Lecture 17 - Nov 9

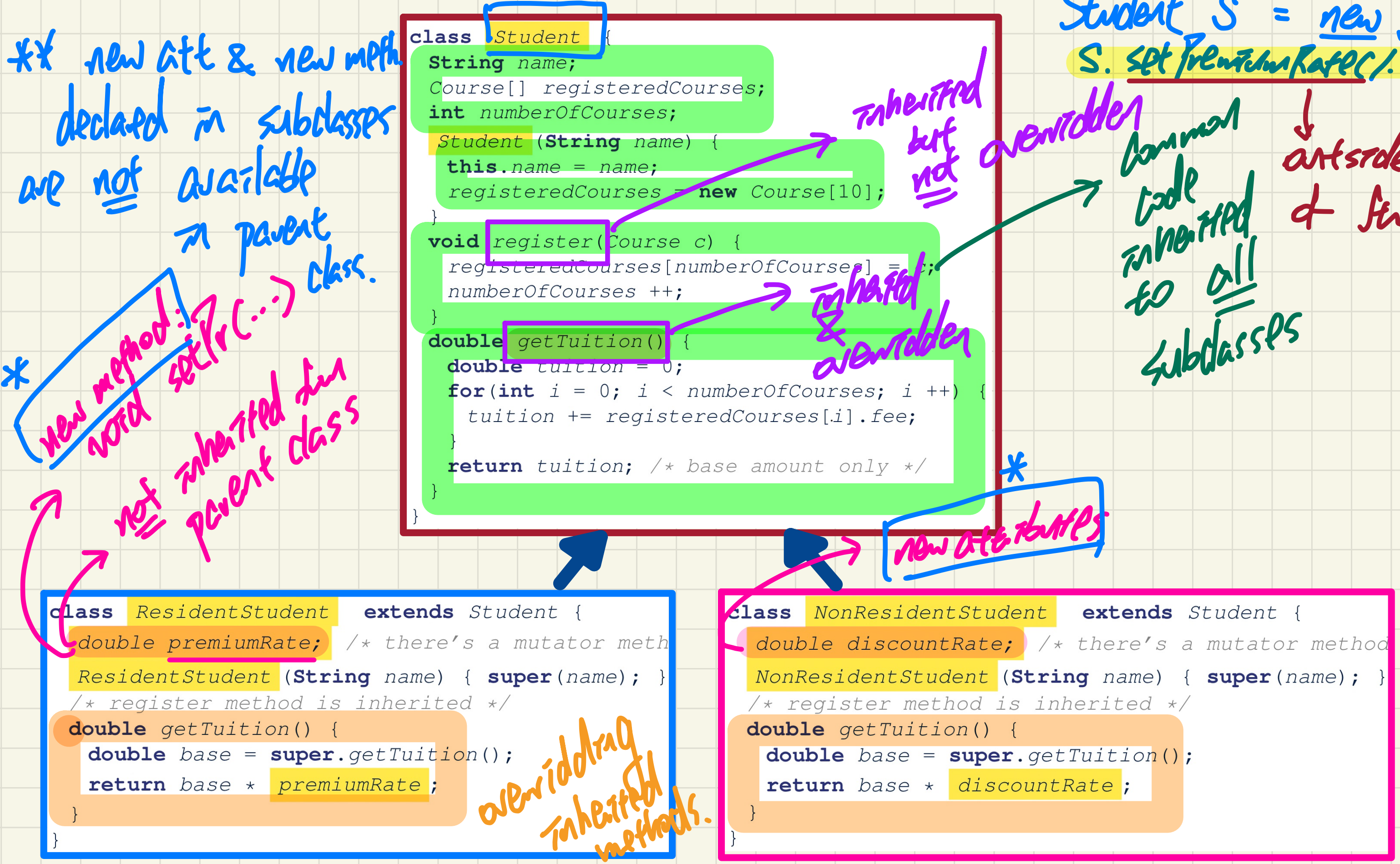
Inheritance

Code Reuse
Static Types & Expectation
Intuition: Polymorphism
Intuition: Dynamic Binding

Announcements

- ProgTest2: postponed to Tuesday, November 15
- Lab3 due today at 2pm

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 */
  }
}
```

```
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;
  }
}
```

```
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;
  }
}
```

** new att & new meth declared in subclasses are not available in parent class.

* new method! void setPr(...)

not inherited in parent class

inherited but not overridden

inherited & overridden

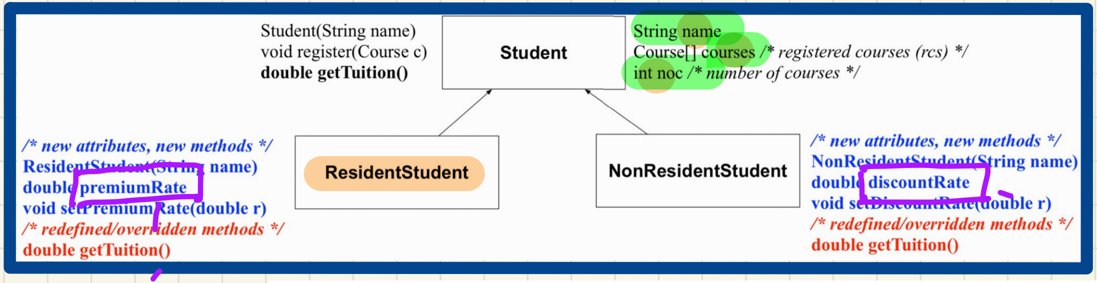
common code inherited to all subclasses

Student S = new Student(..);
S.setPremiumRate(1.25); X
outside expectation of Student.

* new attributes

overriding inherited methods.

Visualizing Parent and Child Objects



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

Student

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

ResidentStudent

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

NonResidentStudent

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

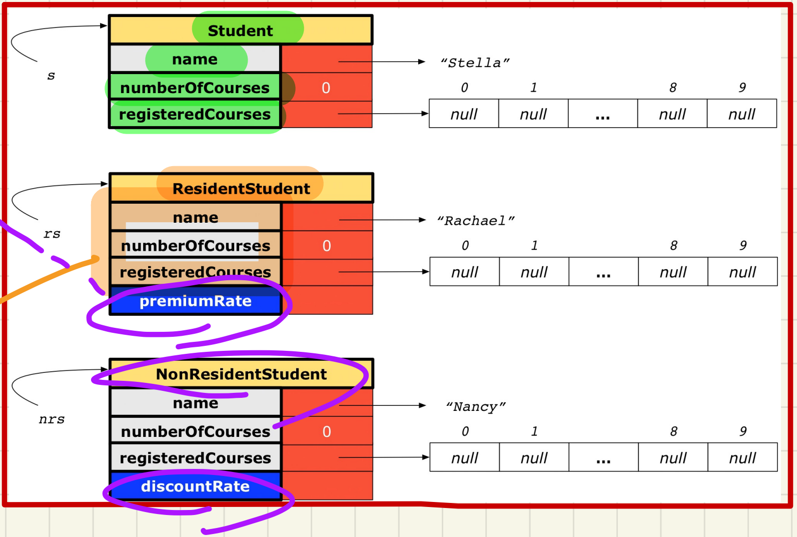
Inheritance Hirarchy

new attribute

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

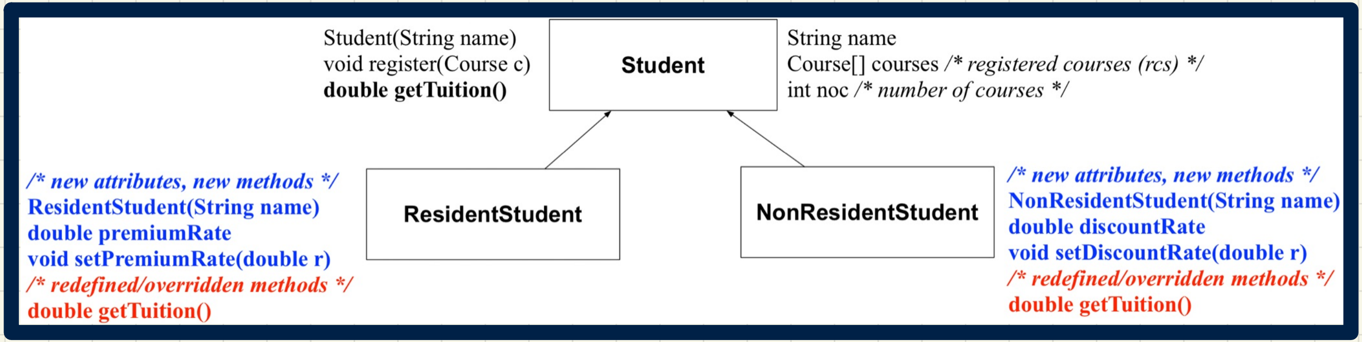
Declaring Static Types

Runtime Object Structure



Inherited from Student class

Testing Student Classes (with inheritance)



Student

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

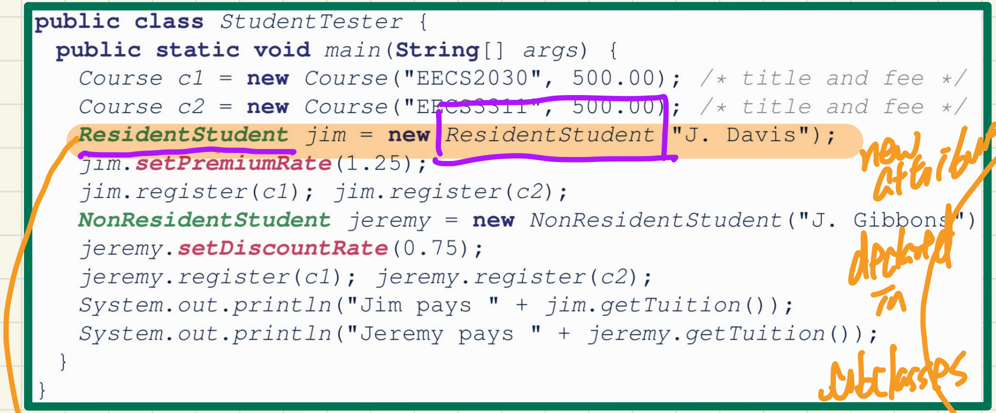
String name
Course[] courses /* *registered courses (rcs)* */
int noc /* *number of courses* */

ResidentStudent

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

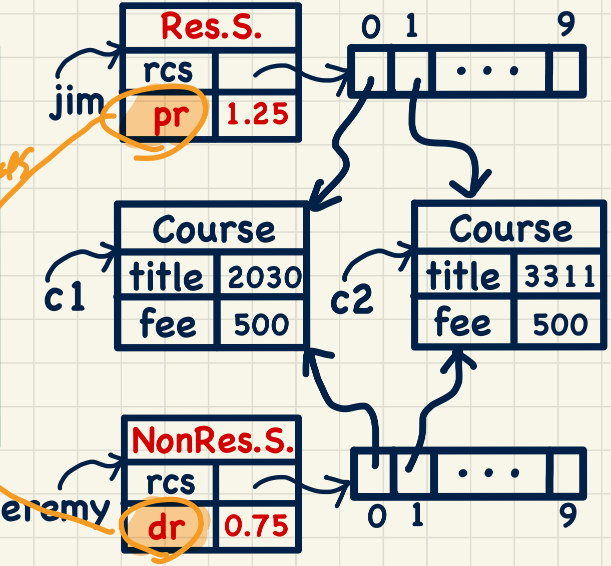
NonResidentStudent

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



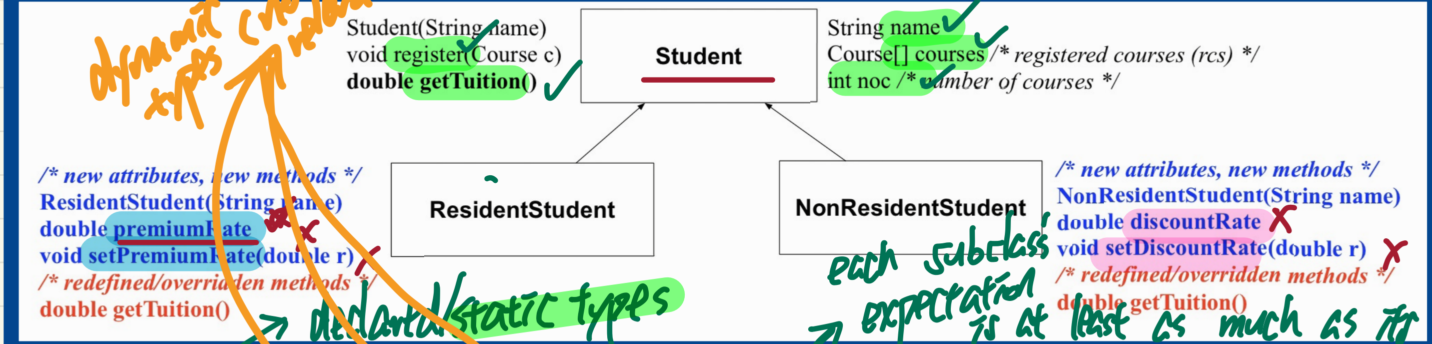
```
public class StudentTester {
  public static void main(String[] args) {
    Course c1 = new Course("EECS2030", 500.00); /* title and fee */
    Course c2 = new Course("EECS3311", 500.00); /* title and fee */
    ResidentStudent jim = new ResidentStudent("J. Davis");
    jim.setPremiumRate(1.25);
    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());
  }
}
```

new attributes declared in subclasses



what if: Student jim = new RS(..);

Student Classes (with inheritance): Expectations



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()

dynamic types (not relevant).

declared/static types

each subclass' expectation is at least as much as its parent

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

sibling classes share expec. from parent

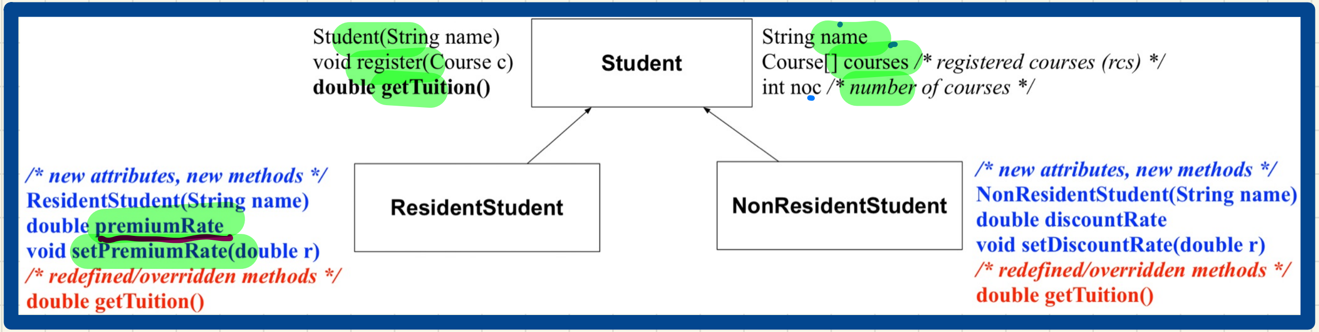
expectation

	name	rcs	noc	reg	getT	pr	setPR	dr	setDR
s.	✓	✓	✓	✓	✓	✗	✗	✗	✗
rs.	✓	✓	✓	✓	✓	✓	✓	✗	✗
nrs.	✓	✓	✓	✓	✓	✗	✗	✓	✓

beyond parent's expp. & no overlap.

Intuition: Polymorphism

② expectation on rs:
rs.setPremiumRate(1.25).



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

Student

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

ResidentStudent

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

NonResidentStudent

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

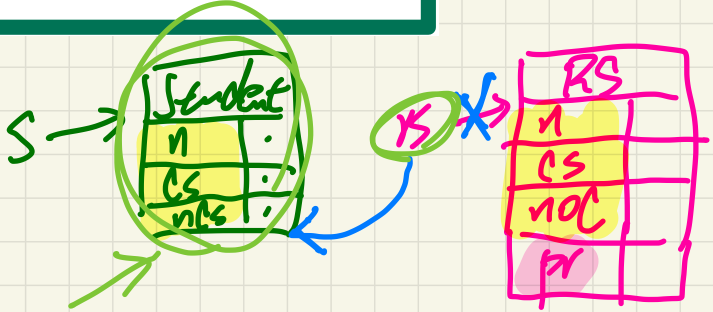
crash

↳ rs = s should be invalid

```
Student s = new Student("Stella");
ResidentStudent rs = new ResidentStudent("Rachael");
rs.setPremiumRate(1.25);
s = rs; ✓/* Is this valid? */
rs = s; ✗/* Is this valid? */
```

Assume rs = S was valid

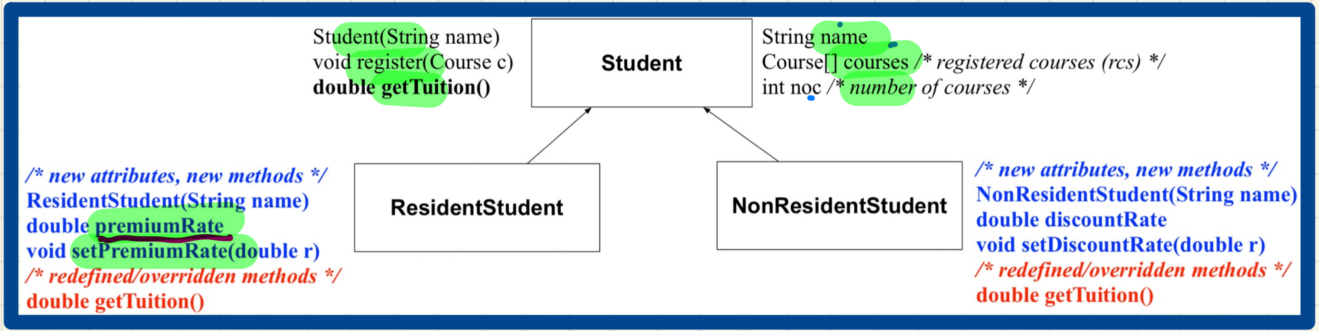
↳ ① executing the assignment points rs to a student obj.



Intuition: Polymorphism

② expectation on rs:
rs.setPremiumRate(1.25).

↳ crash



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

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

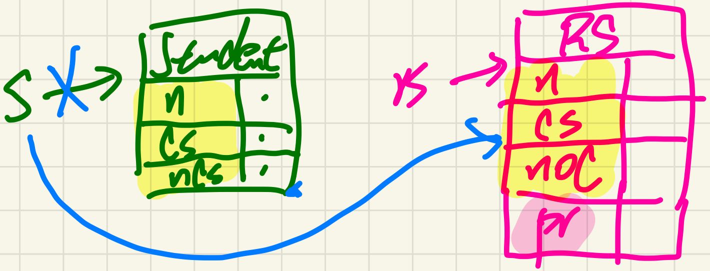
/ new attributes, new methods */*
ResidentStudent(String name)
double premiumRate
void setPremiumRate(double r)
/ redefined/overridden methods */*
double getTuition()

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

↳ rs = s should be invalid

```
Student s = new Student("Stella");
ResidentStudent rs = new ResidentStudent("Rachael");
rs.setPremiumRate(1.25);
s = rs;✓/* Is this valid? */
rs = s;✗/* Is this valid? */
```

③ type casting can make this work



① s.setPremiumRate(1.25)

↳ not valid ∵ ST of s (Student) does not declared pr.

C_1 obj1;
C_2 obj2;
⋮

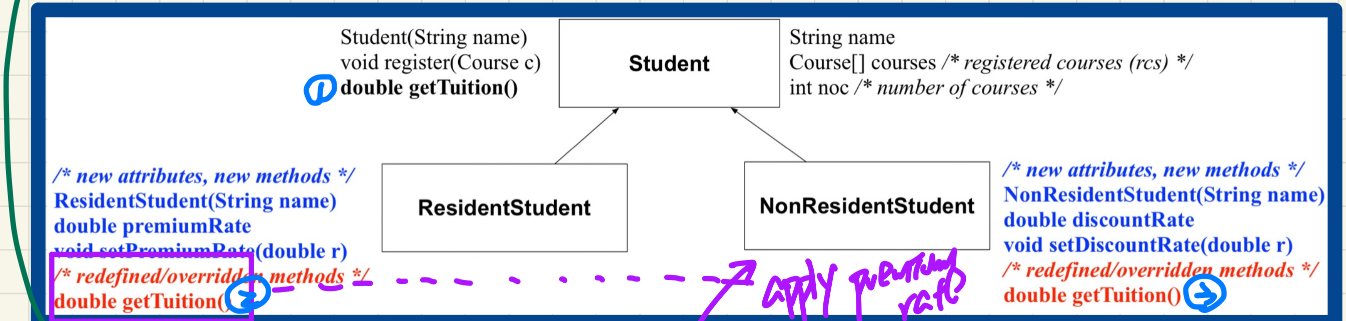
obj1 = obj2

↳ to be valid the ST of obj2 (C_2) should be a subclass of the ST of obj1 (C_1).

subclass ↓ descendants class.

s: ST expectation determined by ST of C.O.

Intuition: Dynamic Binding



Student(String name)
void register(Course c)
1 double getTuition()
Student
String name
Course[] courses /* registered courses (rcs) */
int noc /* number of courses */
/* new attributes, new methods */
ResidentStudent(String name)
double premiumRate
void setPremiumRate(double r)
/* redefined/overridden methods */
double getTuition() 2
ResidentStudent
NonResidentStudent
apply premium rate
/* new attributes, new methods */
NonResidentStudent(String name)
double discountRate
void setDiscountRate(double r)
/* redefined/overridden methods */
double getTuition() 3

```
Course eecs2030 = new Course("EECS2030", 100.0);
Student s;
ResidentStudent rs = new ResidentStudent("Rachael");
NonResidentStudent nrs = new NonResidentStudent("Nancy");
rs.setPremiumRate(1.25); rs.register(eecs2030);
nrs.setDiscountRate(0.75); nrs.register(eecs2030);
s = rs; System.out.println(s.getTuition());
s = nrs; System.out.println(s.getTuition());
```

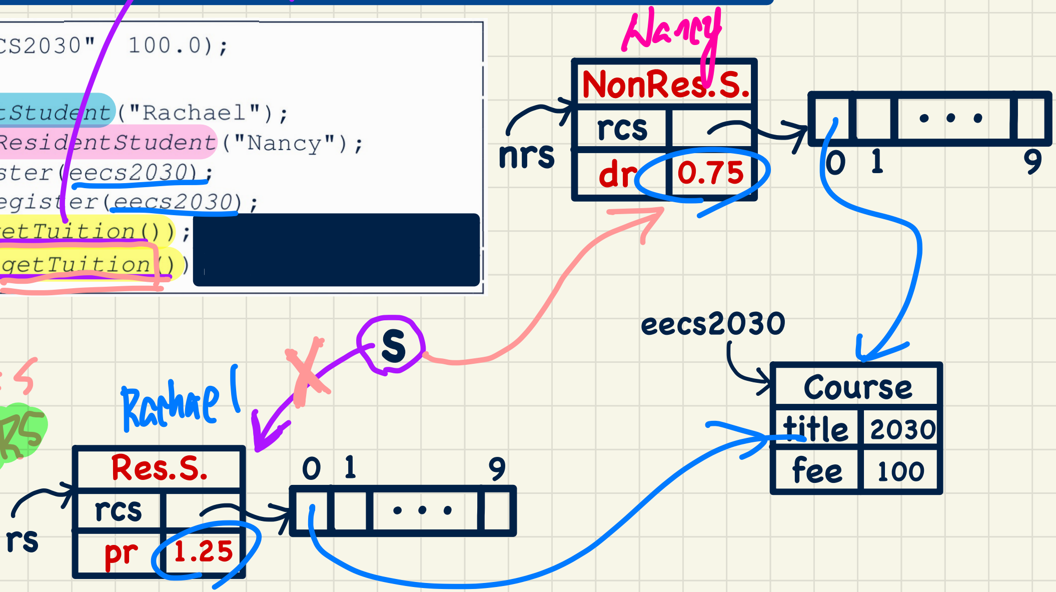
[redacted]

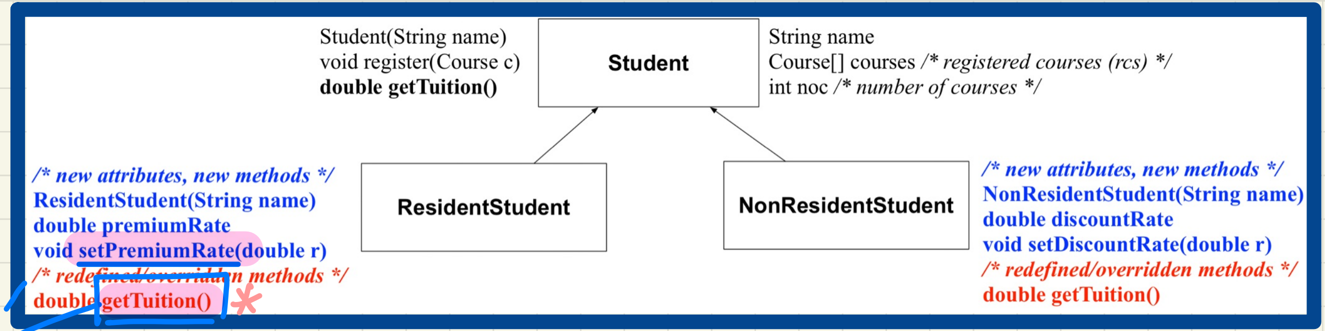
ST

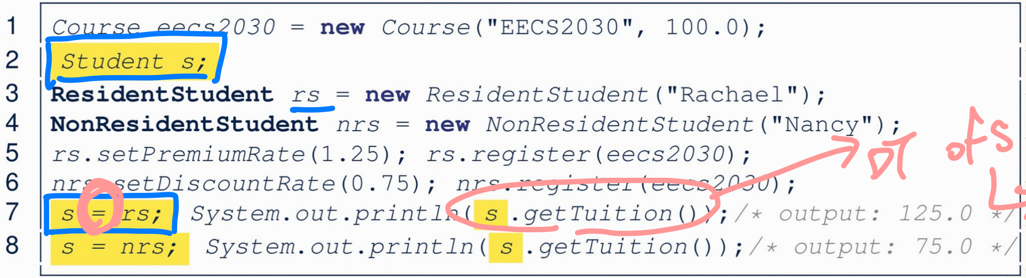
changes the dynamic type of s from RS to NRS

Dynamic type of s becomes RS

DT of s is NRS




Student(String name)
void register(Course c)
double getTuition()
Student
String name
Course[] courses /* registered courses (rcs) */
int noc /* number of courses */
/* new attributes, new methods */
ResidentStudent(String name)
double premiumRate
void setPremiumRate(double r)
/* redefined/overridden methods */
double getTuition()
ResidentStudent
NonResidentStudent
/* new attributes, new methods */
NonResidentStudent(String name)
double discountRate
void setDiscountRate(double r)
/* redefined/overridden methods */
double getTuition()
*



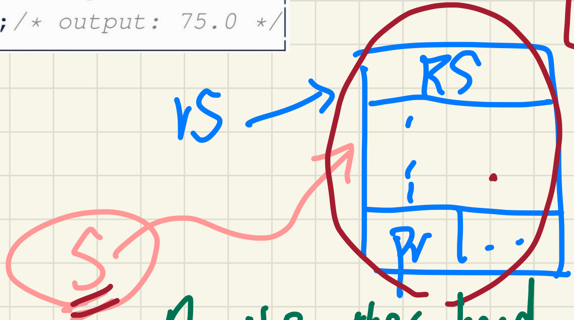
```java
Course eecs2030 = new Course("EECS2030", 100.0);
Student s;
ResidentStudent rs = new ResidentStudent("Rachael");
NonResidentStudent nrs = new NonResidentStudent("Nancy");
rs.setPremiumRate(1.25); rs.register(eecs2030);
nrs.setDiscountRate(0.75); nrs.register(eecs2030);
s = rs; System.out.println( s .getTuition());/* output: 125.0 */
s = nrs; System.out.println( s .getTuition());/* output: 75.0 */
```

✓

DT of s is RS

↳ version of getT() in RS will be invoked

```
getTuition ( ) {
  :
  this.setPremiumRate(..);
}
```



↳ implicitly call setP on RS object

On the other hand:
s.setPremiumRate X

PointV1 p1 = ...

PointV2 p2 = ...

① assertEquals(p1, p2)

↳ p1.equals(p2) → invoke default version in Object

↳ p1 == p2

② p1 == p2

1. Whether a line should compile?

Look at static type

2. Which version of method should be invoked?

Look at dynamic type