Monday Oct. 29

Lecture 14

~ Lab Test 1 Marks (Written)

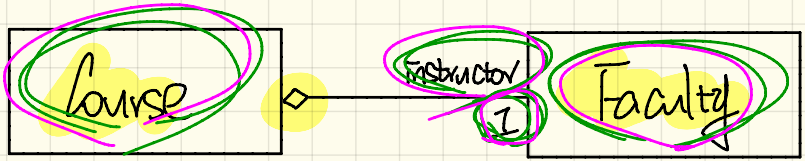
~ Lab Test 2 Guide

↳ Programming 40%

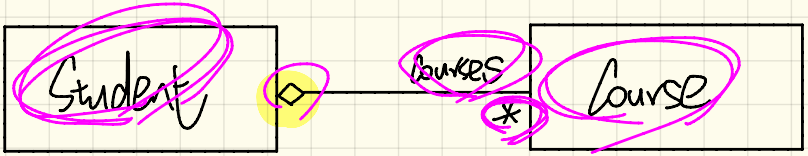
↳ Written 60%

Review: Aggregation

Single Container



Multiple Containees



Java Implementation

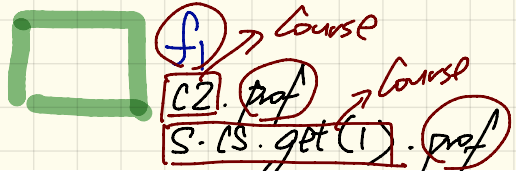
```
class Course {
    Faculty instructor;
    ...
}
```

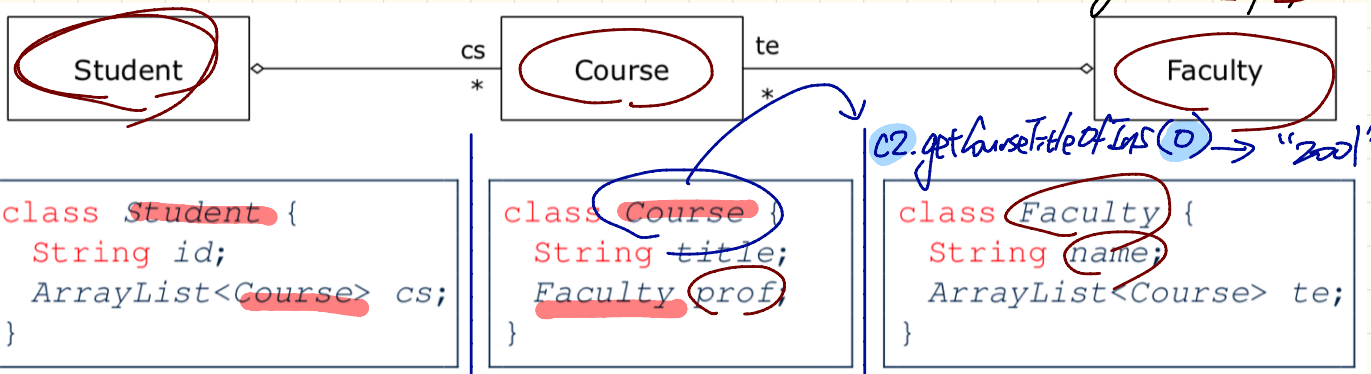
```
class Faculty {
    ...
}
```

```
class Student {
    Course[] courses;
    ...
}
```

```
class Course {
    ...
}
```

Dot Notation for Navigation Aggregations





String getInstructorName(int i)

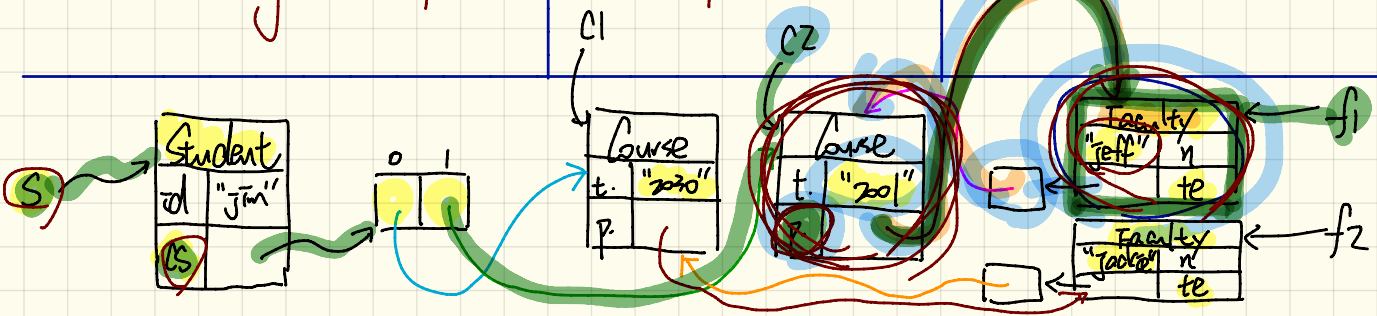
this.cs.get(i).prof.name

String getInstructorName()

this.prof.name

String getName()

this.name



Composition : No Sharing

Composition

```
class File {
  String name;
  File(String name) {
    this.name = name;
  }
}
```

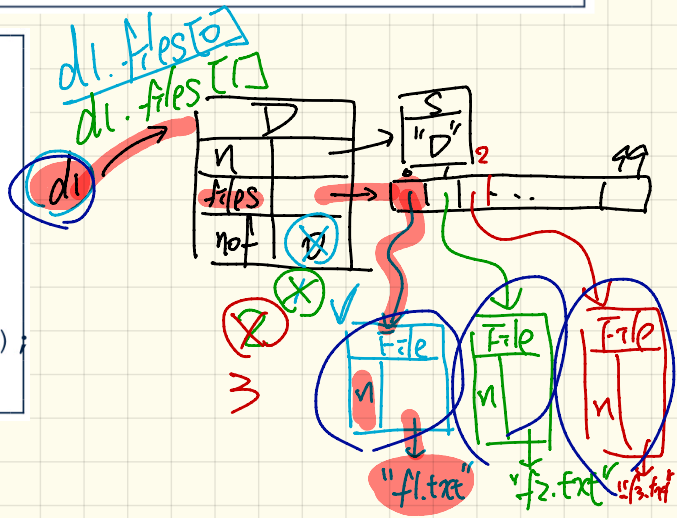
```
class Directory {
  String name;
  File[] files;
  int nof; /* num of files */
  Directory(String name) {
    this.name = name;
    files = new File[100];
  }
  void addFile(String fileName) {
    files[nof] = new File(fileName);
    nof ++;
  }
}
```

"f1.txt"

"f1.txt"

```
1  @Test
2  public void testComposition() {
3    Directory d1 = new Directory("D");
4    d1.addFile("f1.txt");
5    d1.addFile("f2.txt");
6    d1.addFile("f3.txt");
7    assertTrue(
8      d1.files[0].name.equals("f1.txt"));
9  }
```

File[] File String



Copy Constructor

```
class Directory {
    Directory ( Directory other) {
    }
}
```

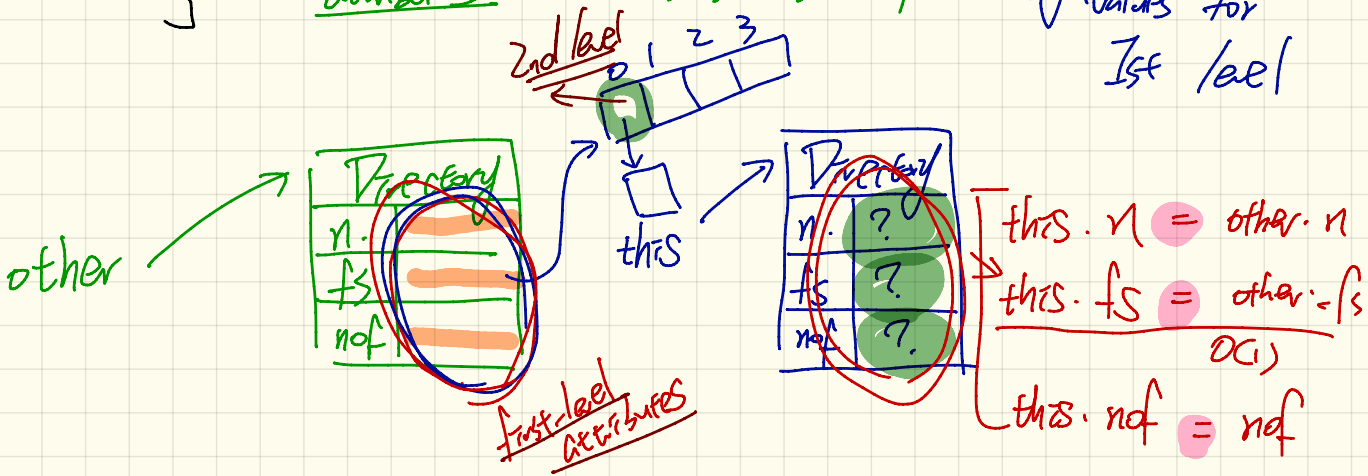
copy constructor

constructor

attributes: name, files, nof

Shallow copy:

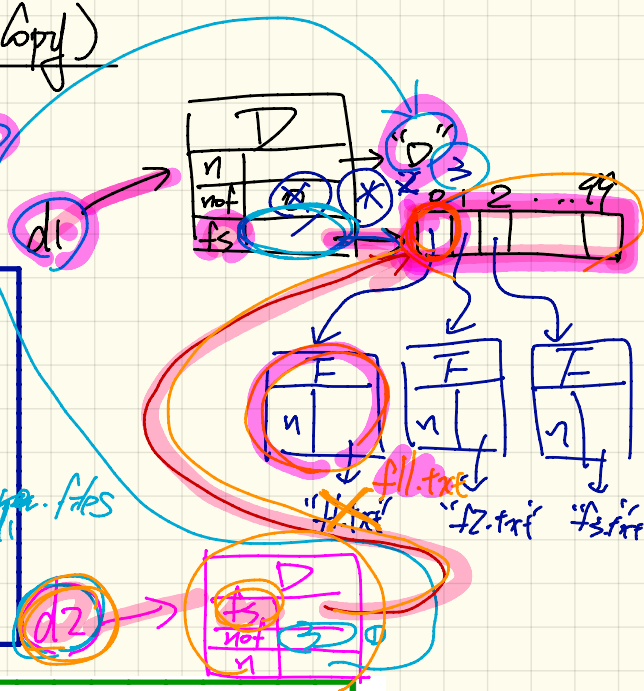
copy attribute values for 1st level



Composition : Copy Constructor (Shallow Copy)

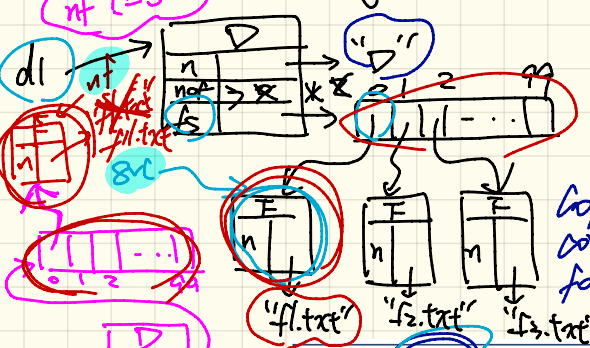
```
class Directory {
  Directory(Directory other) {
    /* value copying for primitive type */
    nof = other.nof;
    /* address copying for reference type */
    name = other.name; files = other.files; }
  void addFile(String fileName) {
    files[nof] = new File(fileName);
    nof ++;
  }
```

```
@Test
void testShallowCopyConstructor() {
  Directory d1 = new Directory("D");
  d1.addFile("f1.txt"); d1.addFile("f2.txt"); d1.addFile("f3.txt");
  Directory d2 = new Directory(d1);
  assertTrue(d1.files == d2.files); /* violation of composition */
  d2.files[0].changeName("f11.txt");
  assertFalse(d1.files[0].name.equals("f1.txt")); }
```



Composition : Copy Constructor (Deep Copy)

d2.files == d1.files F
d2.files[0] == d1.files[0] F



calling a constructor in the current class

```
class Directory {
  Directory(String name) {
    this.name = new String(name);
    files = new File[100]; }
  Directory(Directory other) {
    this(other.name);
    for(int i = 0; i < nof; i ++) {
      File src = other.files[i];
      File nf = new File(src);
      this.addFile(nf); } }
  void addFile(File f) { ... } }
```

```
class File {
  File(File other) {
    this.name =
      new String(other.name);
  }
}
```

```
@Test
void testDeepCopyConstructor() {
  Directory d1 = new Directory("D");
  d1.addFile("f1.txt"); d1.addFile("f2.txt"); d1.addFile("f3.txt");
  Directory d2 = new Directory(d1);
  assertTrue(d1.files != d2.files); /* composition preserved */
  d2.files[0].changeName("f11.txt");
  assertTrue(d1.files[0].name.equals("f1.txt")); }
```