Polymorphism

Polymorphism

- From Greek πολύς, polys, "many, much" and μορφή, morphē, "form, shape."

- ***The ability for a derived class to substitute in code where a base class is used.***

- This concept is not new:

```
void f(double x) {
  /* do something */;
}

int main() {
  int y = 3;
  f(y);
}
```

C++ will automatically convert a derived class object to a base class object when required.

Typical situations:

- Variable assignment
- Calling a function

Caveat emptor

- When C++ automatically converts a derived-class object to a base-class object, the converted object loses all extra abilities the derived class had.

```
class A {
  public:
  void f() { cout << "base f"; }
};
class B : public A {
  public:
  void f() { cout << "derived f"; }
  void g() { cout << "derived g"; }
};
int main() {
  A a;  a.f();
  B b;  b.f();  b.g();
  A copy = b;  copy.f();  copy.g();
}
```

Caveat emptor

- When C++ automatically converts a derived-class object to a base-class object, the converted object loses all extra abilities the derived class had.
- **Copying** the derived-class object into a base-class object means the copy only has the abilities of the base class.
- **How do we avoid making copies?**

Step 1: Use Pointers

- ***A base-class pointer can point to a derived-class object.***
- Because no copy is made, the pointer still points at an object that has all the abilities of the derived class.
- The base-class pointer will still only let you (directly) call functionality specified by the base class.

Step 2: Use virtual methods

- Class methods can be tagged with the keyword "virtual."
- When a virtual method is called using a pointer, C++ uses the version of the method that belongs to **the type of the object being pointed at**, not **the type of the pointer**.