Virtual functions and Inheritance
Virtual functions

- Pure virtual functions
- Simple virtual functions

```cpp
class Shape {
public:
    virtual void draw() const = 0;
    virtual void error(const string &msg);
    ...
};

class Rectangle: public Shape {
    ...
};
```
Virtual functions

• The purpose of a pure virtual function is to have derived classes inherit a function interface only

• The purpose of a simple virtual function is to have derived classes inherit a function interface as well as a default implementation
Non-virtual functions

• The purpose of a non-virtual function is to have derived classes inherit a function interface as well as a mandatory implementation
C++ Tip

• Never redefine an inherited non-virtual function
  – violates “is-a” inheritance rule
  – can lead to programming errors

```cpp
class B {
public:
  void mf();
};

class D: public B {
};

d D x; // x is an object of type D
b *pB = &x; // pB is pointer to x
pB->mf(); // call mf through pointer
D *pD = &x; // pD is pointer to x
pD->mf(); // call mf through pointer
```
class D: public B {
    public:
        void mf();
};

pB->mf();        // calls B::mf
pD->mf();        // calls D::mf

• Non-virtual functions are statically bound (at compile time)
• Virtual functions are dynamically bound (at runtime)