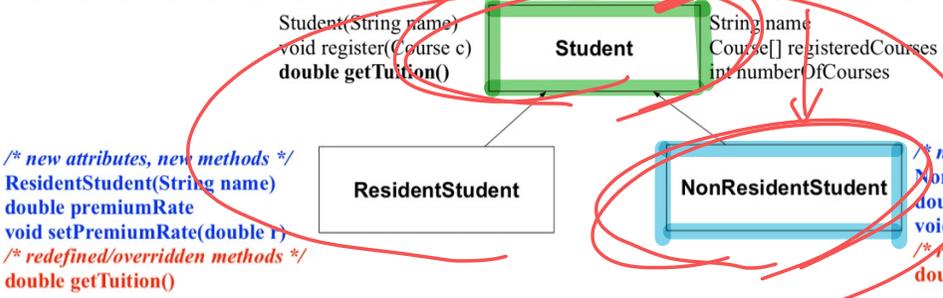


LECTURE 18

MONDAY NOVEMBER 11

Static Types determine Expectations

Inheritance Hierarchy: Students



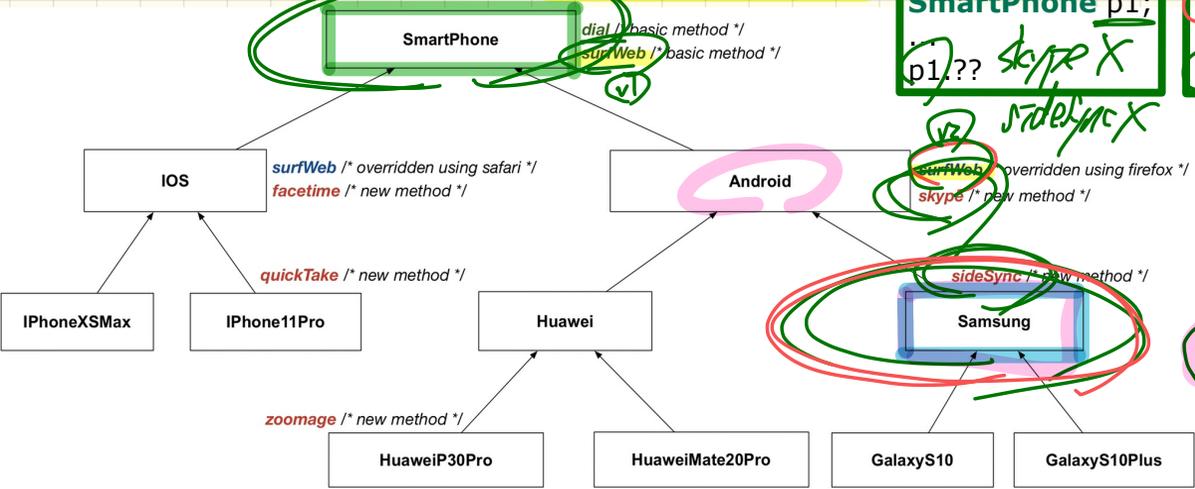
```

Declare:
Student jim;
...
jim ??
  
```

```

Declare:
NRS alan;
...
alan ??
  
```

Inheritance Hierarchy: Smart Phones



```

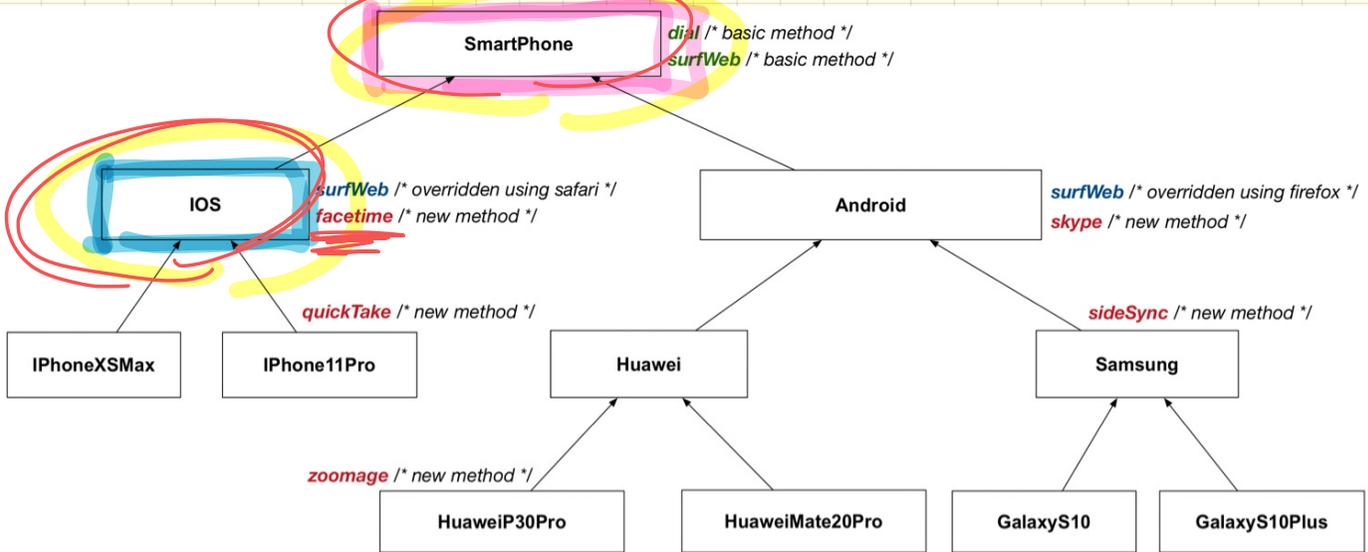
Declare:
SmartPhone p1;
...
p1 ??
  
```

```

Declare:
Samsung p2;
...
p2 ??
  
```

skype
sideSync
p2. surfWeb

Rules of Substitutions



Declarations:

`IOS sp1;`
`SmartPhone sp2;`

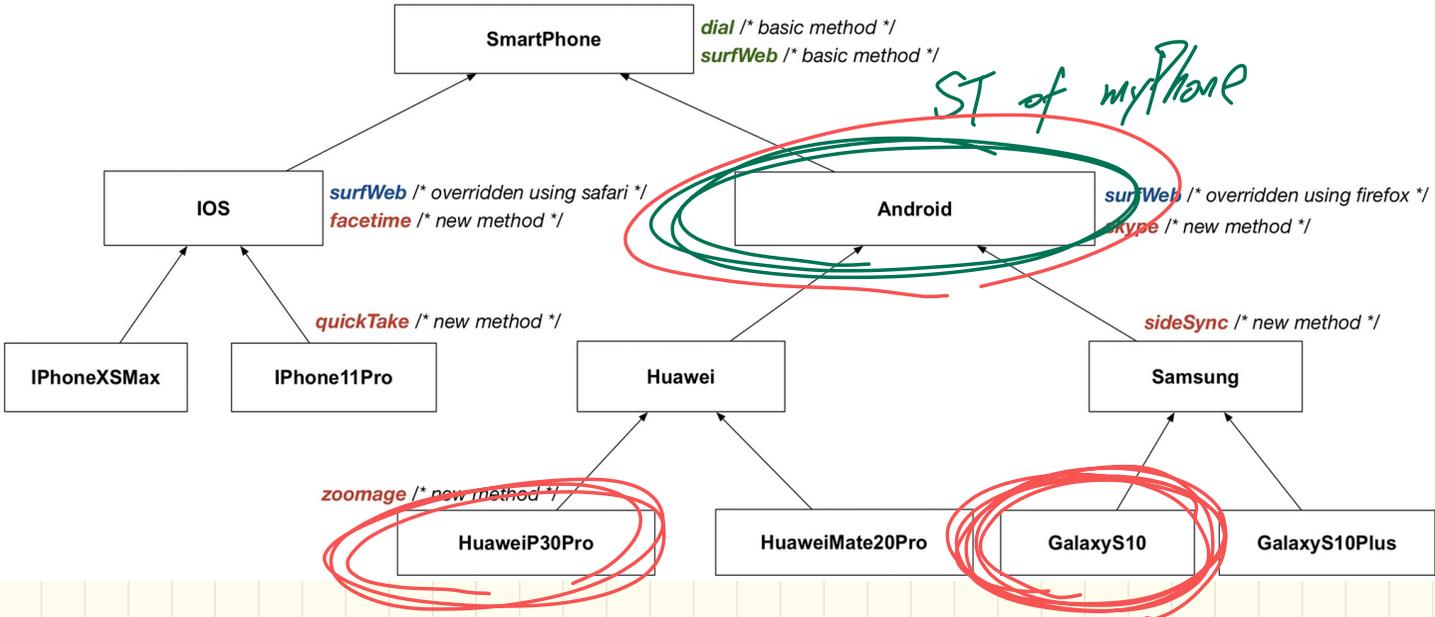
Substitutions:

`sp1 = sp2;`

Can **ST** of sp2 fulfill what's expected on **ST** of sp1?

NO
facetime not supported on sp2.

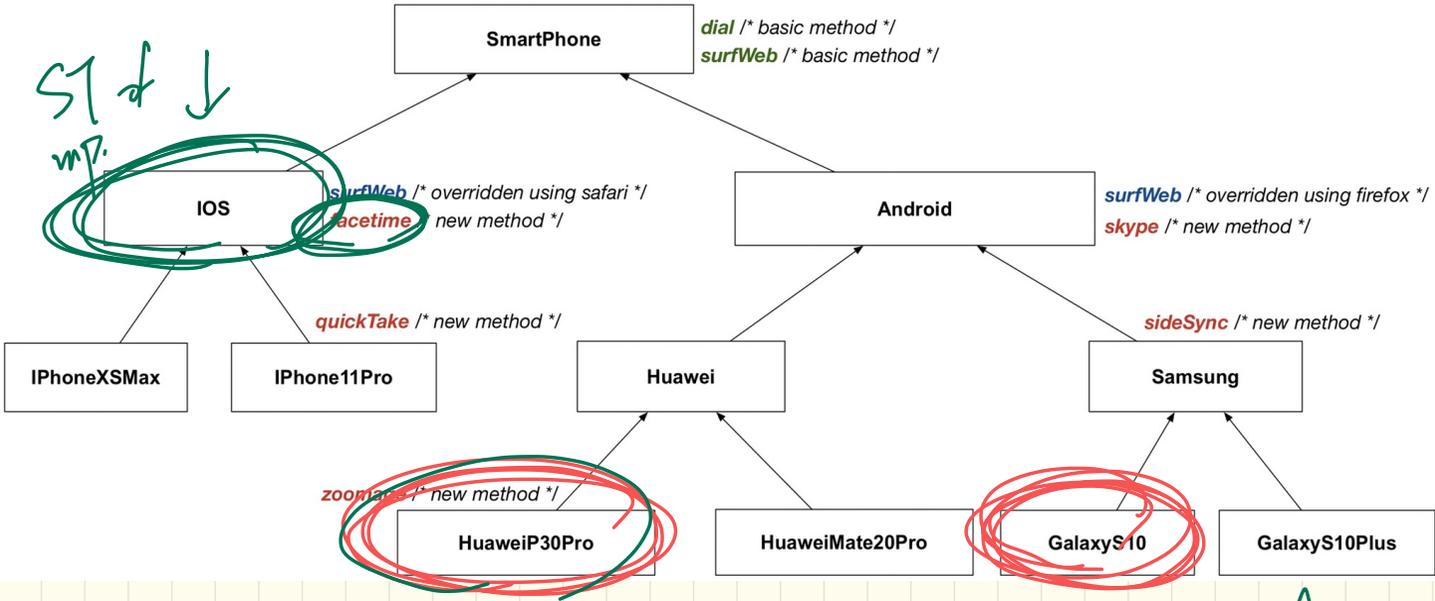
Change of **Dynamic** Type: Exercise (1)



Exercise 1:

```
Android myPhone = new HuaweiP30Pro(...);  
myPhone = new GalaxyS10(...);
```

Change of **Dynamic** Type: Exercise (2)



Exercise 2:

```

IOS myPhone = new HuaweiP30Pro(...); ①
myPhone = new GalaxyS10(...); ②
  
```

e.g. facetime not supported.
f.t. not supported.

Change of **Dynamic** Type (2.1)

```

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

base (V1)



before ID: Jim's DT is Student

after	ST of Jim	DT of
1	Student	Jim
2		RS
3		RS
4		NRS
		NRS

```

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

→ pr (V2)



```

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

← dr (V3)

Given:

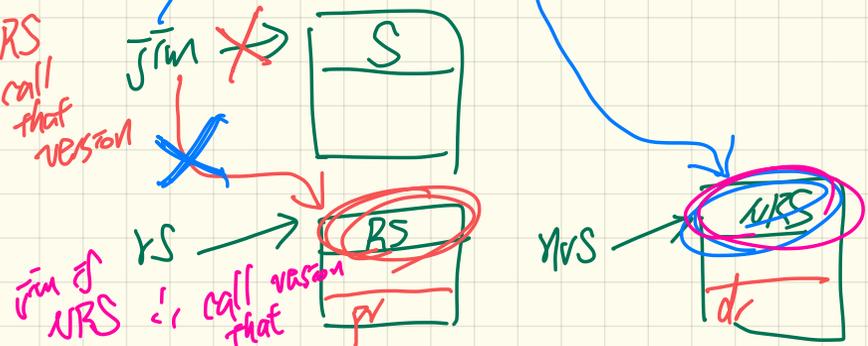
```

Student jim = new Student(...);
ResidentStudent rs = new ResidentStudent(...);
NonResidentStudent nrs = new NonResidentStudent(...);
    
```

Example 1:

```

jim = rs; ①
println(jim.getTuition()); ②
jim = nrs; ③
println(jim.getTuition()); ④
    
```



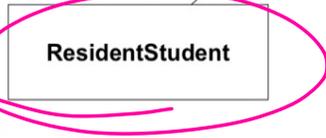
Change of **Dynamic** Type (2.2)

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



```
String name  
Course[] registeredCourses  
int numberOfCourses
```

```
/* 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()
```

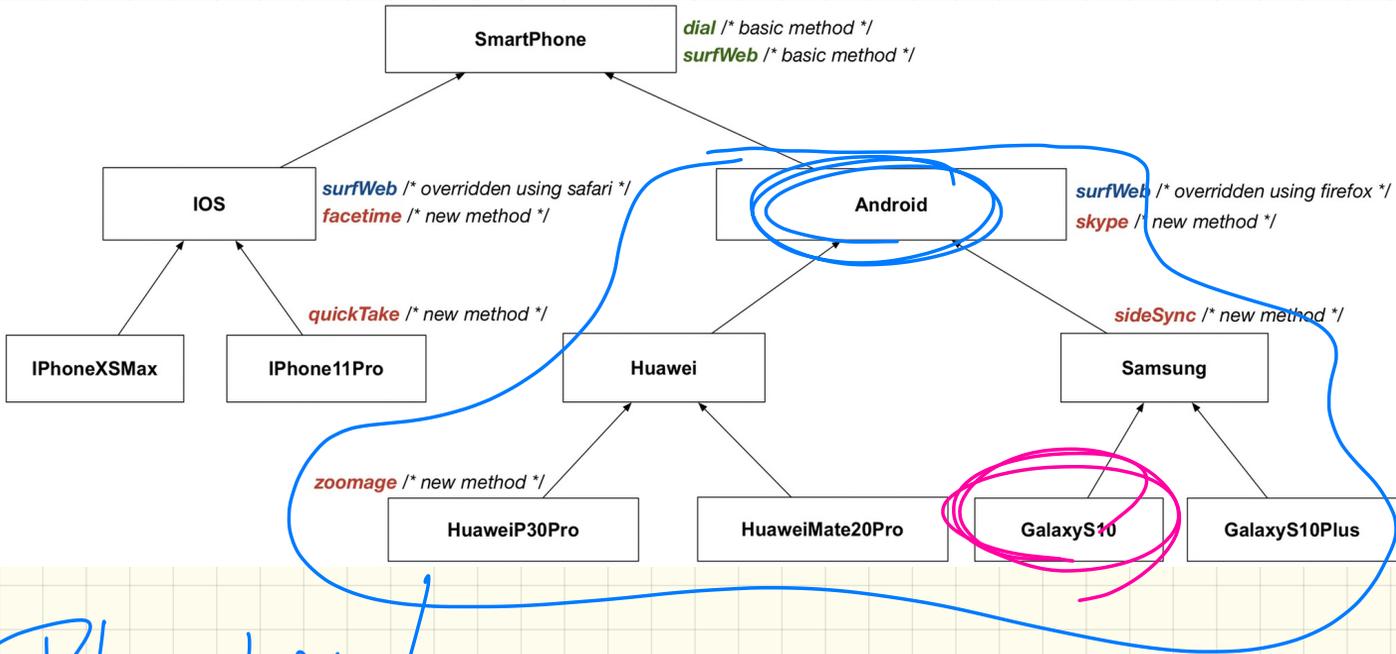
Given:

```
Student jim = new Student(...);  
ResidentStudent rs = new ResidentStudent(...);  
NonResidentStudent nrs = new NonResidentStudent(...);
```

Example 2:

```
rs = jim;  
println(rs.getTuition());  
nrs = jim;  
println(nrs.getTuition());
```

cannot be executed
because the previous line does not compile.



Polymorphism

Android myPhone = ? ;

all valid types of objects for substituting Android.

new G(S10());

Type Cast: Motivation

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

Student

String name
Course[] registeredCourses
int numberOfCourses

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

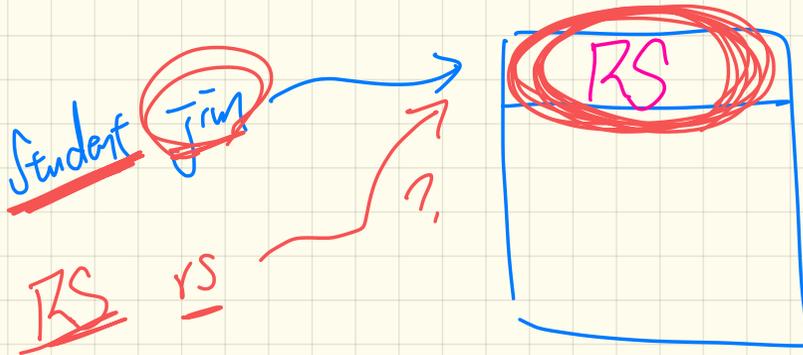
ResidentStudent

NonResidentStudent

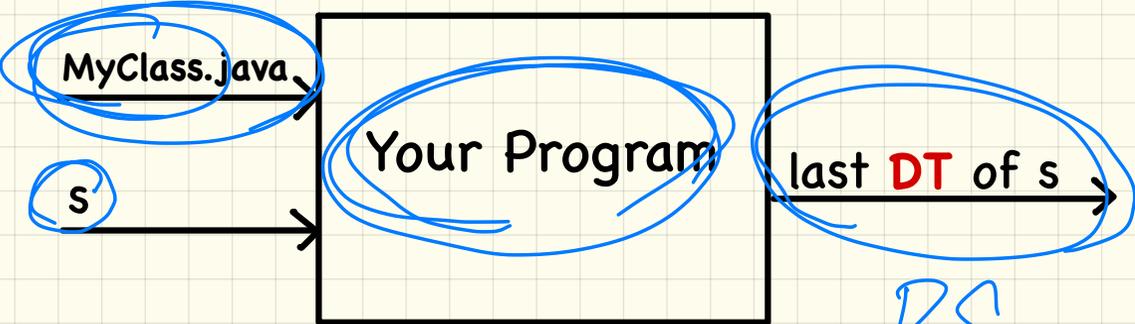
/ new attributes, new methods */*
NonResidentStudent(String name)
double discountRate
void setDiscountRate(double r)
/ redefined/overridden methods */*
double getTuition()

```
1 Student jim = new ResidentStudent ("J. Davis");  
2 ResidentStudent rs = (jim);  
3 rs.setPremiumRate(1.5);
```

dynamically points to RS



An A+ Challenge: Inferring the DT of a Variable

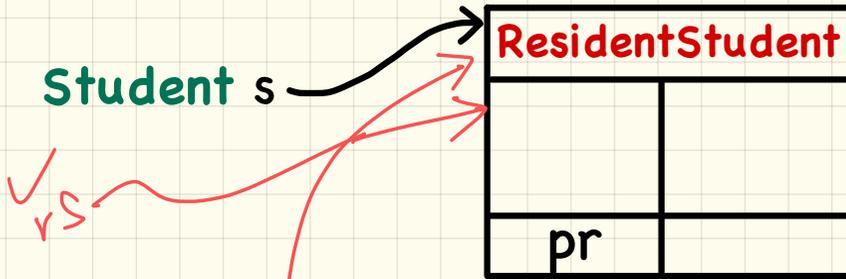


```
class MyClass {  
    main (...)  
    Student s = ...;  
    ...  
    s = new ResidentStudent(...);  
}
```

RS
undecidable
B
while (true) {
 ...
}

Anatomy of a Type Cast (1)

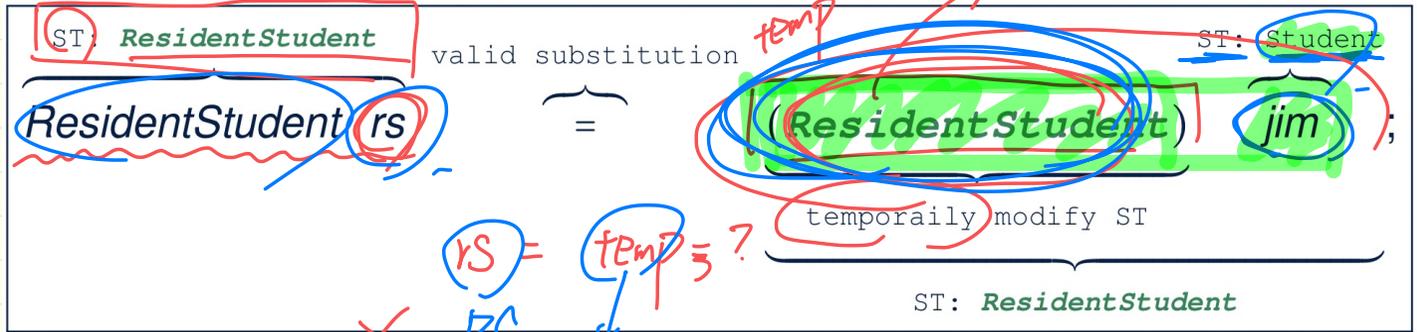
```
Student s = new ResidentStudent("Jim");
```



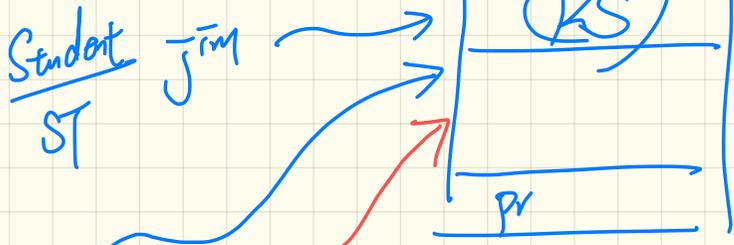
RS temp

```
ResidentStudent rs = (ResidentStudent) s;
```

Anatomy of a Type Cast (2)



$rs = jim$ ✗
 $rs = temp$?
 $ST: rs$



- Purging a cast
1. No new object created
 2. ST of `jim` not modified
 3. a temp. obj's of ST `RS` is created

Type Cast

↳ change the expectation

1. does the cast compile?

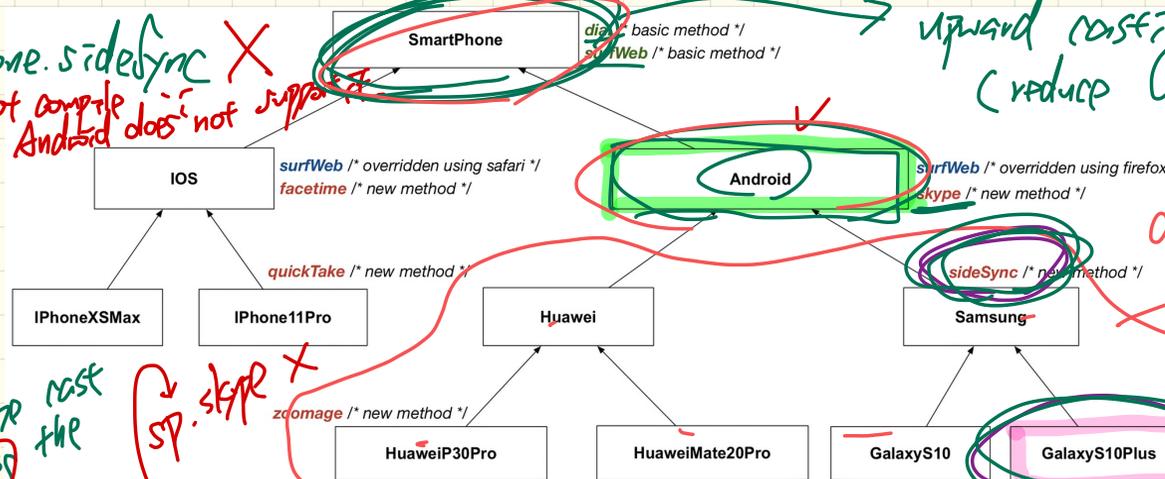
2. if the cast compiles,
does it cause an

Exception (ClassCastException)

Compilable Type Casts: Upwards vs. Downwards

myPhone.sideSync X
not compile ST Android does not support

upward casting.
(reduce expectations)



downward casting
? wider expect.

A type cast changes the compiler's expectations.

SP: skype X

expectations? (ST).

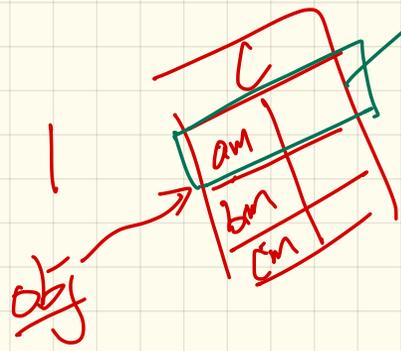
Expectations

```

Android myPhone = new GalaxyS10Plus();
SmartPhone sp = (SmartPhone) myPhone;
GalaxyS10Plus ga = (GalaxyS10Plus) myPhone;
  
```

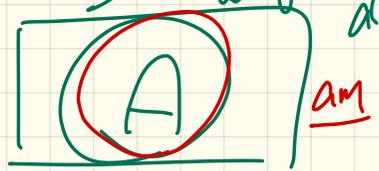
upward cast

	sp	myPhone	ga
dial	✓	✓	✓
surfWeb	✓	✓	✓
skype	X	✓	✓
sideSync	X	X	✓
facetime	X	X	X
quickTake	X	X	X
zoomage	X	X	X

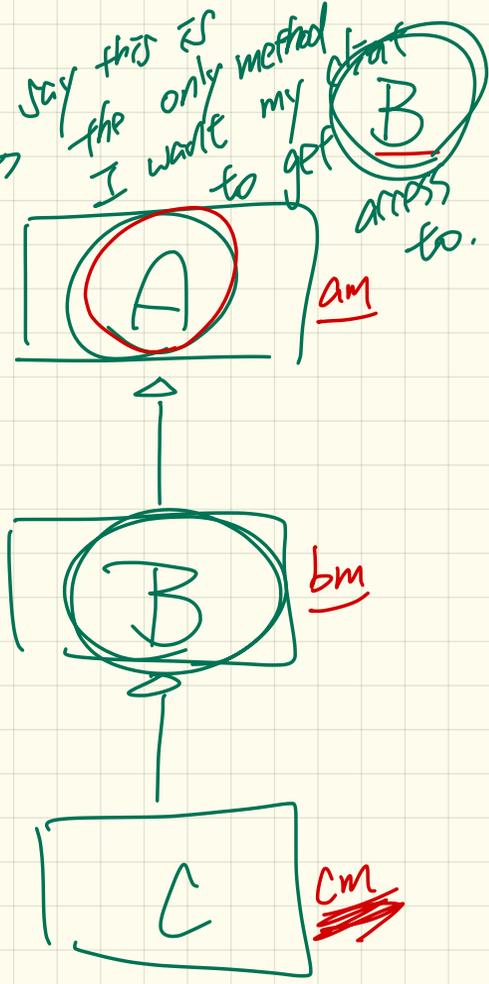
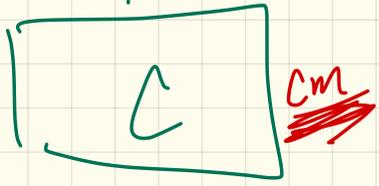
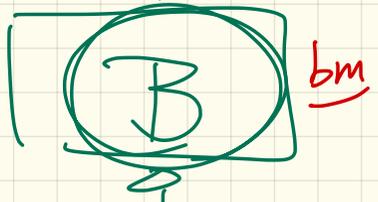


say this is the only method I want to get access to.

obj = new C();



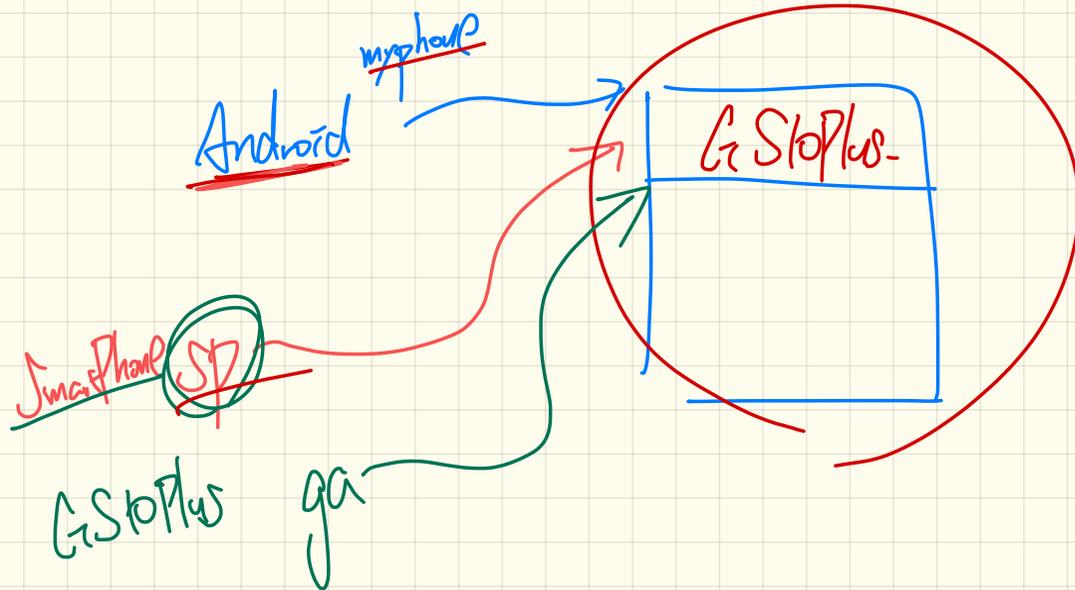
(A) client-obj = (A)
 client-obj.cm?



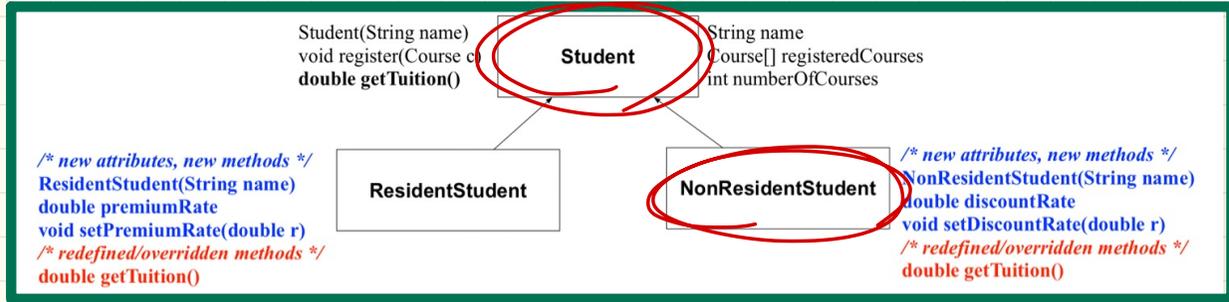
```
Android myPhone = new GalaxyS10Plus();
```

```
SmartPhone sp = (SmartPhone) myPhone;
```

```
GalaxyS10Plus ga = (GalaxyS10Plus) myPhone;
```



Compilable Type Cast May Fail at Runtime (1)



```

1 Student jim = new NonResidentStudent("jim");
2 ResidentStudent rs = ((ResidentStudent) jim);
3 rs.setPremiumRate(1.5);
  
```

cast version of jim of

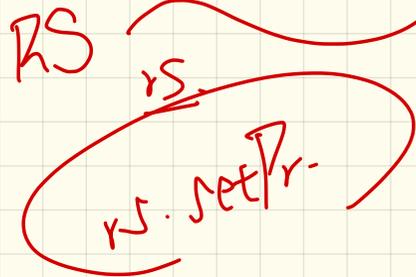
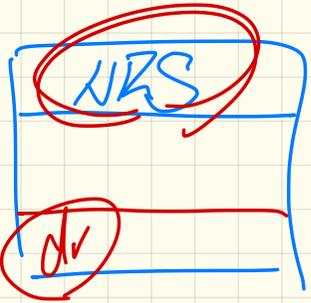
ResidentStudent rs = jim ✓
 rs = cast version ✓
 valid downward casting!

ST: Student
 ST: RS

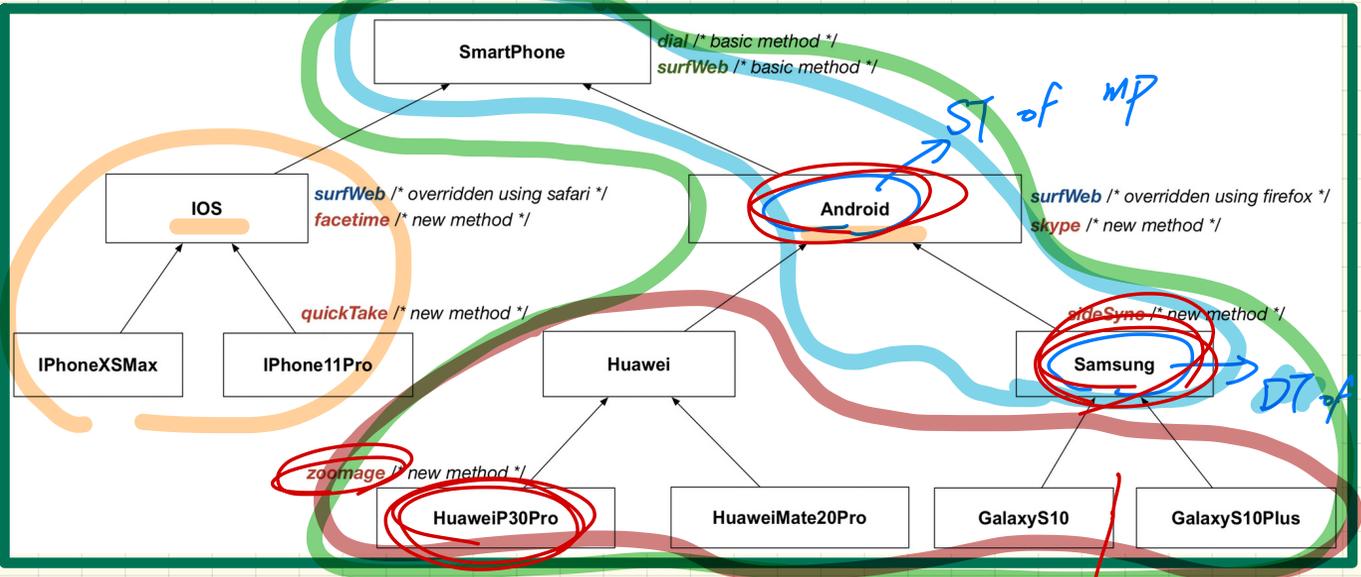
```
1 Student jim = new NonResidentStudent("J. Davis");  
2 ResidentStudent rs = (ResidentStudent) jim; ✓  
3 rs.setPremiumRate(1.5);
```

class cast
exception.

Student jim



Compilable Cast vs. Exception-Free Cast



```
Android myPhone = new Samsung();
```

downward casting
but DT of mp
cannot support it.

Compilable Casts



Exception-Free Casts



Non-Compilable Casts



ClassCastException

