Lab 6

Classes!

Classes!!

My favorite topic ☺

Definition of a class

- How many different definitions can we come up with?

What is a class?

- A class is a container that can hold **different** types of objects (objects with different data types)

- What is another type of container we've learned, where all objects must be the **same data type**?

What is a class?

- A class is a container that can hold **different** types of objects (objects with different data types)

- What is another type of container we've learned, where all objects must be the **same data type**? An Array

What is a class?

- A class is a container that can hold **different** types of objects (objects with different data types)
- A class is a user-defined data type
  - Just like `int` and `double` and `string` are data types, so is the class you define
- A class is a way to group together related information

How to think about classes

- I like to think of a class as an object that holds multiple pieces of information
  - You pass around a box that holds some more information inside it
  - If you have access to the box, you also have access to what's inside it

How to think about classes

- Most of you have a backpack with a computer and notebook inside
  - So far in EECS183 you could hold a notebook in one variable, and a computer in the other
  - But this means you you would have to carry the computer and notebook separately
  - But most likely you'd want a backpack to store them in
  - **The backpack is the class – you create an object to store related items together**
    - **By 'opening' the backpack you can get access to the computer and notebook**

A Student

- What are some defining characteristics of a student?

A Student

- What are some defining characteristics of a student?
  - uniqname
  - UMID

Let's create a Student class

- What are some defining characteristics of a student?
  - uniqname
  - UMID
- A student has all of these attributes, so we could make a Student class, where these attributes are member variables

Let's make a Student class

- What are some defining characteristics of a student?
  - uniqname
  - UMID
- A student has these attributes, so we could make a Student class, where these attributes are member variables
- This allows us to declare a Student (our custom data type) that holds attributes of the student as well

Student class

```
class Student{
      //what goes in here?




};
```

Student class

```
class Student{

private:
	string uniqname;
	int UMID;

public:
	Student();
	Student(string name_in);
	string getUniqname();
	int getUMID();
	void setUMID(int UMID_in);
};
```

What are each of these?

Student class

```
class Student{

private:
        string uniqname;                private member variable
        int UMID;                       private member variable

public:
        Student();                      default constructor
        Student(string name_in);        non-default constructor
        string getUniqname();           getter to get name
        int getUMID();                  getter to get umid
        void setUMID(int UMID_in);      setter to set umid
};
```

Public vs Private

- A private member variable or member function means that only members of the class can access it
- Member variables are usually private, and we use setters and getters to access them
- A public member variable or function means that any part of the code can access that function or variable. Setters and getters are always public.

How do we know that a member function is a constructor?

- There are 3 ways to tell
  - 1) **The function name is exactly the same as the class name**
  - 2) The function **has no return type**
  - 3) The function starts with a capital letter (this is the standard)
- How do we know which type of constructor it is?
  - If it has no parameters (e.g., `Student()`) then it is a **default constructor**
  - If is has parameters then it is a **non-default constructor**

Declaring an object of type Student

How would we declare and initialize this Student object?

```
int main(){




}
```

Declaring an object of type Student

How would we declare and initialize this Student object?

```
int main(){
      Student student1 = Student();
      Student s2 = Student("Johnny");
      s2.setUMID(12345678);
}
```

Why couldn't we say: `s2.UMID = 12345678;` ???

Declaring an object of type Student

How would we declare and initialize this Student object?

```
int main(){
      Student student1 = Student();
      Student s2 = Student("Johnny");
      s2.setUMID(12345678);
}
```

Why couldn't we say: **`s2.UMID = 12345678;`** ???

Because `UMID` is a `private` member variable

Header files vs source files

- Header files have the `.h` extension
- Source files have the `.cpp` extension

- Header files are where class definitions and function declarations go
- Source files are where function implementations go

In the Header file (.h)

- Some member functions
  - including a print function
- Public and private members
- A default constructor
- A non-default constructor
- Getters
- Setters

.cpp file

- Has all the implementation of the declaration in the header file
- Be careful with syntax here!!!

What goes in a **header** file?

```
class Student{
      //class definition
      //as defined a few slides ago
};
```

What goes in the **source** file

- Member function implementations
- Don't forget the scope resolution operator ::
  - *Student*::_____
    - says "the following function is a member function of *the Student class*"

What goes in the source file

```
#include "Student.h"
Student::Student() {…}
Student::Student(string name_in) {…}

string Student::getUniqname() {
…
}
int Student::getUMID() {
…
}
void Student::setUMID(int UMID_in) {
…
}
```

Member variable Scope

```
string Student::getUniqname() {
    return uniqname;
}
```

- Member variables can be accessed just by name in scope of class's member functions
- Outside of class:
  - Use member functions (public or private variables)
  - Or dot operator (public variables only!)

Member variable Scope

```
void Student::setUMID(int UMID_in){
    UMID = UMID_in;
    return;
}
```

- Member variables can be accessed just by name in scope of class's member functions
- Outside of class:
  - Use member functions (public or private variables)
  - Or dot operator (public variables only!)

Lab

- Two parts to the lab for today
- Exam practice
  - Answer the question in lab6.pdf
  - Write the answer with paper and pencil!
- Practice using classes
  - Point.cpp from Project 4

Exam Practice

- Write your answer to the question just like you will on the exam
- Take at most ten minutes to solve the problem. Remember you have only 90 minutes for the upcoming exam
- After 10 minutes, we will show the answer and discuss the solution
- You will not submit your exam practice answer, but keep it to help you review!

Exam Practice – Solution

```
// get file name
cout << "Filename: ";
string filename;

// both cin >> and getline are okay
getline(cin, filename);

// declare and open file
ifstream inputFileStream;
inputFileStream.open(filename);

int wordcount = 0;
string word;
while (inputFileStream >> word) {
    wordcount += 1;
}

cout << "Word count: " << wordcount << endl;
inputFileStream.close();
```

Classes Exercise

- Now complete the programming section of the lab assignment
- Feel free to ask any questions you have
- To receive the 8 points for the lab, you will need to submit:
  - Point.cpp to the autograder (6 points)
  - Test.cpp to the autograder (2 points)
  - The grade from the autograder is your grade for the lab.