CS 32: Discussion 1D

TA: Shichang Zhang
LA: Stephanie Doan, Rish Jain

About Us

Discussion:

- Discussion 1D: 2:00 - 3:50 PM PST, Friday
- Discussion will be recorded and uploaded on CCLE

Office Hours:

- Shichang: Wednesday 3:30-5:30PM PST, Thursday 10:30-12:30PM PST
- Rish: Tuesdays 2:30-4:30PM PST
- Stephanie: Thursdays 11:30AM-1:30PM PST

Email:

- Shichang: shichang@cs.ucla.edu
- Rish: rishabjain@g.ucla.edu
- Stephanie: stephaniekdoan@gmail.com

Course Website:

- http://web.cs.ucla.edu/classes/spring21/cs32/

Announcements

- Homework 1 due Tuesday(April 13th)
- Linked List LA Workshop Wednesday, 04/14 7-8 pm

Overview

- Order of Construction & Destructor
- Copy Constructor
- Assignment Operator

Order of Construction & Destruction

A *Class* is constructed in this order:

1. Construct the member variables
2. Construct member functions
3. Invoke the constructor body

A *Class* is destructed in the *opposite* order:

1. Execute body of destructor
2. Destruct member variables in reverse order they were constructed

NOTE: If the member variable is another class, pause everything else and construct that class first.

What do you expect the output to be?

```
class LA
{
  public:
  LA(){cout << "Hi!" << endl;}
  ~LA()
  {cout << "peace" << endl;}
};

class TA
{
  public:
  TA()
  { cout << "Shichang's here" << endl;}
  LA Rish;
  LA Stephanie;
  ~TA() {cout << "Shichang's out!" << endl;}
};
```

```
class Smallberg
{
  public:
  Smallberg():age(20)
  {cout << "Smallberg's here!" << endl;}
  int age;
  TA s;
  ~Smallberg()
  {cout << "Smallberg's out!" << endl;}
};

int main() {
  Smallberg s;
  cout << s.age << endl;
}
```

What do you expect the output to be?

```
class LA
{
  public:
  LA(){cout << "Hi!" << endl;}
  ~LA()
  {cout << "peace" << endl;}
};

class TA
{
  public:
  TA()
  { cout << "Shichang's here" << endl;}
  LA Rish;
  LA Stephanie;
  ~TA() {cout << "Shichang's out!" << endl;}
};
```

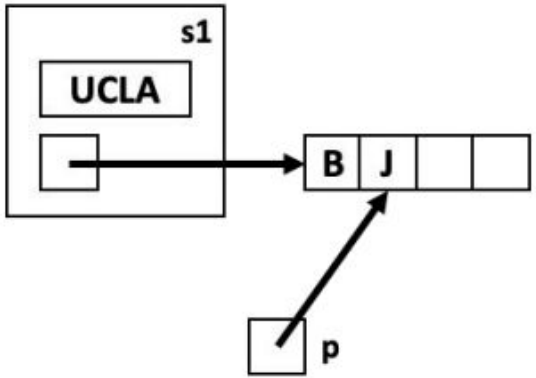
```
class Smallberg
{
  public:
  Smallberg():age(20)
  {cout << "Smallberg's here!" << endl;}
  int age;
  TA s;
  ~Smallberg()
  {cout << "Smallberg's out!" << endl;}
};

int main() {
  Smallberg s;
  cout << s.age << endl;
}
```

```
./main
Hi!
Hi!
Shichang's here
Smallberg's here!
20
Smallberg's out!
Shichang's out!
peace
peace
```

Copy Constructor: Motivation

Consider an example of UCLA and its students:

```
Student st1("Brian");
Student st2("John");
School s1("UCLA");
s1.addStudent(st1);
S1.addStudent(st2);
Student *p = s1.getStudent("John");
```



We want to create a new School called s2, with exactly the same content as s1. In other words, we want to clone s1.

```
School s2(""); s2=s1; // Is this correct?
```

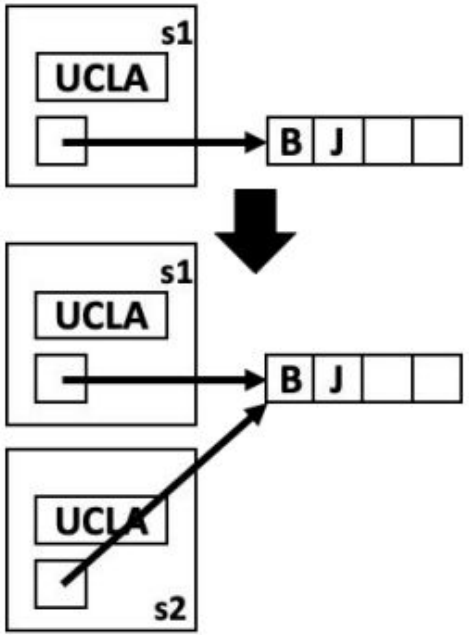
Copy Constructor: Shallow Copy Problem

We want to create a new School called s2, with exactly the same content as s1. In other words, we want to clone s1.

```
School s2(“”); s2=s1; // Definitely not!
```

What if grab values out of s1 and manually copy them into s2?

```
School s2(“”);
s2.setName(s1.getName( ));
... // copy all members and properties
```

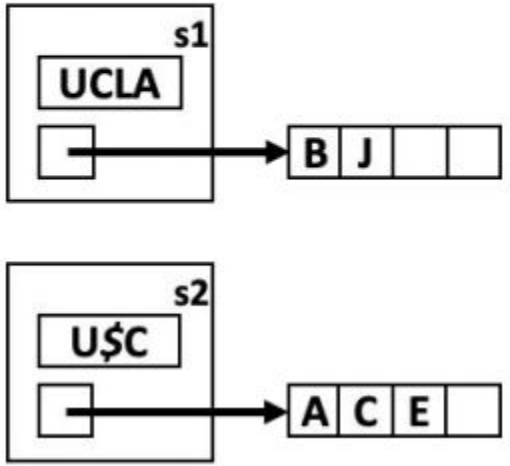


Copy Constructor: Example

Immutable. We don't change the object we're copying from.

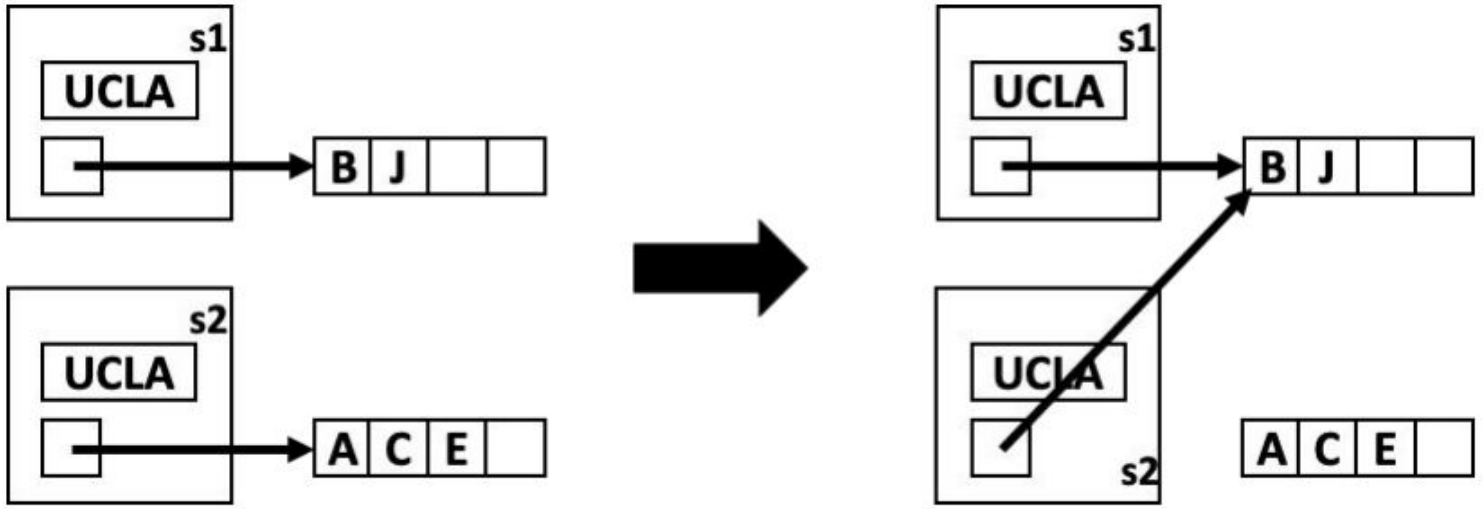
```
School::School(const School &aSchool){
    m_name = aSchool.m_name;
    m_numStudents = aSchool.m_numStudents;
    m_students = new Students[m_numStudents];
    for (int i = 0; i < m_numStudents; ++i)
        m_students[i] = aSchool.m_students[i];
}
```

Now you can do: `School s2(s1);`



Assignment Operator: Motivation

What if s2 = s1?



Assignment Operator: Example

Assignment operator

```
School& School:: operator=(const School &aSchool){
    m_name = aSchool.m_name;
    m_numStudents = aSchool.m_numStudents;
    m_students = new Students[m_numStudents];
    for (int i = 0; i < m_numStudents; i++)
        m_students[i] = aSchool.m_students[i];

    return *this;
}
```

*Do not forget *this!*

Assignment Operator: Example

Make it better:

```
School& School:: operator=(const School &aSchool){
     if (this != &aSchool)
     {
          m_name = aSchool.m_name;
          m_numStudents = aSchool.m_numStudents;
          delete[] m_students;
          m_students = new Students[m_numStudents];
          for (int i = 0; i < m_numStudents; i++)
              m_students[i] = aSchool.m_students[i];
     }
     return *this;
}
```

Worksheet

Codeshare

Room 1

Room 2

Room 3

Room 4

Room 5

Room 6