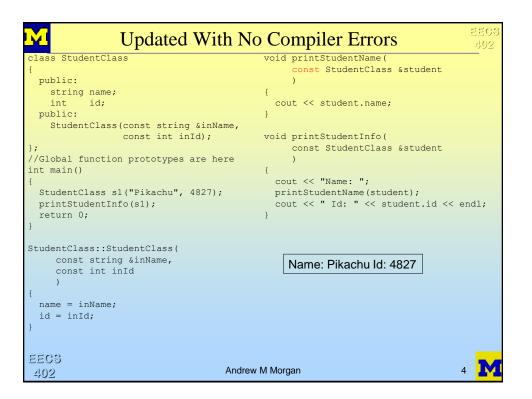


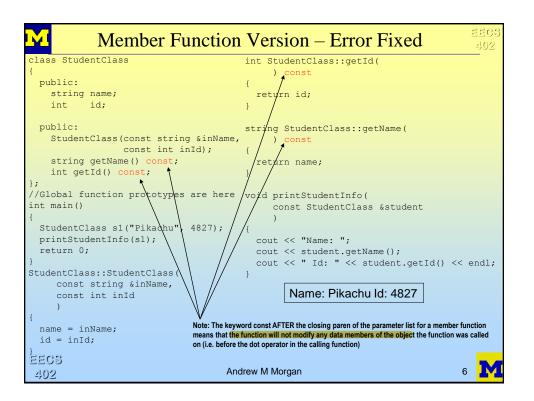
```
Compiler Error Example
  lass StudentClass
                                                void printStudentName(
                                                     StudentClass &student
  public:
    string name;
    int
                                                  cout << student.name;</pre>
    StudentClass(const string &inName,
                  const int inId);
                                                void printStudentInfo(
                                                     const StudentClass &student
//Global function prototypes are here
int main()
                                                  cout << "Name: ";
  StudentClass s1;
                                                  printStudentName(student);
  s1.name = "Pikachu";
                                                  cout << " Id: " << student.id << endl;
  s1.id = 4827;
  printStudentInfo(s1);
                                  c.cpp: In function 'int main()':
  return 0;
                                  c.cpp:56: no matching function for call to 'StudentClass::StudentClass()'
                                  Recall: Once a ctor is provided, the default ctor is no longer available (unless a
StudentClass::StudentClass(
                                  default ctor is provided as an overloaded ctor)
    const string &inName,
     const int inId
 name = inName;
id = inId;
EECS
                                        Andrew M Morgan
 402
```

why? lec7 page9

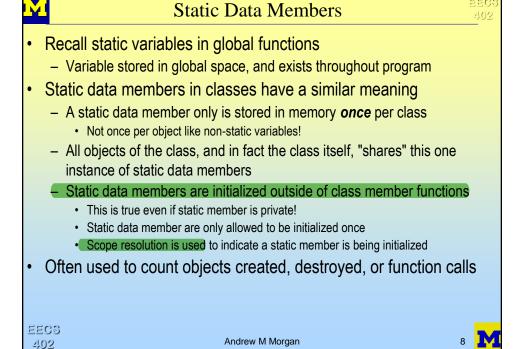
```
Fix One Error, Add Another
                                                      void printStudentName(
  lass StudentClass
                                                            StudentClass &student
  public:
     string name;
                                                          out << student.name;
  public:
     StudentClass(const string &inName,
                     const int inId);
                                                       oid printStudentInfo(
                                                            const StudentClass &student
//Global function prototypes are here
int main()
                                                        cout << "Name: ";
  StudentClass s1("Pikachu", 4827);
                                                        printStudentName(student);
                                                        cout << " Id: " << student.id << endl;
  printStudentInfo(s1);
  return 0:
                                     c.cpp: In function `void printStudentInfo(const StudentClass &)': c.cpp:$0: conversion from `const StudentClass' to `StudentClass &' discards qualifiers
StudentClass::StudentClass(
     const string &inName,
                                     c.cpp:41: in passing argument 1 of `printStudentName(StudentClass &)'
     const int inId
                                     Note: printStudentInfo() has "promised" that the student object passed in will not be
                                     changed, via the keyword const in the parameter list.
  name = inName;
                                     The object is then passed into the printStudentName() function, which does NOT
  id = inId;
                                     make the same promise. Even though the object isn't changed, the compiler sees that
                                     it CAN change, and provides the compiler error shown.
EECS
                                             Andrew M Morgan
 703
```



```
New Version Using Member Function, Error
class StudentClass
                                                    int StudentClass::getId(
  public:
     string name;
                                                       return id:
    int
             id;
                                                    string StudentClass::getName(
     StudentClass(const string &inName,
                    const int inId);
     string getName();
                                                      return name;
     int getId();
//Global function prototypes are here
                                                    void printStudentInfo(
int main()
                                                          const StudentClass &student
  StudentClass s1("Pikachu", 4827);
  printStudentInfo(s1);
                                                       cout << "Name: ";
  return 0;
                                                       cout << student.getName();</pre>
                                                       cout << " Id: " << student.id << endl;
StudentClass::StudentClass(
     const string &inName,
                                      c.cpp: In function 'void printStudentInfo(const StudentClass &)':
                                      c.cpp:50: passing 'const StudentClass' as 'this' argument of 'class string StudentClass::getName()' discards qualifiers
      const int inId
                                      Same problem as earlier, but now it's a member function that isn't making the promise
  name = inName;
                                      not to change the object! printStudentInfo() promises not to change the student object
  id = inId;
                                      passed in, but then the student object is used to call the member function getName(),
                                      which may change the data members of the object.
EECS
                                           Andrew M Morgan
703
```



```
Better All Around – Make print a Member
  class StudentClass
                                           StudentClass::StudentClass(
                                               const string &inName,
    public:
                                               const int inId
      string name;
      int
            id;
                                            name = inName;
                                            id = inId;
    public:
      StudentClass(const string &inName,
                  const int inId);
      void printInfo() const;
                                           void StudentClass::printInfo() const
  };
                                            cout << "Name: " << name <<
                                                    " Id: " << id << endl;
  int main()
    StudentClass s1("Pikachu", 4827);
    s1.printInfo();
    return 0:
                           Name: Pikachu Id: 4827
EECS
                                  Andrew M Morgan
703
```



```
Static Data Member Example
       class MiscClass
                                           Static data member. Even though 5 MiscClass objects
         private:
                                           were created below, and each has its own id associate
            static int numObjs;
                                          with it, all 5 share the ONE static member "numObjs".
            int id;
         public:
           MiscClass()
                                          Since a ctor is called every time an object is created,
                                           the id for the new object is assigned using the value of
              id = numObjs;
                                          the shared static variable, and it is updated.
              numObjs++;
            void printInfo() const
            { cout << "Misc Id: " << id << endl; } ← Note: Avoid horrible style like this!!!!
       };
                                          This is the syntax for initializing a static data member.
       int MiscClass::numObjs = 0;
                                           The data member is private, but initialization must be
                                           done outside the class, as shown.
       int main()
         MiscClass m1, m2, m3;
                                                          Misc Id: 0
         MiscClass miscObj;
                                                          Misc Id: 1
         MiscClass temp;
                                                          Misc Id: 2
         ml.printInfo();
                                                          Misc Id: 3
         m2.printInfo();
                                                          Misc Id: 4
         m3.printInfo();
         miscObj.printInfo();
         temp.printInfo();
EECS
        return 0;
402 }
                                         Andrew M Morgan
```

```
Public Static Data Members
class NewMiscClass
                                      int NewMiscClass::numObjs = 0;
  private:
                                      int main()
    int id;
  public:
                                        NewMiscClass m1, m2, m3;
                                        NewMiscClass miscObj;
    static int num;
  public:
                                        NewMiscClass temp;
    NewMiscClass()
                                        m1.printInfo();
      id = num;
                                        m2.printInfo();
                                        m3.printInfo();
      num++;
                                        miscObj.printInfo();
                                        temp.printInfo();
    void printInfo() const
                                        cout << "Next ID: " << ml.num << endl;
cout << "Next ID: " << temp.num << endl;</pre>
      cout << "NewMisc Id: " <<
                                        cout << "Next ID: " << NewMiscClass::num << endl;</pre>
               id << endl;
                                        return 0:
};
                                                                           can't we do this when its private?
             NewMisc Id: 0
                                When the static member is public, it can be accessed using and
             NewMisc Id: 1
                                individual object.
             NewMisc Id: 2
             NewMisc Id: 3
                                More appropriately and commonly, the static member can be
             NewMisc Id: 4
                                accessed using the class name and scope resolution. This makes
             Next ID: 5
                                sense, since the static member doesn't belong to an object, but rather
             Next ID: 5
                                the class as a whole.
EECS
             Next ID: 5
                                        Andrew M Morgan
402
```



Static Member Functions

芝三〇 402

- 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

402 402

Andrew M Morgan

P

```
Static Member Function, Example
                                         int NewMiscClass::numObjs = 0;
 class NewMiscClass
     int id;
     static int numObjs;
                                           NewMiscClass m1, m2, m3;
                                           NewMiscClass miscObj;
   public:
     NewMiscClass()
                                          NewMiscClass temp;
                                          m1.printInfo();
                                          m2.printInfo();
       numObjs++;
                                          m3.printInfo();
                                          miscObj.printInfo();
     void printInfo() const
                                           temp.printInfo();
       cout << "NewMisc Id: " <<
                                          //cout << "Next ID: " << m1.numObjs << endl;
                                          //cout << "Next ID: " << temp.numObjs << endl;
//cout << "Next ID: " << NewMiscClass::numObjs << endl;
                                           m1.printNumObjects();
     static void printNumObjects()
                                          temp.printNumObjects();
                                          NewMiscClass::printNumObjects();
       cout << "Num Objs: " <<
                                          return 0;
              numObjs << endl;
             NewMisc Id: 0
             NewMisc Id: 1
             NewMisc Id: 2
                                Commented lines would now cause compiler errors (since numObjs is
             NewMisc Id: 3
                                 private). Can access via the static member function, however.
             NewMisc Id: 4
             Next ID: 5
             Next ID: 5
EECS
             Next ID: 5
                                          Andrew M Morgan
                                                                                            12
402
```



Inline Functions

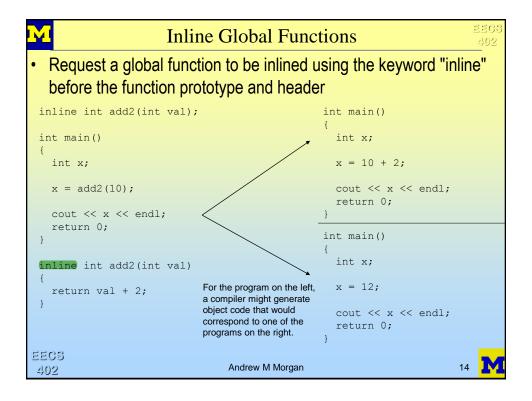
로**르**08

- Recall from discussion of compiling vs. linking
 - Compiler leaves "holes" in place of function calls
 - Linker fills in holes with address of function later
- When the compiler comes across a call to an "inline function", it may replace the function call with the function body
 - The function body must be in scope so the compiler knows what the function body contains
 - Allows the programmer to use function calls, without the loss of efficiency associated with a function call
 - The compiler may choose not to inline a function, even when requested by the programmer

EECS 402

Andrew M Morgan

M



```
Inline Member Functions
                                        class OperationClass
   Request a member function to
   be inlined by providing the
                                          private:
                                            int leftVal;
   function body within the class
                                            char oper;
   definition
                                            int rightVal;

    Recall this is usually poor design,

                                          public:
       as it does not separate the
                                            OperationClass(int 1, char o, int r):
                                                    leftVal(1), oper(o), rightVal(r)
       implementation from the interface

    Generally only very short member

                                            int getLeftVal() const
       functions (i.e. one or two
                                              return leftVal;
       statements) are requested for
       inlining
                                            char getOper() const
  Constructor, getLeftVal,
                                              return oper;
   getOper, and getRightVal may
   be inlined
                                            int getRightVal() const
  performOperation will not be
                                              return rightVal;
   inlined
                                            int performOperation() const;
                                        };
EECS
                                    Andrew M Morgan
 703
```

```
Inline Member Functions Example Continued
          OperationClass op1(5, '+', 2);
OperationClass op2(8, '-', 2);
          cout << opl.getLeftVal() << opl.getOper() << opl.getRightVal() <<</pre>
                    "=" << opl.performOperation() << endl;
          cout << op2.getLeftVal() << op2.getOper() << op2.getRightVal() <<
    "=" << op2.performOperation() << endl;</pre>
        int OperationClass::performOperation(
              ) const
          int result;
          if (oper == '+')
                                                                    5+2=7
                                                                    8-2=6
             result = leftVal + rightVal;
          else if (oper == '-')
             result = leftVal - rightVal;
          return result;
EECS
                                       Andrew M Morgan
402
```



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 "i" 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!

EECS 402

Andrew M Morgan



