count
      }
    }
    ```

**What is a GrammarElement?**

- Part of a grammar? Useful?
  - In grammar-part below, what are the different parts?
    ```cpp
    {
      <dubious-excuse> my <person> doesn't like you ;I'm in love with <another> ;
      I haven't told you this before but <harsh> ;
    }
    ```
  - What is a Production?
    - What can a production do? What is its state?
    - Example above, then generalize
    - Where does this fit into the code in rsg.cpp?