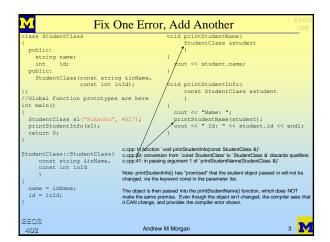
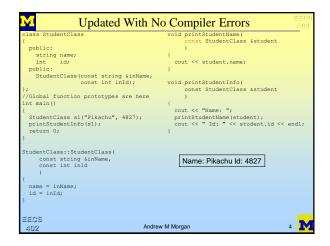
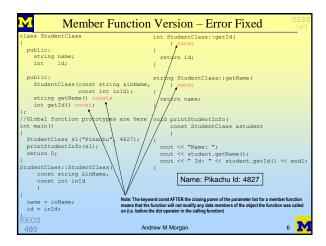


```
Compiler Error Example
                                                                         StudentClass &student
                                                                   cout << student.name;
void printStudentInfo(
    const StudentClass &student
   )
/Global function prototypes are here
nt main()
                                                                   cout << "Name: ";
printStudentName(student);
cout << " Id: " << student.id << endl;</pre>
StudentClass s1;
s1.name = "Pikachu"
s1.id = 4827;
printStudentInfo(s1);
                                             c.cpp: In function 'int main()':
c.cpp:56: no matching function for call to 'StudentClass::StudentClass()'
 return 0;
                                             Recall: Once a ctor is provided, the default ctor is no longer available (unless a default ctor is provided as an overloaded ctor)
StudentClass::StudentClass(
     const string &inNa
const int inId
ECS
                                                     Andrew M Morgan
                                                                                                                        2 1
```







```
Recall static variables in global functions

- Variable stored in global space, and exists throughout program

- Static data members in classes have a similar meaning

- A static data member only is stored in memory once per class

- Not once per object like non-static variables!

- All objects of the class, and in fact the class itself, "shares" this one instance of static data members

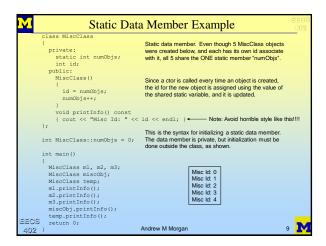
- Static data members are initialized outside of class member functions

- This is true even if static member is private!

- Static data member are only allowed to be initialized once

- Scope resolution is used to indicate a static member is being initialized

- Often used to count objects created, destroyed, or function calls
```



```
Public Static Data Members
    ass NewMiscClass
                                                                  int NewMiscClass::numObjs = 0;
    private:
                                                                   int main()
                                                                      NewMiscClass ml, m2, m3;
NewMiscClass miscObj;
NewMiscClass temp;
    static int num;
public:
        NewMiscClass()
                                                                      ml.printInfo();
        void printInfo() const
                                                                      cout << "Next ID: " << ml.num << endl;
cout << "Next ID: " << temp.num << endl;
cout << "Next ID: " << NewMiscClass::num << endl;</pre>
                                                                      return 0;
                   NewMisc Id: 0
NewMisc Id: 1
NewMisc Id: 2
NewMisc Id: 3
NewMisc Id: 4
Next ID: 5
Next ID: 5
Next ID: 5
                                                       When the static member is public, it can be accessed using and individual object.
                                                      More appropriately and commonly, the static member can be accessed using the class name and scope resolution. This makes sense, since the static member doesn't belong to an object, but rather the class or a utable.
FECS
                                                                      Andrew M Morgan
                                                                                                                                                         10
```

```
Static Member Functions

• A static member function also "belongs to a class"

• Static member functions can access private (or public) static data member variables

- A static member function can NOT access non-static data members!

• Static functions can be called using the class name and scope resolution

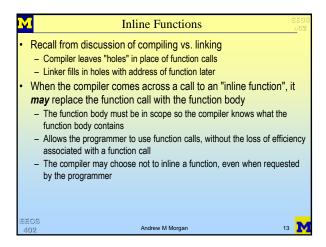
• Allows static data members to be private

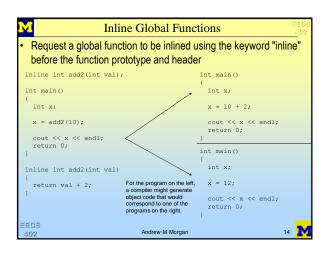
- Public data members are essentially global and should be avoided
```

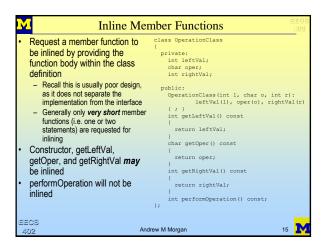
```
Static Member Function, Example

class NewMiscClass

{
    private:
        int id;
        public:
        NewMiscClass()
        int main()
        int main()
```







```
Designing Classes
What should be a class? What should not?
 - Remember, OOP is beneficial because it can be used to generate
    programs that look and work like the "real world"
 - Create classes to group data and functionality for "things" that will be
    used in the program
 - "Actions" should be designed and implemented as functions, not classes

    For example, CardClass, DeckClass, BankRollClass, etc

      · NOT DealCardClass, CalculateWinnerClass, etc
What should be member variables? What should not?
 - Member variables should be data that will describe attributes of all objects
    of the class

    Values that don't describe attributes of objects should not be members

      • For example, if you notice that every member function of a class uses a variable
        named "" as a loop variable, you may be tempted to make it a member variable, so it doesn't have be declared in every individual function. This would be a poor design, however, since the loop variable "i" does not describe an attribute of the objects of
        the class!
                                     Andrew M Morgan
```