CS 162
Intro to CS II
Classes
Odds and Ends...

• Assignment 2 questions
• ACM-W meets tonight, Wed.  
  – 5:30-6:30pm KEC 1005
• ACM meets tomorrow, Thurs.  
  – 6-7pm KEC 1005
Review Patron/Library classes...

```cpp
#include <string>
using namespace std;

class patron {
  private:
    //direct initialization of members in class declaration only 2011
    //standard and above
    string name = "test";
  public:
    patron(string); //non-default constructor
    string get_name() const;
    void set_name(const string &);
};
```
Review Patron/Library classes...

```cpp
#include "patron.h"

class library {
    private:
        // direct initialization of object in class only allowed in 2011
        // standard and above, and it doesn't have the patron p("jen")
        // format as it does outside the class as a non-member
        patron p=patron("jen");

    public:
        library(); //default constructor
        patron get_patron() const;
        void set_patron(const patron &);
};
```
Review Patron/Library classes...
What is wrong with our library
get_patron() accessor???
Return a Reference to the patron, p

```cpp
#include "patron.h"

class library {
    private:
        //direct initialization of object in class only allowed in 2011
        //standard and above, and it doesn't have the patron p("jen")
        //format as it does outside the class as a non-member
        patron p=patron("jen");

    public:
        library(); //default constructor
        //need to return a reference to a patron to not get a clone and
        //refer to the patron in the library class outside the class
        //However, this means get_patreon cannot be const anymore because
        //you are using the function to change a member indirectly outside
        //the function!!!
        patron & get_patreon();
        void set_patreon(const patron &);
};
```
Return a pointer to the patron, p.

```cpp
#include "patron.h"

class library {
    private:
    // Direct initialization of object in class only allowed in 2011
    // standard and above, and it doesn't have the patron p("jen")
    // format as it does outside the class as a non-member
    patron p=patron("jen");

    public:
    library(); // default constructor
    // You can return pointer to a patron to not get a clone and
    // refer to the patron in the library class outside the class
    // However, this means get_patron cannot be const anymore because
    // you are using the function to change a member indirectly outside
    // the function!!!
    patron * get_patron();
    void set_patron(const patron &);
};
```
Then you need to return “address of”
If we return the address of a patron and have a pointer, then we need -> to dereference.
Passing an object by value calls copy constructor, which also is destructed.

```cpp
#include "library.h"
#include "patron.h"
#include <iostream>
using namespace std;

//copy constructor used to pass object by value, and destructor called
//when object goes out of scope

void fun(library l) {

}

int main() {
    patron p("jen");

    cout << p.get_name() << endl;
    p.set_name("jennifer");
    cout << p.get_name() << endl;

    library l;
    fun(l); //call the copy constructor to pass by value
    cout << l.get_patron()->get_name() << endl;
    l.get_patron()->set_name("jennifer");
    cout << l.get_patron()->get_name() << endl;

    return 0;
}
```
Create a copy constructor & destructor

```cpp
#include "patron.h"

class library {
    private:
        //direct initialization of object in class only allowed in 2011
        //standard and above, and it doesn't have the patron p("jen")
        //format as it does outside the class as a non-member
        patron p=patron("jen");
    public:
        library(); //default constructor
        library(const library &); //copy constructor
        ~library(); //destructor
        patron * get_patron();
        void set_patron(const patron &);
};
```
Create a copy constructor & destructor

```
#include "library.h"
#include <iostream>
using namespace std;

// need to call patron's non-default constructor before
// the curly brace because that is when the object is
// created
library::library() : p("jen") {
    // can't do this because p would have already been created
    // with the default constructor, & we don't have a default
    // p=patron("jen");
}

// copy constructor takes argument of same type as the object being
// constructed
library::library(const library &l) {
    cout << "Library object copy constructor!" << endl;
}

// destructor created with ~ in front of class name
library::~library() {
    cout << "Library object being destructed!" << endl;
}

patron * library::get_patron() {
    return &p;
}
```
Copy constructor and destructor

```bash
flp1 ~/cs162/private/sec1 175% a.out
jen
jennifer
Library object copy constructor!
Library object being destructed!
jen
jennifer
Library object being destructed!
flp1 ~/cs162/private/sec1 176%
```
The Big “Three”

• If dynamic memory allocation in class, then...
  – Destructor
  – Copy Constructor
  – Assignment operator overload
What is a Destructor?

• Deallocate any member variable dynamically allocated...

• What would this destructor look like then?

```cpp
string::~string() {
    delete [] s; //delete ignores NULL
}
```
What is a copy constructor?

- Used in pass by value
- Returning an object from a function
- Pass the class type to a constructor

```cpp
string:: string(const string &other) {
    len=other.len;
    if(len == 0) s=NULL;
    else {
        s=new char[len];
        for(int i=0; i<len; i++)
            s[i] = other.s[i];
    }
}
```