Lecture 17 - Nov 5

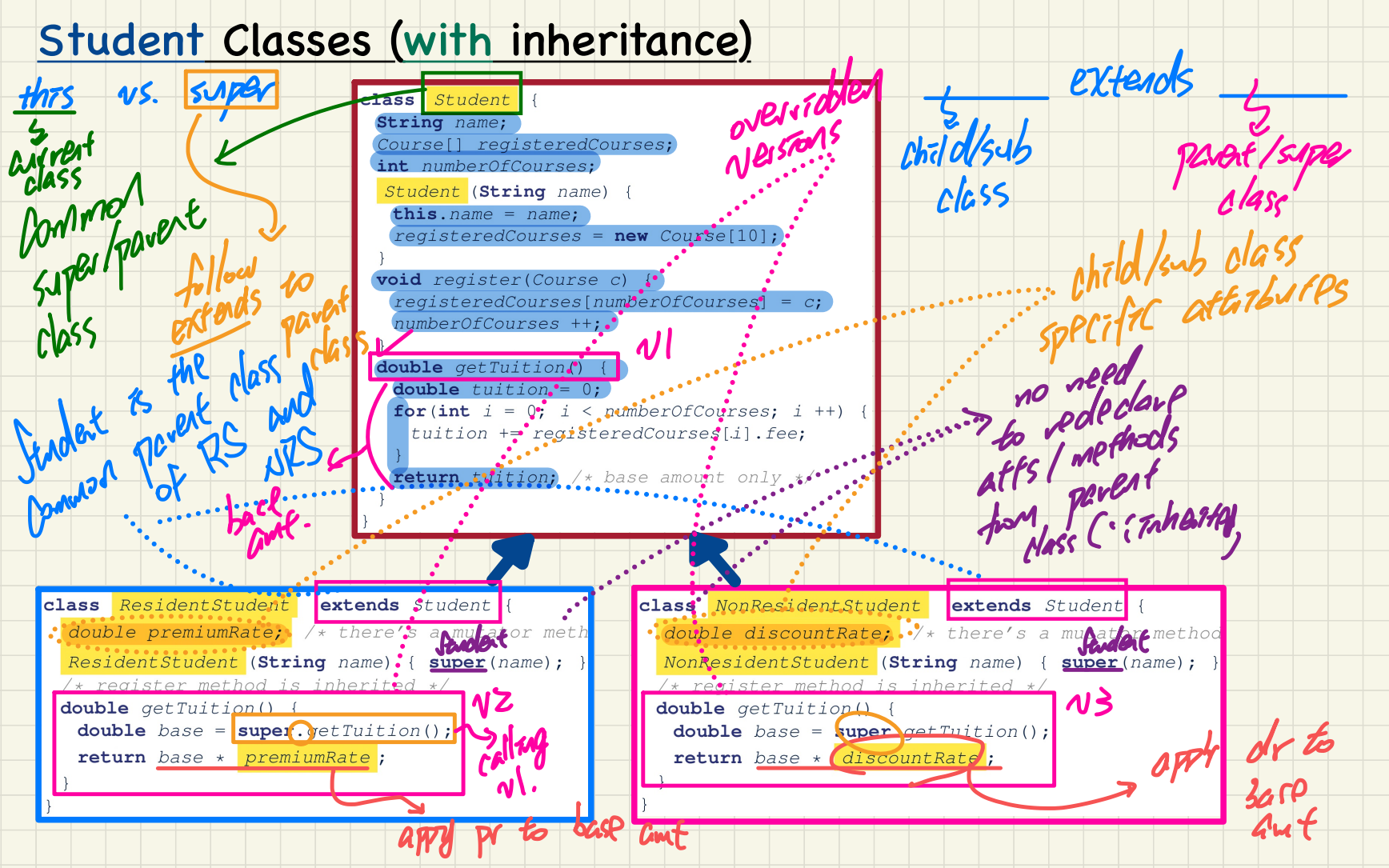
Inheritance

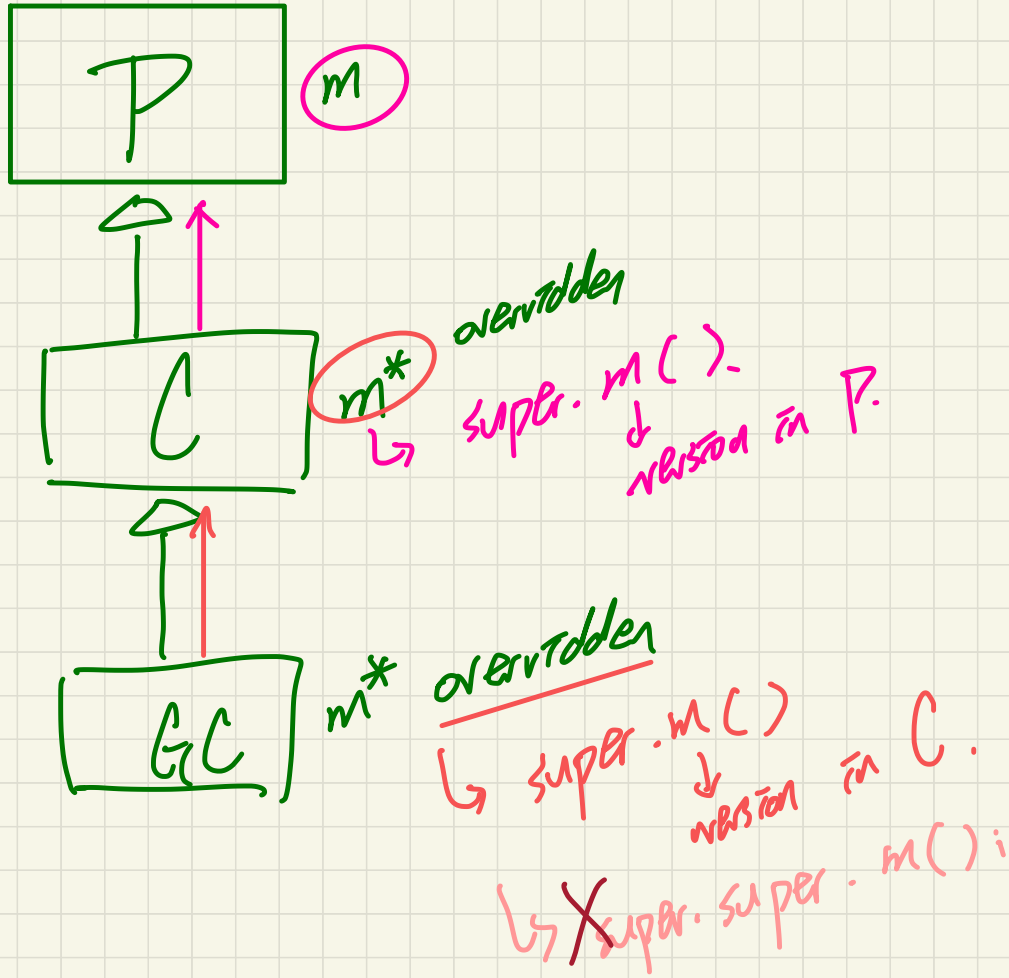
Modifiers in Java
Static Types and Expectations

Announcements/Reminders

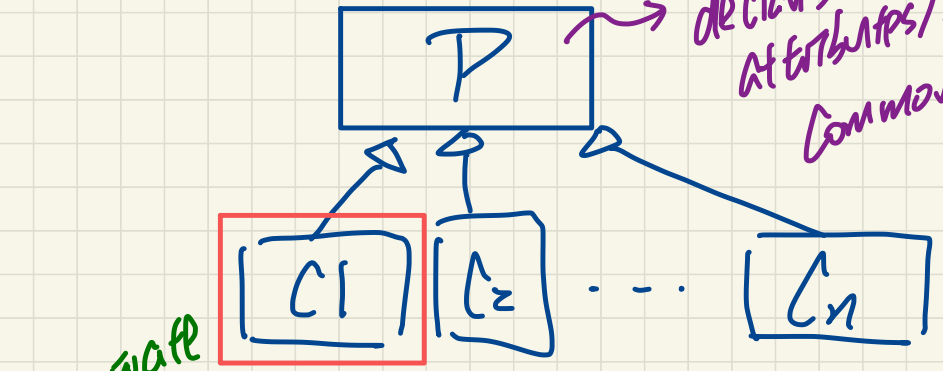
- Today's class: notes template posted
- Lab4 to be released
- Tracing Exercises: assertEquals and Person, PersonCollector

Student Classes (with inheritance)




P
m
C
m*
overridden
super.m().
version in P.
GC
m*
overridden
super.m()
version in C.
super.super.m();

When programming with inheritance:



declares attributes/methods common to all subclasses

(min. duplicates ↳ sat S.C.P.)

non-private

(2a) everything from parent is inherited

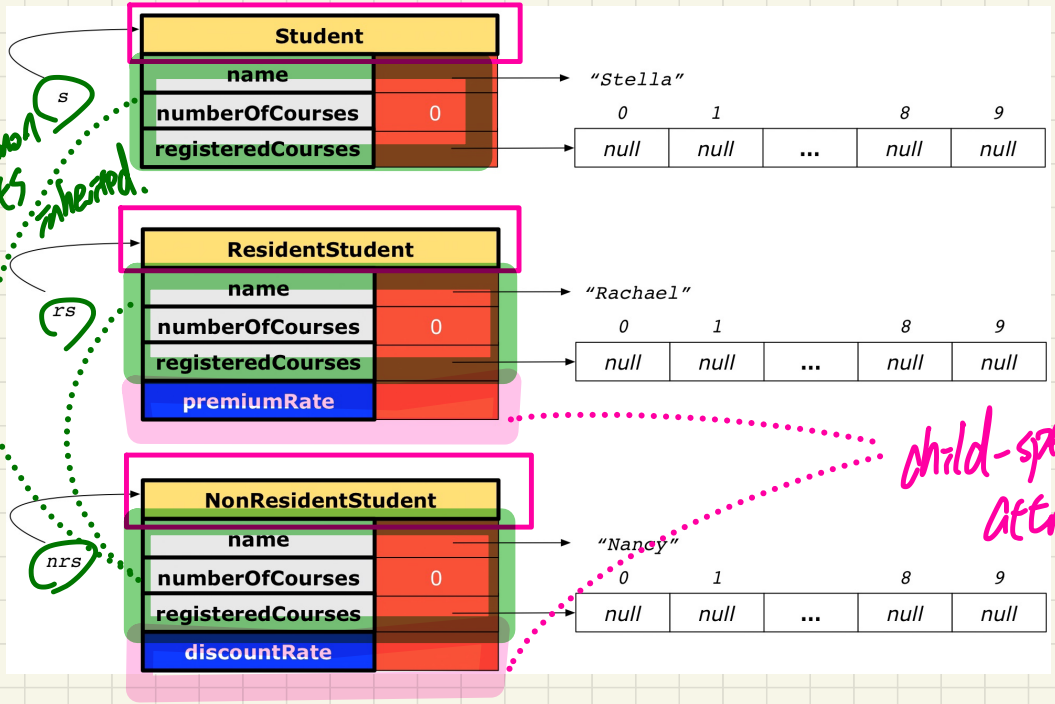
(2b) additional, unique attrs/methods for this child class

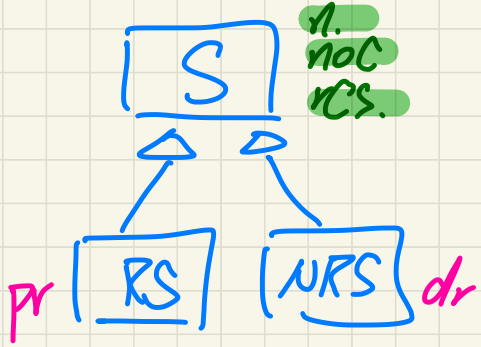
(2c) override/redefine methods inherited from parent class

Visualizing Parent and Child Objects

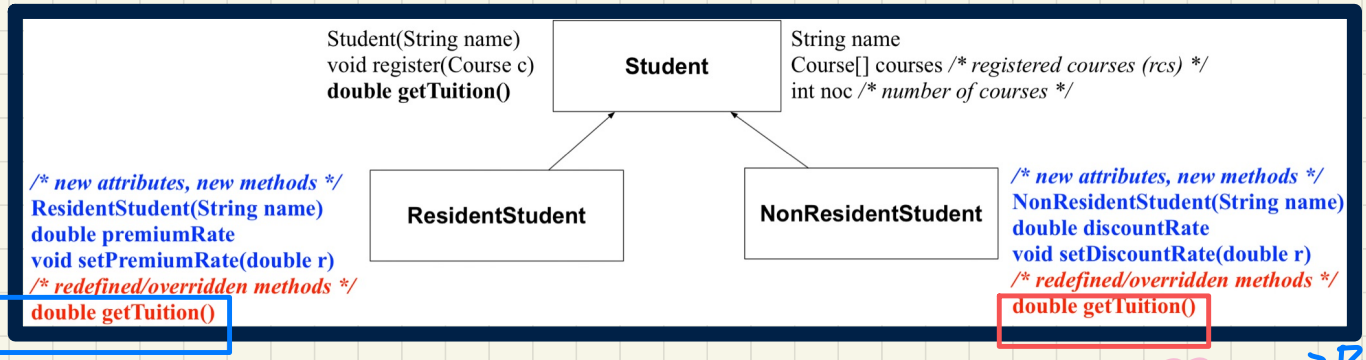
D.T.s

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





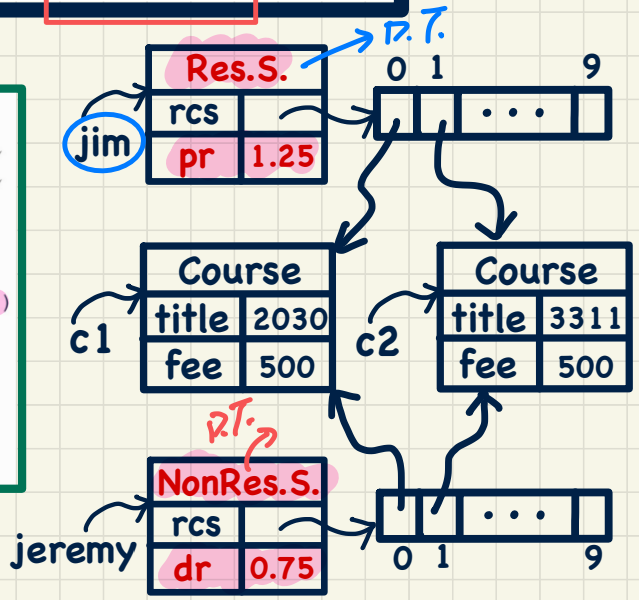
Testing Student Classes (with inheritance)




```
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());
  }
}
```

D.T. RS

D.T. NRS



Modifiers in Java

visibility.

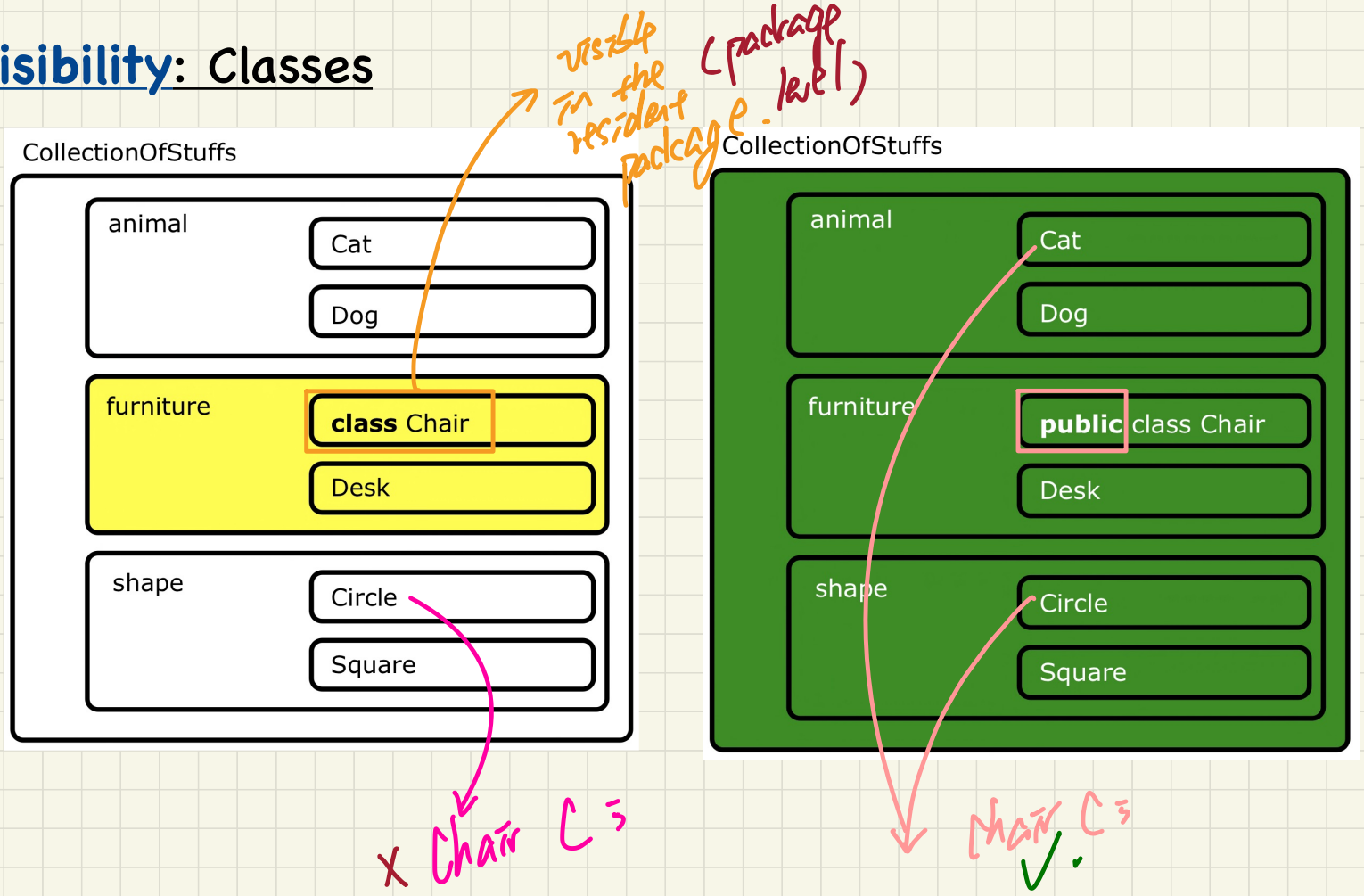
1. private
2. protected
3. no modifier
4. public

attributes / methods

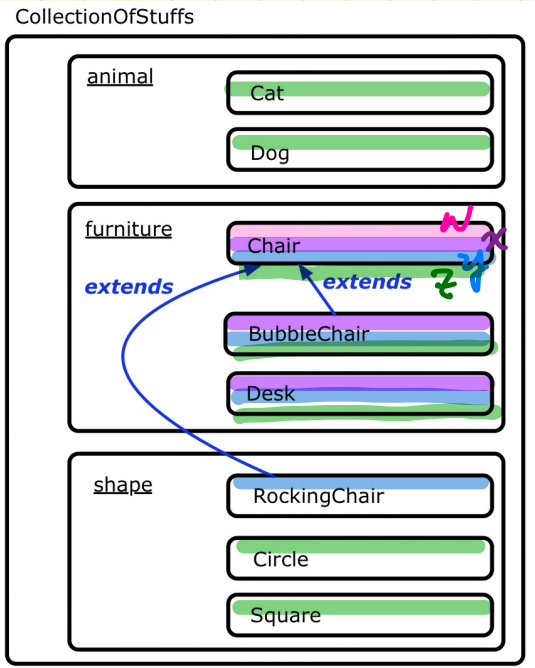
inheritance (e.g. Lab4)

classes.

Visibility: Classes



Visibility: Attributes and Methods

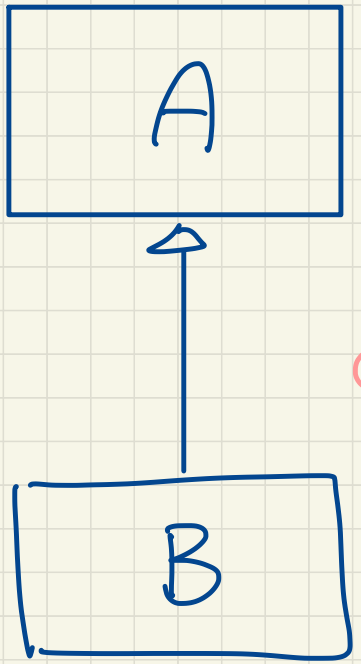



```
public class Chair {
    private int w;
    int x;
    protected int y;
    public int z;
}
```

package-level

package level + inheritance (same or not package)

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



(1) declare all attributes to be private
↳ not inherited.

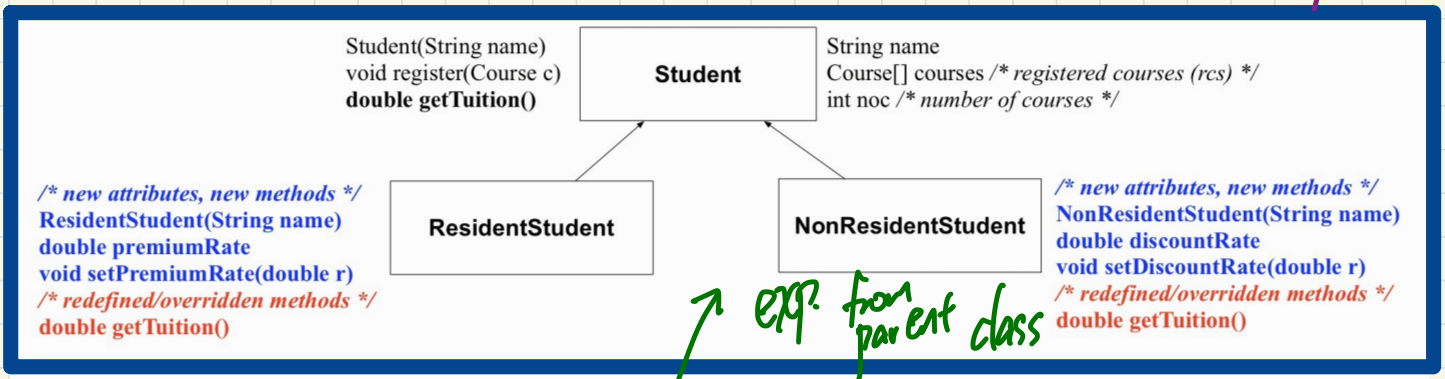
(2) those attributes that should be inherited to B, make them protected.
↓
or no modifiers.

* each child class inherits exp. of its parent.

Student Classes (with inheritance): Expectations

range of attr/method callable

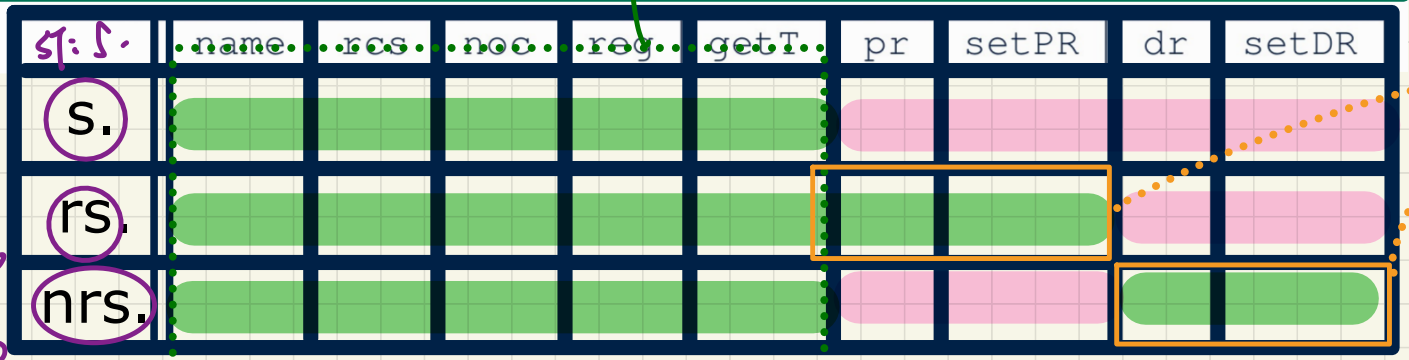
static/declared type



exp. from parent class

expectations of a ref. var. are determined by its static type

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



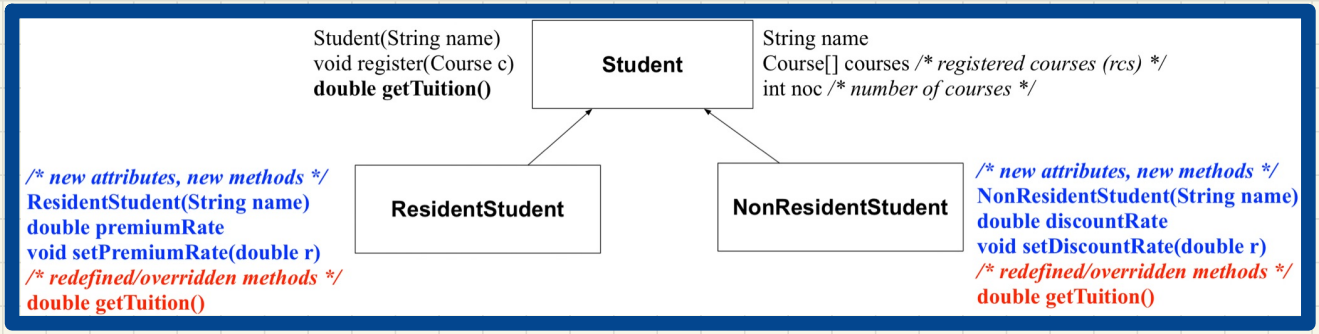
SI: S.

SI: RS

SI: NRS

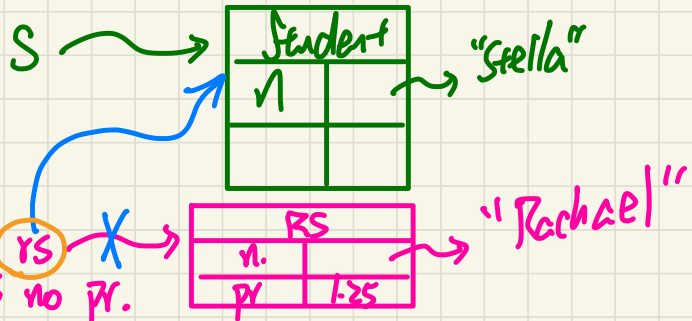
Intuition: Polymorphism

4. assumption is wrong ∵ rs = s should not compile.

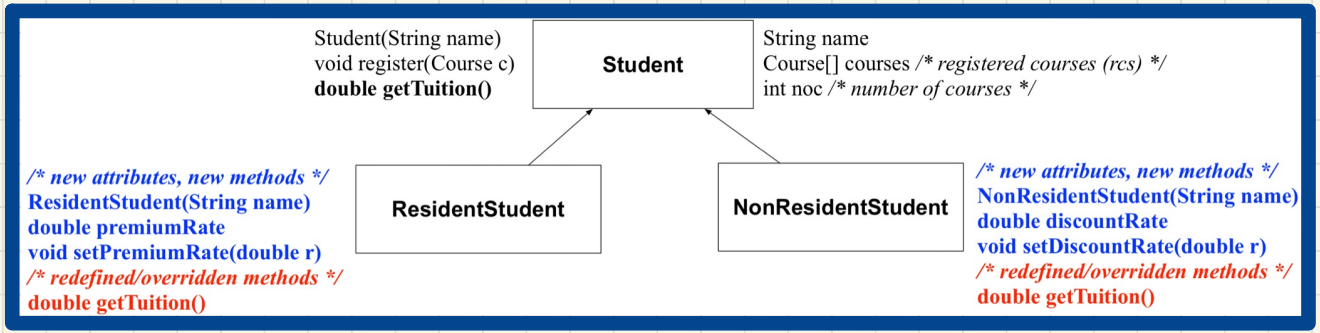


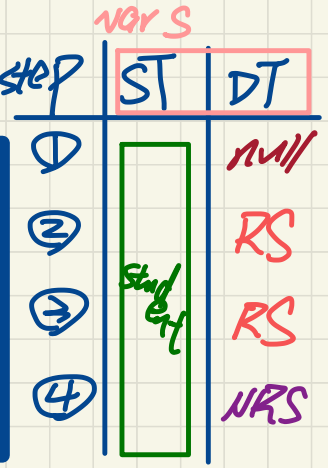

```
1  Student s = new Student("Stella");
2  ResidentStudent rs = new ResidentStudent("Rachael");
3  rs.setPremiumRate(1.25);
4  s = rs; /* Is this valid? */
5  rs = s; /* Is this valid? */
```

1. Assume rs = s compiles.
2. Execute the var assignment.
3. Exp of rs' ST (RS) includes: pr
   rs.pr → crash ∵ Student has no pr.

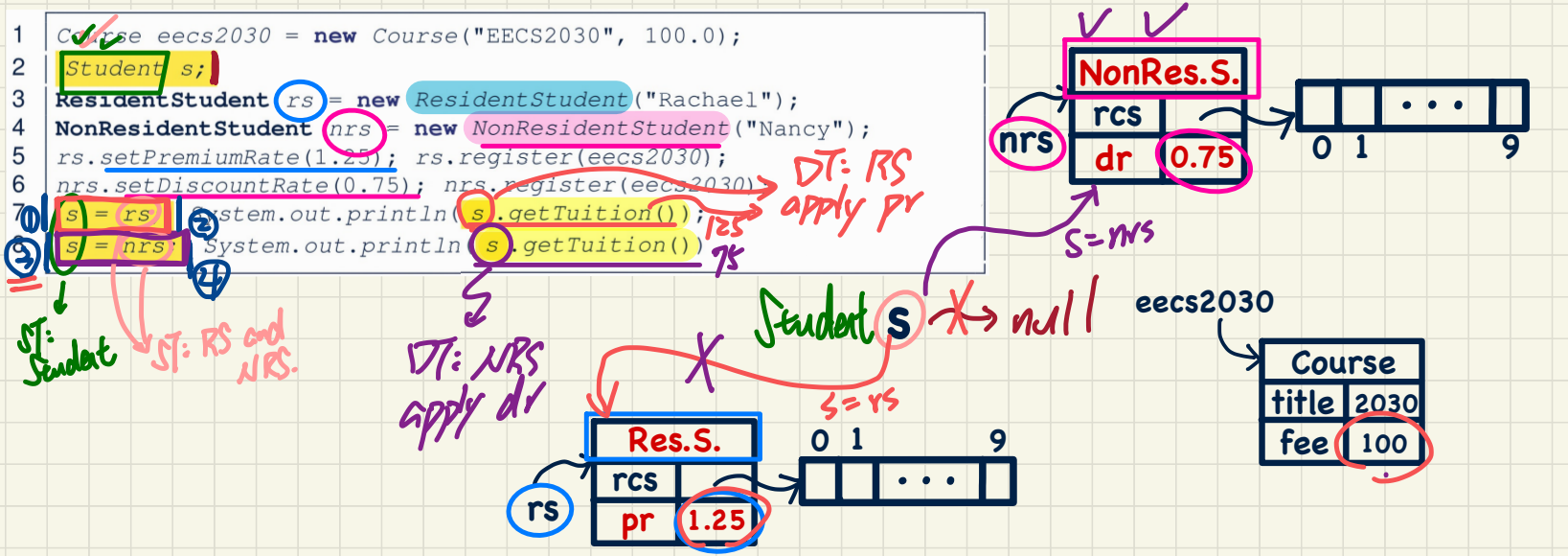


Intuition: Dynamic Binding






```
1  Course eecs2030 = new Course("EECS2030", 100.0);
2  Student s;
3  ResidentStudent rs = new ResidentStudent("Rachael");
4  NonResidentStudent nrs = new NonResidentStudent("Nancy");
5  rs.setPremiumRate(1.25); rs.register(eecs2030);
6  nrs.setDiscountRate(0.75); nrs.register(eecs2030);
7  s = rs;  System.out.println(s.getTuition());
8  s = nrs; System.out.println(s.getTuition());
```



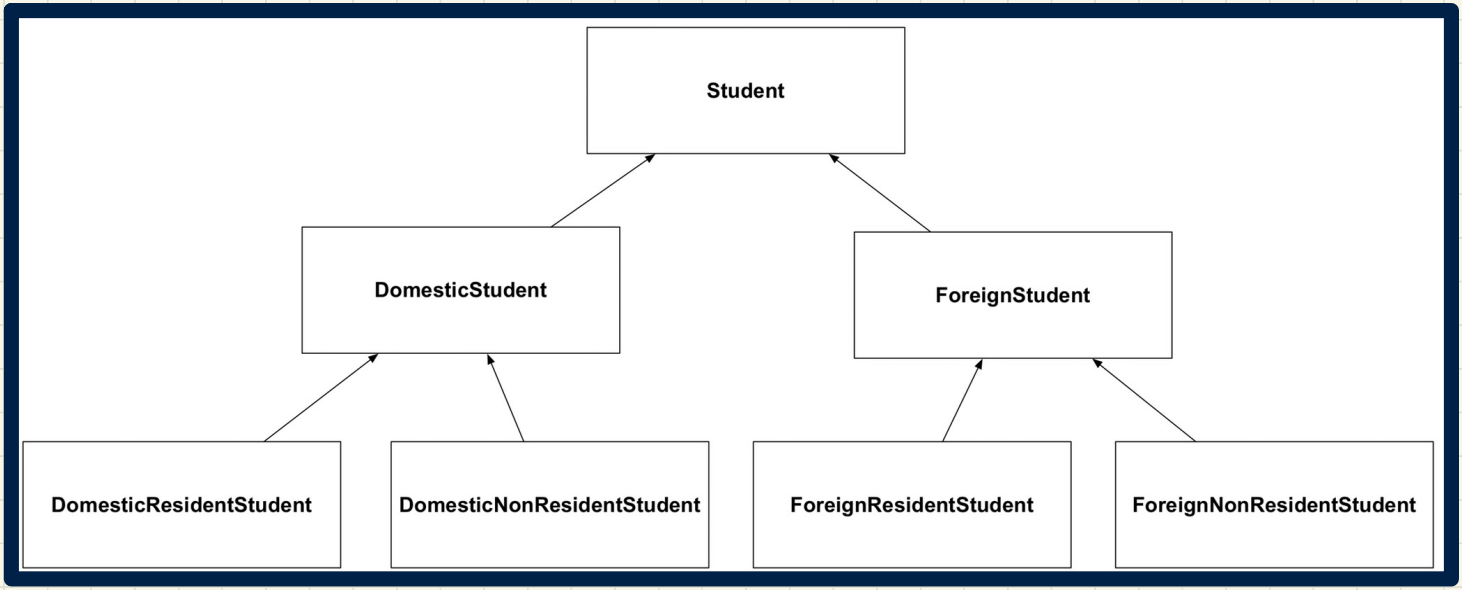
Compilation: static type

runtime: dynamic type

e.g. version of method called

e.g. exception ↓ ClassCastException.

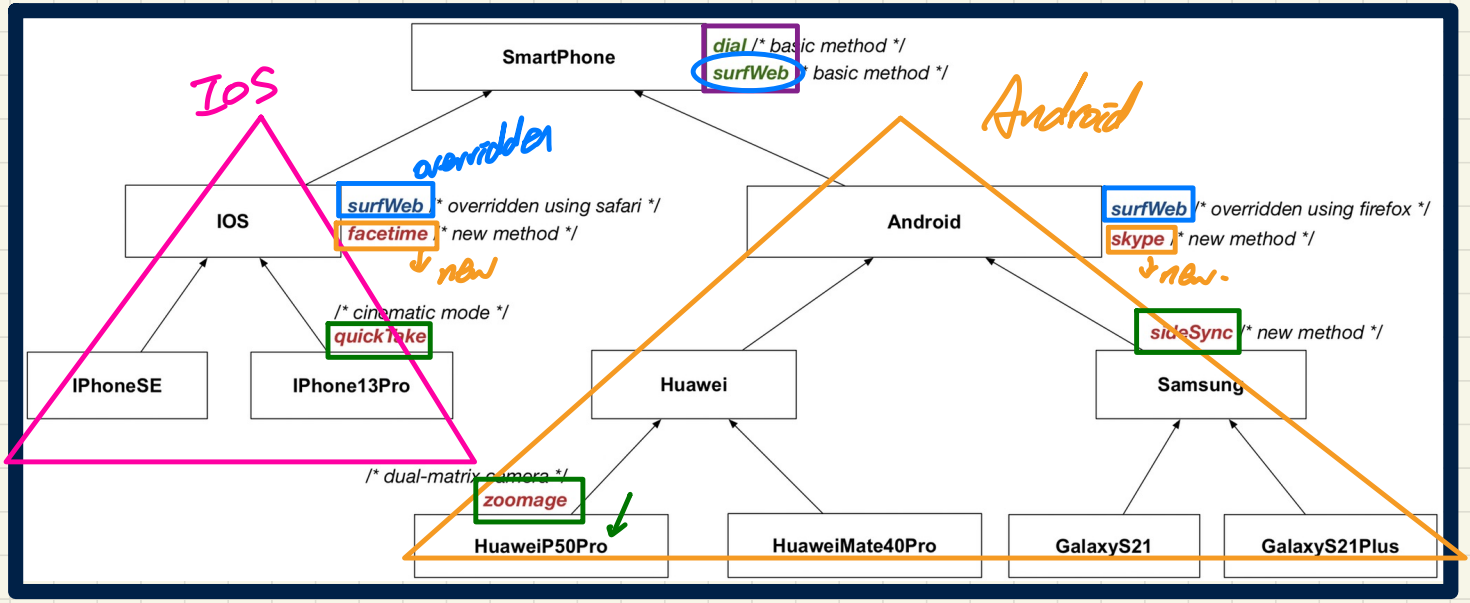
Multi-Level Inheritance Hierarchy: Students



Reflections:

- For **Design 1**, how many encodings to check for each method?
- For **Design 2**, how many arrays to store for SMS?
- For **Design 3**, where are common attributes/methods stored?

Multi-Level Inheritance Hierarchy: Smartphones



Reflections:

- For Design 1, how many encodings to check for each method?
- For Design 2, how many arrays to store for SMS?
- For Design 3, where are common attributes/methods stored?