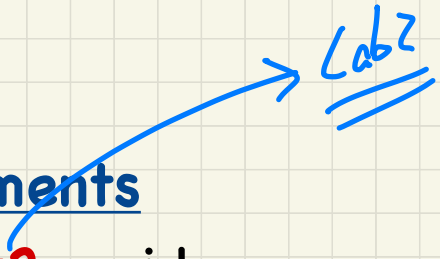
Lecture 16 - Nov 7

Inheritance

SMS: Attempts without Inheritance
SMS: Use of extend, super
Visibility: Project, Package, Class



Announcements

- ProgTest2: guide
- Lab3 due this Wednesday (equals & copy constructor)
- WrittenTest2 results released on Friday
- ProgTest1: Visit office hours to discuss your solution

First Design Attempt

design flaws
no implementation flaws

```
public class Student {
   private Course[] courses;
   private int noc;

   private int kind;
   private double premiumRate;
   private double discountRate;

   public Student (int kind){
      this.kind = kind;
   }
   ...
}
```

RS
nRS

```
public double getTuition(){
   double tuition = 0;
   for(int i = 0; i < this.noc; i++){
      tuition += this.courses[i].fee;
   }
   if (this.kind == 1) {
      return tuition * this. premiumRate;
   }
   else if (this.kind == 2) {
      return tuition * this.discountRate;
   }
}
```

base

```
public void register(Course c){
   int MAX = -1;
   if (this.kind == 1) { MAX = 6; }
   else if (this.kind == 2) { MAX = 4; }
   if (this.noc == MAX) { /* Error */ }
   else {
      this.courses[this.noc] = c;
      this.noc ++;
   }
}
```

```
Student rs = new Student(1);
Student nrs = new Student(2);
...
rs.getTuition()
nrs.getTuition()
```

First Design Attempt

```
public class Student {
   private Course[] courses;
   private int noc;

   private int kind;
   private double premiumRate;
   private double discountRate;

   public Student (int kind){
      this.kind = kind;
   }
   ...
}
```

RS

NRS

should it be separated to diff classes

a class collects attr/methods relevant to a common theme

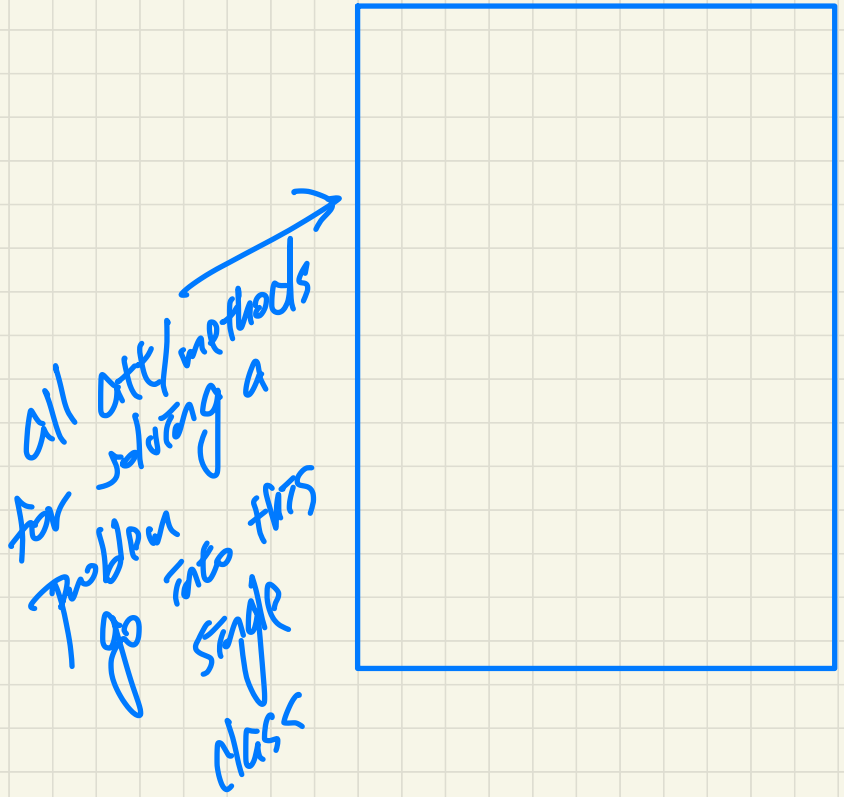
Good design?

Judge by **Cohesion**

```
public double getTuition(){
   double tuition = 0;
   for(int i = 0; i < this.noc; i++){
      tuition += this.courses[i].fee;
   }
   if (this.kind == 1) {
      return tuition * this. premiumRate;
   }
   else if (this.kind == 2) {
      return tuition * this.discountRate;
   }
}
```

```
public void register(Course c){
   int MAX = -1;
   if (this.kind == 1) { MAX = 6; }
   else if (this.kind == 2) { MAX = 4; }
   if (this.noc == MAX) { /* Error */ }
   else {
      this.courses[this.noc] = c;
      this.noc ++;
   }
}
```

Superman Class



First Design Attempt

```
public class Student {
   private Course[] courses;
   private int noc;

   private int kind;
   private double premiumRate;
   private double discountRate;

   public Student (int kind){
      this.kind = kind;
   }
   ...
}
```

```
public double getTuition(){
   double tuition = 0;
   for(int i = 0; i < this.noc; i++){
      tuition += this.courses[i].fee;
   }
   if (this.kind == 1) {
      return tuition * this. premiumRate;
   }
   else if (this.kind == 2) {
      return tuition * this.discountRate;
   }
}
```

```
public void register(Course c){
   int MAX = -1;
   if (this.kind == 1) { MAX = 6; }
   else if (this.kind == 2) { MAX = 4; }
   if (this.noc == MAX) { /* Error */ }
   else {
      this.courses[this.noc] = c;
      this.noc ++;
   }
}
```

Good design?

Judge by Single Choice Principle

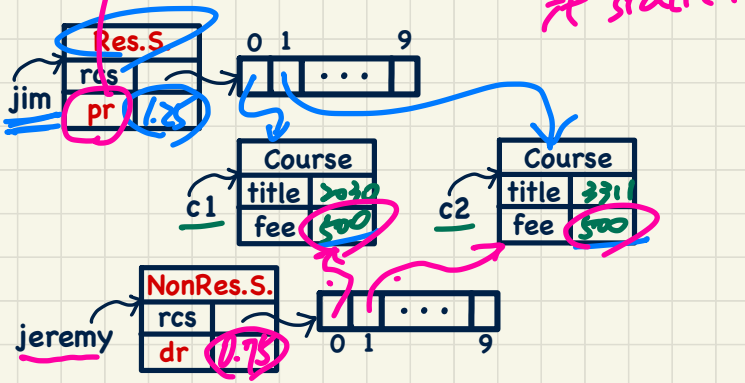
- Repeated if-conditions
- A new kind is introduced?
- An existing kind is obselete?

Testing Student Classes (without inheritance)

```java
public class ResidentStudent {
  private String name;
  private Course[] courses; private int noc;
  private double premiumRate; /* assume a m
  public ResidentStudent(String name) {
    this.name = name;
    this.courses = new Course[10];
  }
  public void register(Course c) {
    this.courses[this.noc] = c;
    this.noc ++;
  }
  public double getTuition() {
    double tuition = 0;
    for(int i = 0; i < this.noc; i ++) {
      tuition += this.courses[i].fee;
    }
    return tuition * this.premiumRate;
  }
}
```

```java
public class NonResidentStudent {
  private String name;
  private Course[] courses; private int noc;
  private double discountRate; /* assume a
  public NonResidentStudent(String name) {
    this.name = name;
    this.courses = new Course[10];
  }
  public void register(Course c) {
    this.courses[this.noc] = c;
    this.noc ++;
  }
  public double getTuition() {
    double tuition = 0;
    for(int i = 0; i < this.noc; i ++) {
      tuition += this.courses[i].fee;
    }
    return tuition * this.discountRate;
  }
}
```

```java
public class StudentTester {
  public static void main(String[] args) {
    Course c1 = new Course("EECS2030", 500.00); /* title and fee */
    Course c2 = new Course("EECS3311", 500.00); /* title and fee */
    ResidentStudent jim = new ResidentStudent("J. Davis");
    jim.setPremiumRate(1.25);
    jim.register(c1); jim.register(c2);
    NonResidentStudent jeremy = new NonResidentStudent("J. Gibbons");
    jeremy.setDiscountRate(0.75);
    jeremy.register(c1); jeremy.register(c2);
    System.out.println("Jim pays " + jim.getTuition());
    System.out.println("Jeremy pays " + jeremy.getTuition());
  }
}
```



Student Classes (without inheritance): Maintenance (1)

```
public class ResidentStudent {
  private String name;
  private Course[] courses; private int noc;
  private double premiumRate; /* assume a m
  public ResidentStudent(String name) {
    this.name = name;
    this.courses = new Course[10];
  }
  public void register(Course c) {
    this.courses[this.noc] = c;
    this.noc ++;
  }
  public double getTuition() {
    double tuition = 0;
    for(int i = 0; i < this.noc; i ++) {
      tuition += this.courses[i].fee;
    }
    return tuition * this.premiumRate;
  }
}
```

add constraint

```
public class NonResidentStudent {
  private String name;
  private Course[] courses; private int noc;
  private double discountRate; /* assume a
  public NonResidentStudent(String name) {
    this.name = name;
    this.courses = new Course[10];
  }
  public void register(Course c) {
    this.courses[this.noc] = c;
    this.noc ++;
  }
  public double getTuition() {
    double tuition = 0;
    for(int i = 0; i < this.noc; i ++) {
      tuition += this.courses[i].fee;
    }
    return tuition * this.discountRate;
  }
}
```

add constraint

Maintenance e.g., a new registration constraint:

```
if(numberOfCourses >= MAX_ALLOWANCE) {
   throw new TooManyCoursesException("Too Many Courses");
}
else { ... }
```

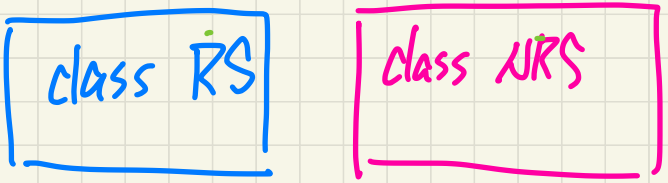
Student Classes (without inheritance): Maintenance (2)

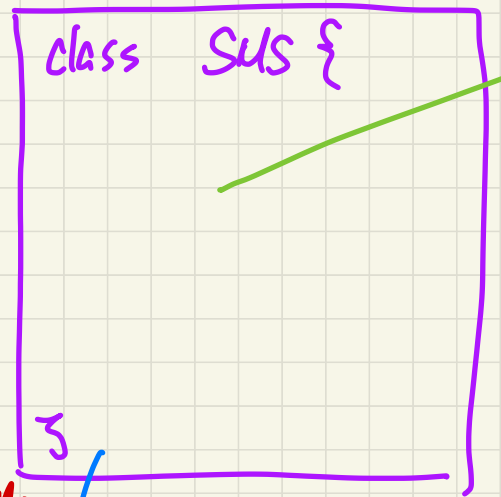
```
public class ResidentStudent {
  private String name;
  private Course[] courses; private int noc;
  private double premiumRate; /* assume a m
  public ResidentStudent(String name) {
    this.name = name;
    this.courses = new Course[10];
  }
  public void register(Course c) {
    this.courses[this.noc] = c;
    this.noc ++;
  }
  public double getTuition() {
    double tuition = 0;
    for(int i = 0; i < this.noc; i ++) {
      tuition += this.courses[i].fee;
    }
    return tuition * this.premiumRate;
  }
}
```

```
public class NonResidentStudent {
  private String name;
  private Course[] courses; private int noc;
  private double discountRate; /* assume a
  public NonResidentStudent(String name) {
    this.name = name;
    this.courses = new Course[10];
  }
  public void register(Course c) {
    this.courses[this.noc] = c;
    this.noc ++;
  }
  public double getTuition() {
    double tuition = 0;
    for(int i = 0; i < this.noc; i ++) {
      tuition += this.courses[i].fee;
    }
    return tuition * this.discountRate;
  }
}
```

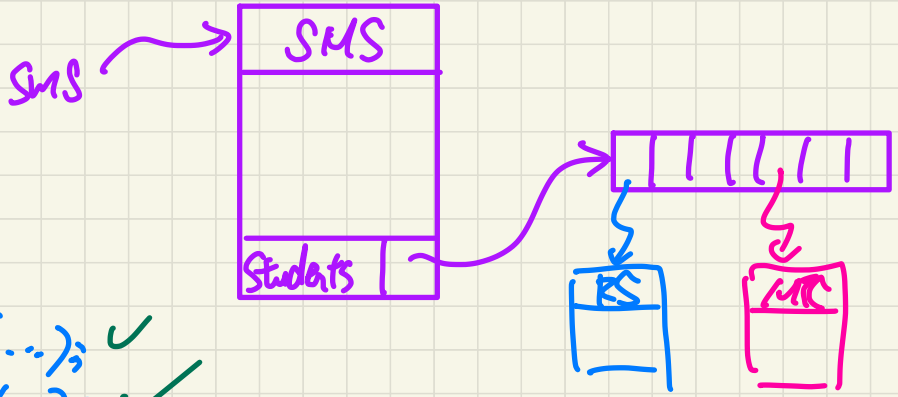
Maintenance e.g., a new tuition formula:

```
/* ... can be premiumRate or discountRate */
...
return tuition * inflationRate * ...;
```





RS[] students = new RS[100];
students[0] = new RS(...); ✓
students[1] = new NRS(..); ✗
at runtime:



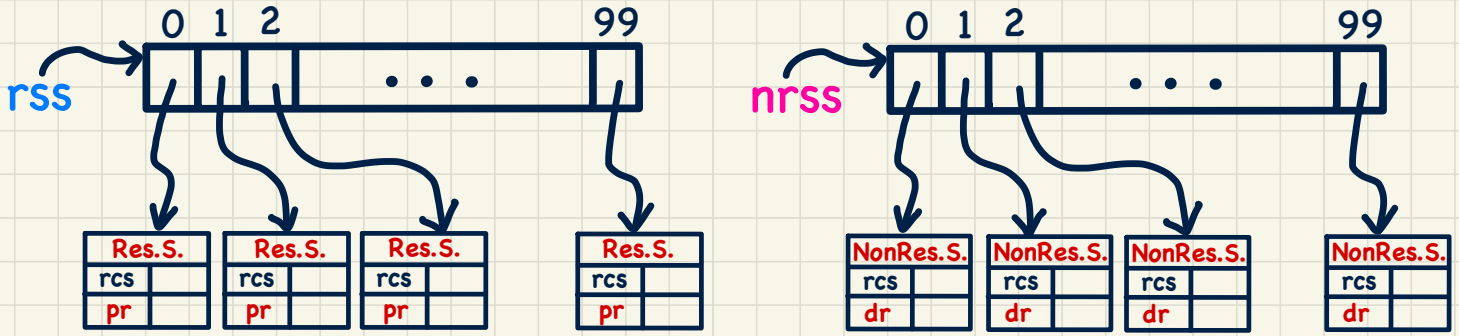
No No! poor design.

Object[] students;
students[0] = new RS(...); ✓
students[1] = new NRS(..); ✓

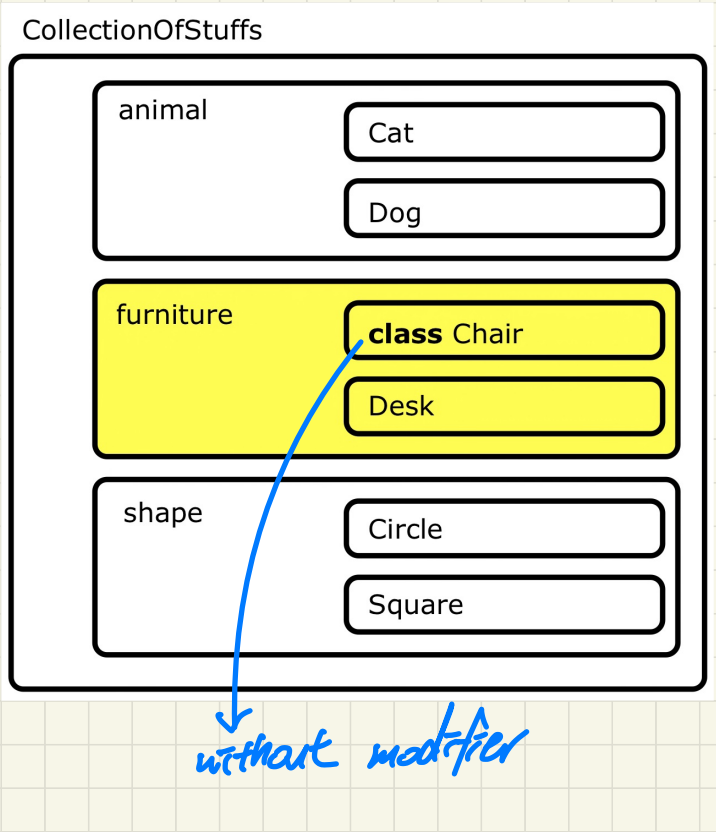
A Collection of Students (without inheritance)

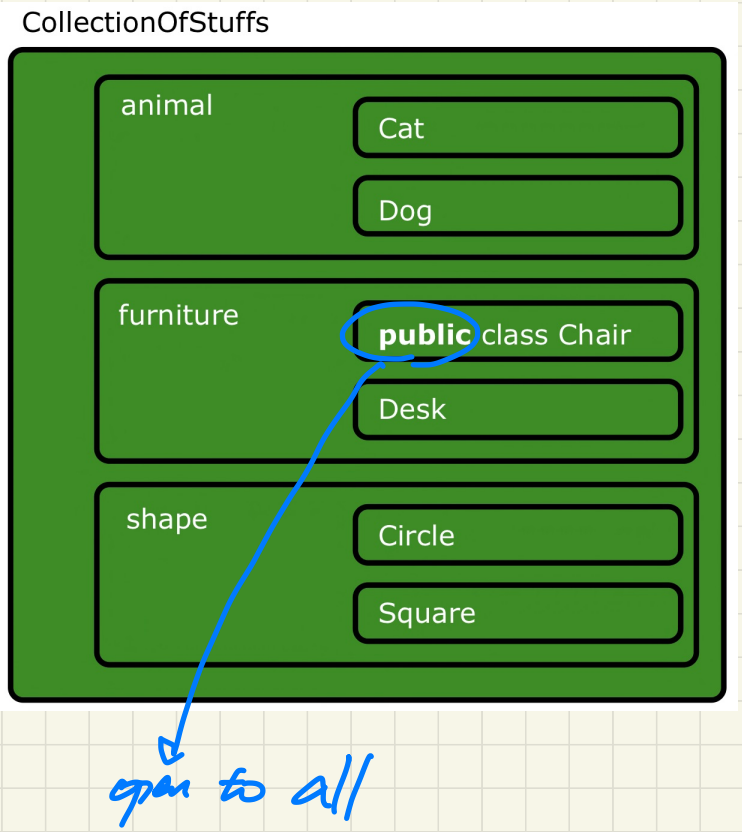
```
public class StudentManagementSystem {
  private ResidentStudent[] rss;
  private NonResidentStudent[] nrss;
  private int nors; /* number of resident students */
  private int nonrs; /* number of non-resident students */
  public void addRS(ResidentStudent rs){ rss[nors]=rs; nors++; }
  public void addNRS(NonResidentStudent nrs){ nrss[nonrs]=nrs;nonrs++; }
  public void registerAll(Course c) {
    for(int i = 0; i < nors; i ++) { rss[i].register(c); }
    for(int i = 0; i < nonrs; i ++) { nrss[i].register(c); }
  }
}
```

→ multiple & duplicated loops are necessary ∵ multiple arrays

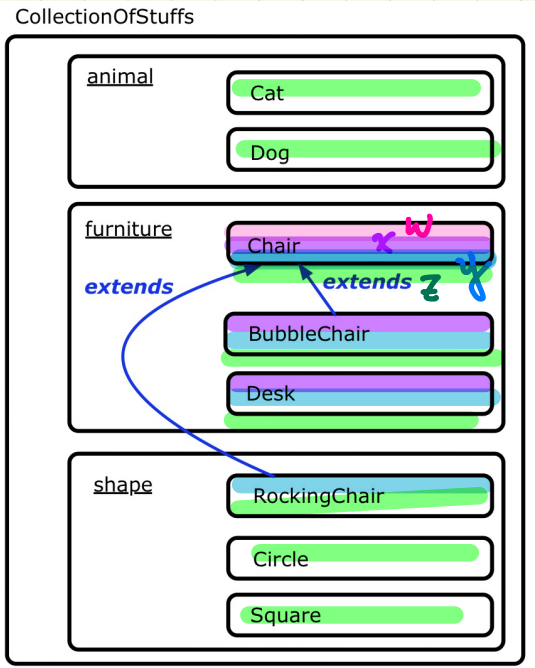


Visibility: Classes





Visibility: Attributes and Methods



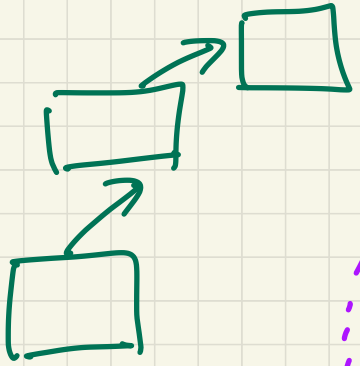

```
public class Chair {
    private w;
    int x;
    protected int y;
    public int z;
}
```

1. visible to subclasses in either the same package of the package.
2. other classes in the same package

as f: no modifier + subclasses in other packages.

	Class	Package	Subclass (same pkg)	Subclass (different pkg)	Non-Subclass (across Project)
public					
protected					
no modifier					
private					

Student Classes (with inheritance)



```
class Student {
  String name;
  Course[] registeredCourses;
  int numberOfCourses;
  Student(String name) {
    this.name = name;
    registeredCourses = new Course[10];
  }
  void register(Course c) {
    registeredCourses[numberOfCourses] = c;
    numberOfCourses ++;
  }
  double getTuition() {
    double tuition = 0;
    for(int i = 0; i < numberOfCourses; i ++) {
      tuition += registeredCourses[i].fee;
    }
    return tuition; /* base amount only */
  }
}
```

declare whats in common among subclasses
↳ RS
↳ NRS

parent version

base

base amt. calculation (shared among classes)

no need to re-declare name, registeredCourses, numberOfCourses

inherit everything from parent

```
class ResidentStudent extends Student {
  double premiumRate; /* there's a mutator meth
  ResidentStudent(String name) { super(name); }
  /* register method is inherited */
  double getTuition() {
    double base = super.getTuition();
    return base * premiumRate;
  }
}
```

calling the constructor in parent class

return the base amt. calculation from parent class

```
class NonResidentStudent extends Student {
  double discountRate; /* there's a mutator method
  NonResidentStudent(String name) { super(name); }
  /* register method is inherited */
  double getTuition() {
    double base = super.getTuition();
    return base * discountRate;
  }
}
```

inherited, overridden version