ANNOUNCEMENTS

• Assignment 2 released (due Monday June 17\textsuperscript{th} @ 5:00pm)
There are three types of relationships between classes which we typically discuss:

**Composition**(owns-a): class A *owns an* instance of class B. This means that class A is responsible for deleting the instance of class B when an object of class A is destroyed.

* Under composition, instances of B cannot be shared.
–**Aggregation (has-a):** class A *has an* instance of class B. This means that class A is not responsible for deleting the instance of class B.
  • ∗if an object of class A is deleted, the instance of class B associated to it lives on.
  • ∗multiple objects of class A can have the same instance of class B.

–**Inheritance (is-a):** class B *is a* class A. This means that an instance of class B can be used in any situation where an instance of class A can be used.
  • **Note:**
    • ∗the converse is not true. That is, an instance of class A cannot always be used where an instance of class B can be used.
    • ∗for this course, we are mainly concerned with public inheritances.
**Note:** If a class A has a pointer to an instance of class B, you cannot know if the relationship is composition or aggregation without looking at documentation.

```c
class B {
    ...
};

class A {
    B b; // This is composition
    B *b2; // This could be composition or aggregation
};
```
INHERITANCE

Example:

```cpp
class A {
    int a;

public:
    A(int a) : a{a} {}
};

class B: public A { int b;

public:
    B(int a, int b) : A{a}, b{b} {}
};
```
• In this example, B inherits from A (this is what the “: public A” is for). This means that every instance of B has the fields and methods which an instance of A has.
• Note the constructor for the B class. The first element of the MIL is A{a} which is calling the constructor for the A portion of the B object.
ENCAPSULATION AND INHERITANCE

• If A has fields which are private, B cannot access these fields (as they are private).
• What are some benefits of an inherited class not having direct access to the fields of the superclass?
  – Other people may inherit from our classes and this means they’d have access to the fields of the superclass in their implementation of their class, and this breaks encapsulation.
• However, we often want to give subclasses “special access” to the class.
  – For instance, we might want to have some accessor methods so that subclasses can access fields in a way that we choose, but we don’t want to let everyone have access to these members.
• For this purpose, we can use the third type of privacy: protected.
• Members which are protected can be accessed directly by subclasses but cannot be accessed by the public.
• **Note:** Most of the time you should not make fields protected as this also breaks encapsulation.