

Lecture 17 - Nov. 7

Inheritance

***Implementing a Child Class: Principles
Visibility: Class, Attribute/Method
Static Types: Expectations
Polymorphism: Intuition***

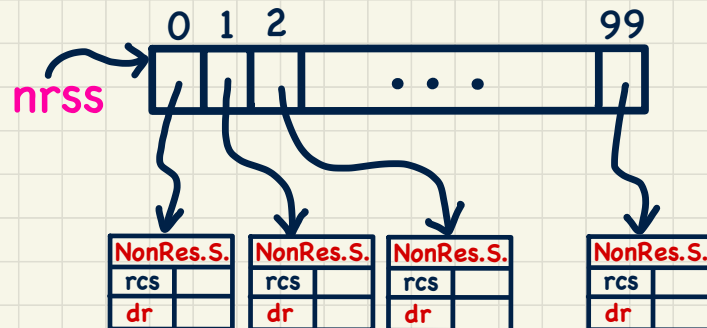
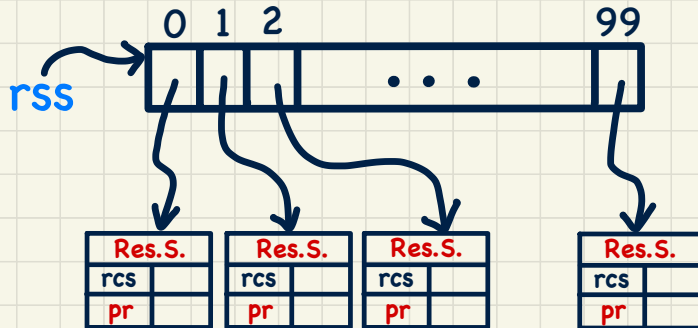
Announcements/Reminders

- **Lab4** released (**ProgTest3** on November 20)
- Guide & Questions for **WrittenTest2** released
- Materials for In-Lab Demo on **Inheritance** released
- **ProgTest2** results & feedback Monday Nov 11

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); }  
    }  
}
```

↓ # for loops correspond to # kinds of students



Recall: 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 */  
    }  
}
```

In writing a subclass :

(1) cannot re-declare inherited attributes

(2) add new attributes/methods.

(3) happy with inherited methods, leave as is

(4) not satisfied with inherited methods → override them

inherited
(and cannot be re-declared)

not overridden

parent version

overridden

new attributes in sub-classes

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

set P

child version 1

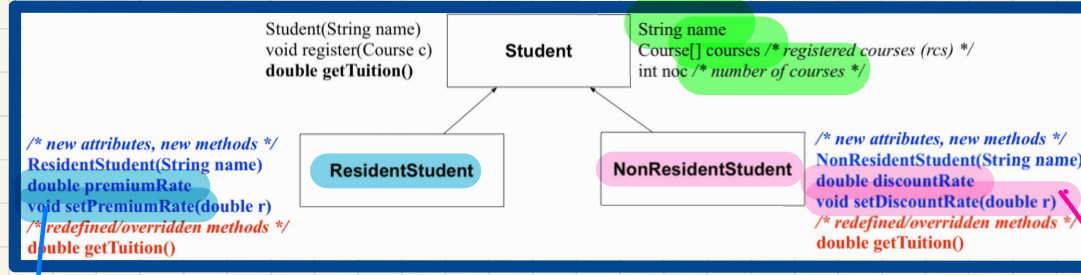
this.name = "...";

```
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;  
    }  
}
```

set P

child version 2

Visualizing Parent and Child Objects



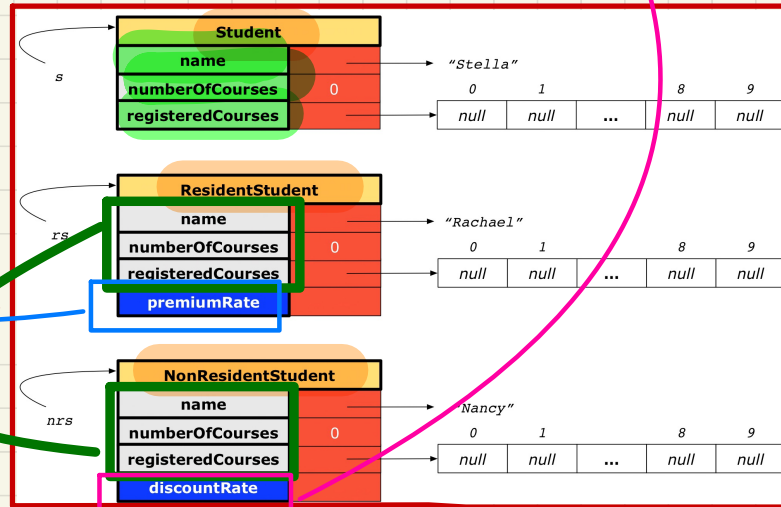
Inheritance Hierarchy

```
Student s = new Student("Stella");
ResidentStudent rs = new ResidentStudent("Rachael");
NonResidentStudent nrs = new NonResidentStudent("Nancy");
```

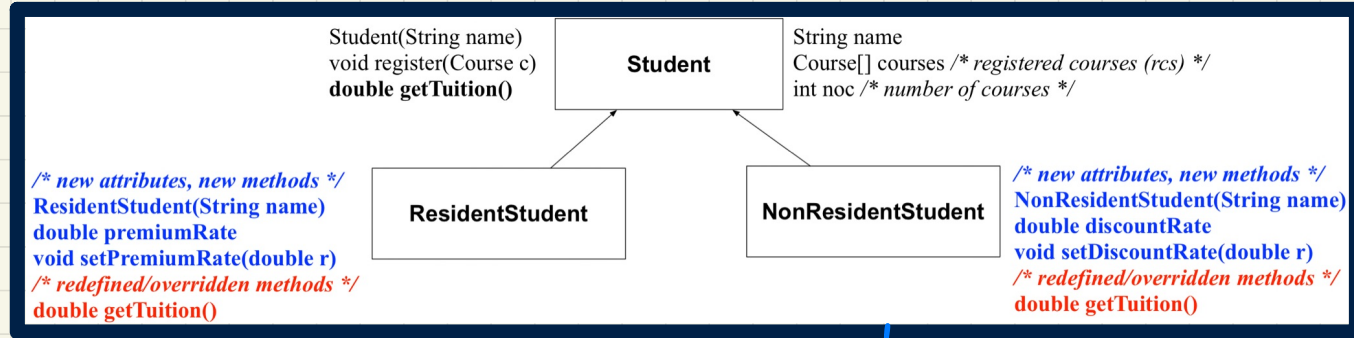
Declaring Static Types

Runtime Object Structure

Inherited from Student



Testing Student Classes (with inheritance)



```

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());
    }
}
  
```

→ in-lab demo

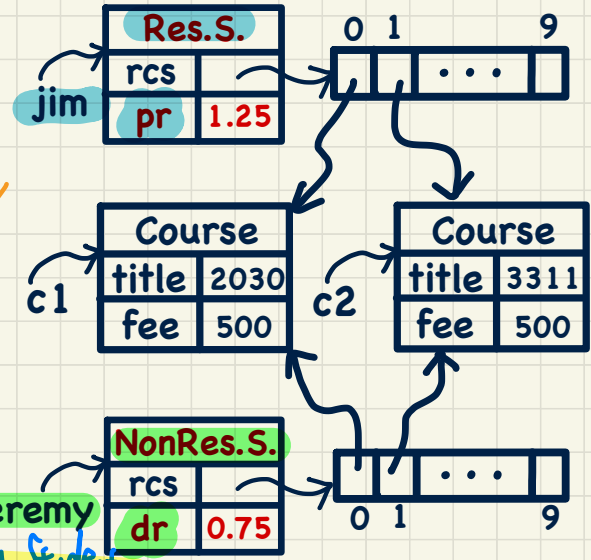
→ invoke Student constructor

version in Student invoked

→ invoke ① the specific

child version

② both first invoke getT from Student



Modifiers in Java

① private

② protected

③ public

④ no modifier

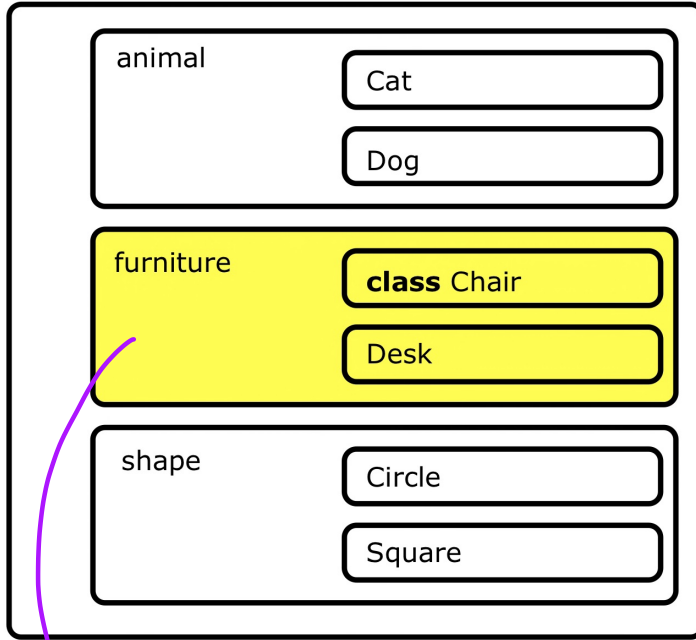
inheritance

classes

attributes/methods

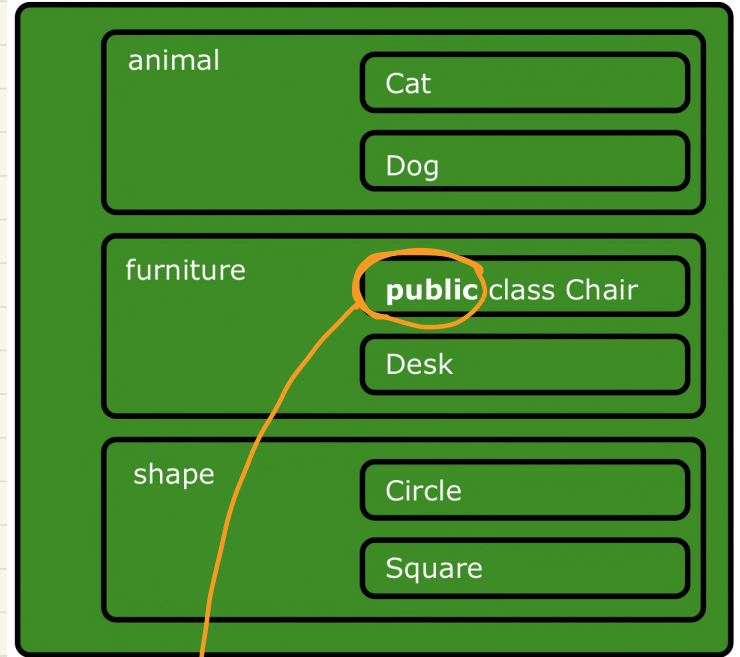
Visibility: Classes

CollectionOfStuffs



package-level
visibility

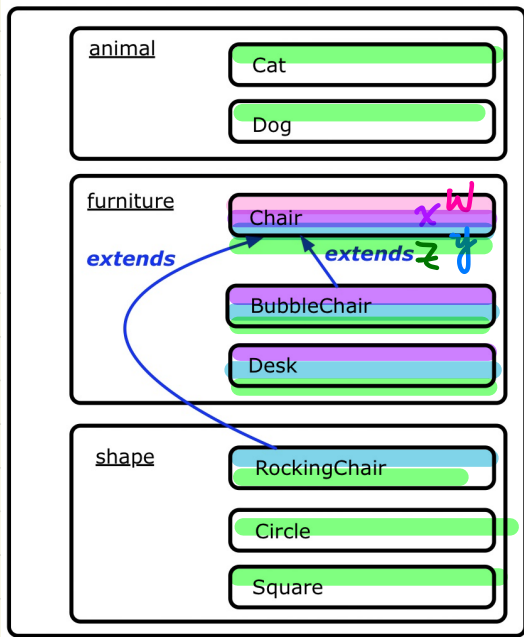
CollectionOfStuffs



universal
visibility.

Visibility: Attributes and Methods

CollectionOfStuffs



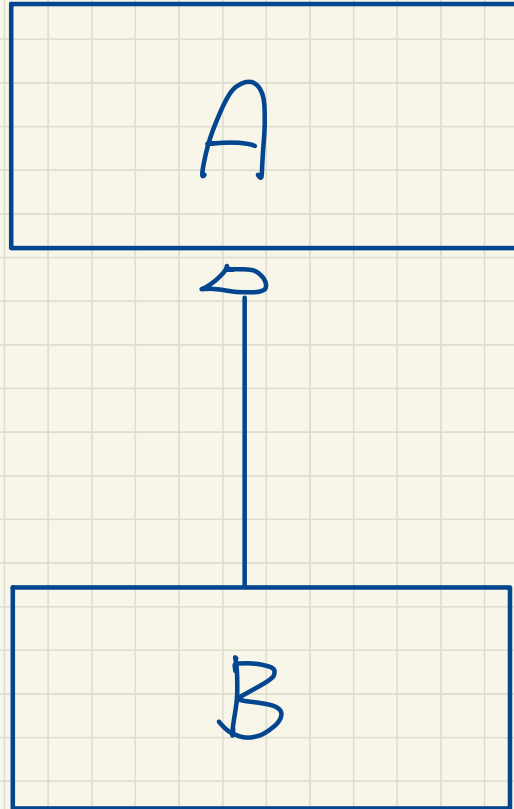
```

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

not inherited to subclasses

as if:
(1) pkg visib.
(2) subclass (in diff. pkg.)

	CLASS	PACKAGE	SUBCLASS (same pkg)	SUBCLASS (different pkg)	NON-SUBCLASS (across Project)
public	Green	Green	Green	Green	Green
protected	Green	Green	Green	Green	Red
no modifier	Green	Green	Green	Red	Red
private	Green	Red	Red	Red	Red



(1) all attributes private.

(2) as you discover
necessarily, make attrs.
protected

Student Classes (with inheritance): Expectations

exp. of a variable determines the range of atts/methods that can be called on that variable

Student(String name)
void register(Course c)
double getTuition()

Student

String name
Course[] courses /* registered courses (rcs) */
int noc /* number of courses */

ResidentStudent

NonResidentStudent

/* new attributes, new methods */
ResidentStudent(String name)
double premiumRate
void setPremiumRate(double r)
/* redefined/overridden methods */
double getTuition()

/* new attributes, new methods */
NonResidentStudent(String name)
double discountRate
void setDiscountRate(double r)
/* redefined/overridden methods */
double getTuition()

exp(RS) = exp(Student) + new atts/methods

sibling classes

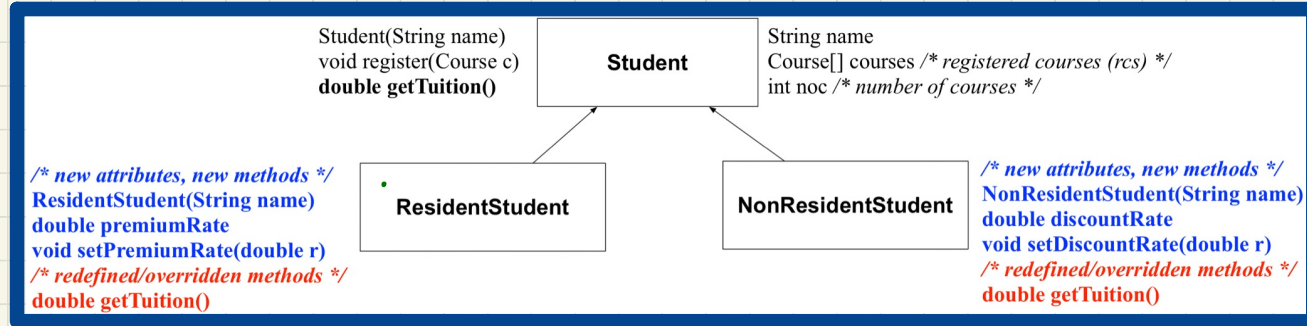
```
Student s = new Student("Stella");
ResidentStudent rs = new ResidentStudent("Rachael");
NonResidentStudent nrs = new NonResidentStudent("Nancy");
```

inherited exp from parent

	name	rcs	noc	reg	getT	pr	setPR	dr	setDR
S.									
rs.									
nrs.									

sibling exp.
new exp introduced in subclasses

Intuition: Polymorphism



```

1 Student s = new Student("Stella");
2 ResidentStudent rs = new ResidentStudent("Rachael");
3 rs.setPremiumRate(1.25);
4 s = rs; (A) Is this valid? */
5 rs = s; (B) Is this valid? */
  
```

4. Assumption was wrong:

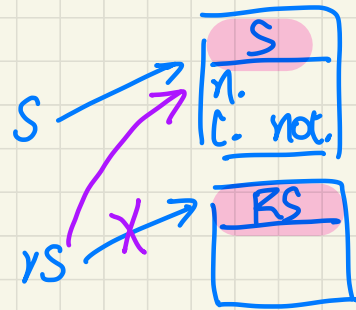
rs = s
not compile

ST: parent class

1. Assume $rs \odot S$ compiles.

2. Execute the re-assignment.

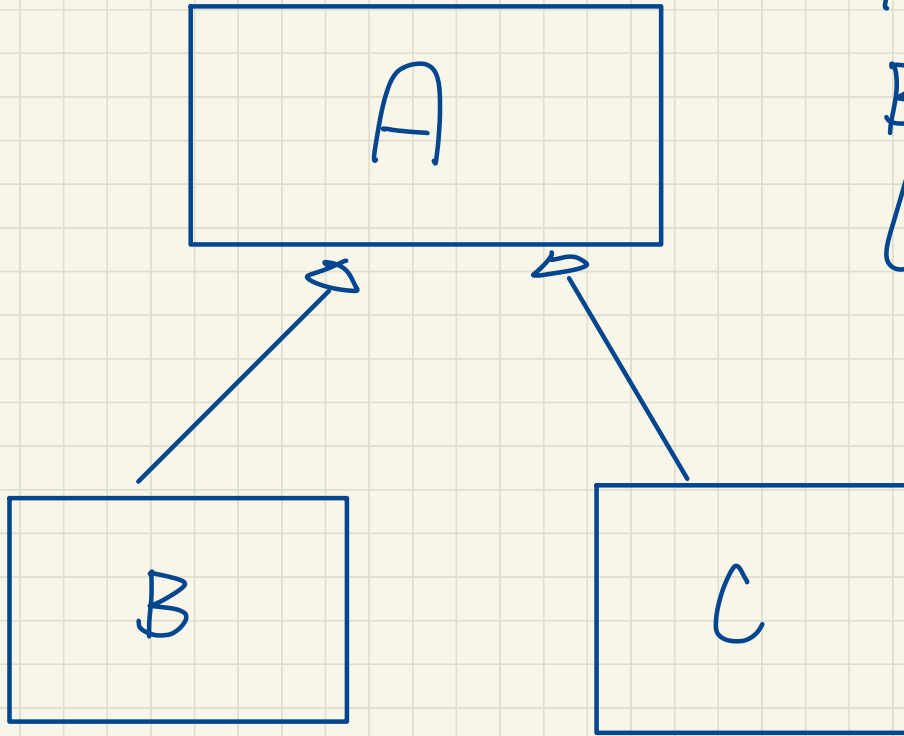
3. Exp. of rs. *



*

rs.name
rs.noc
rs.courses
rs.pr

Student obj does not crash have pr



A o1 = new A();

B o2 = new B();

C o3 = new C();

o1 = o2;

o1 = o3;

o2 = o3;

o2 = o1;

o3 = o1;

o3 = o2;