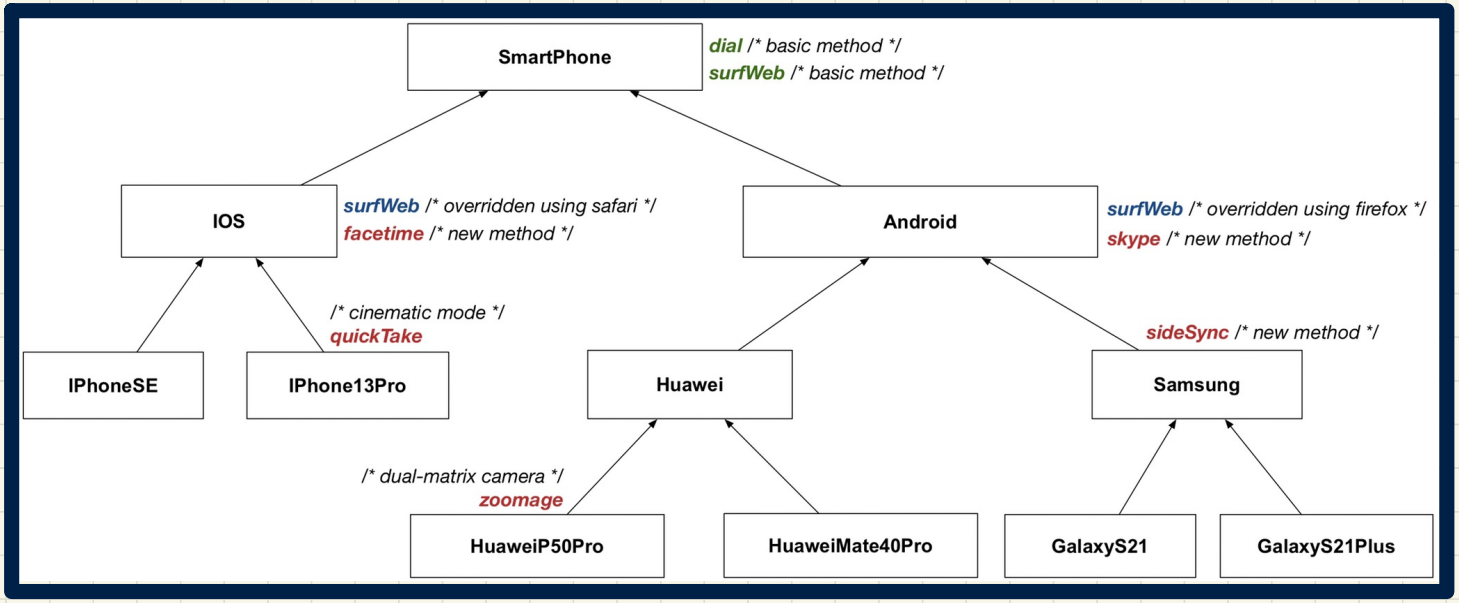
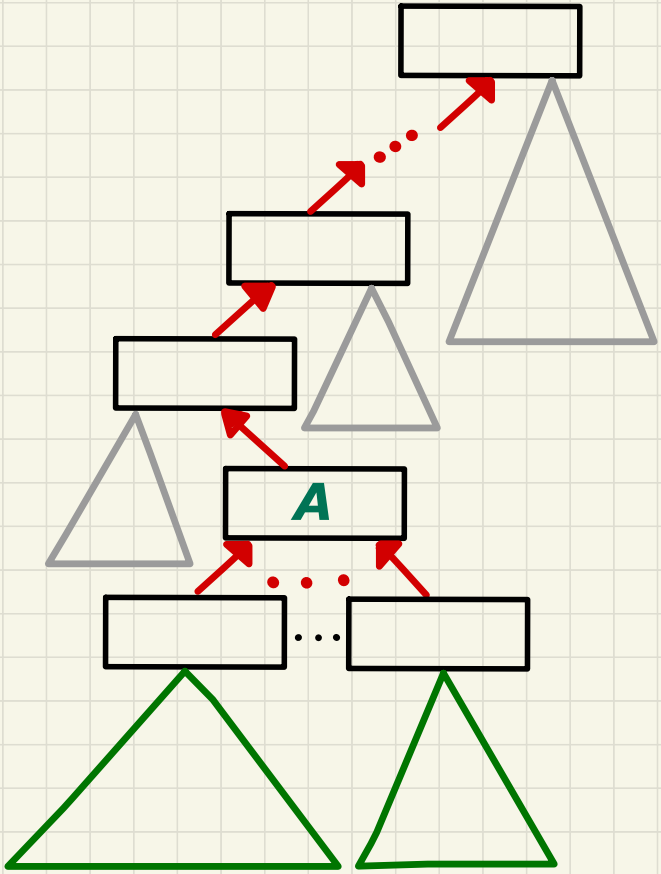
Multi-Level Inheritance Hierarchy: Smartphones



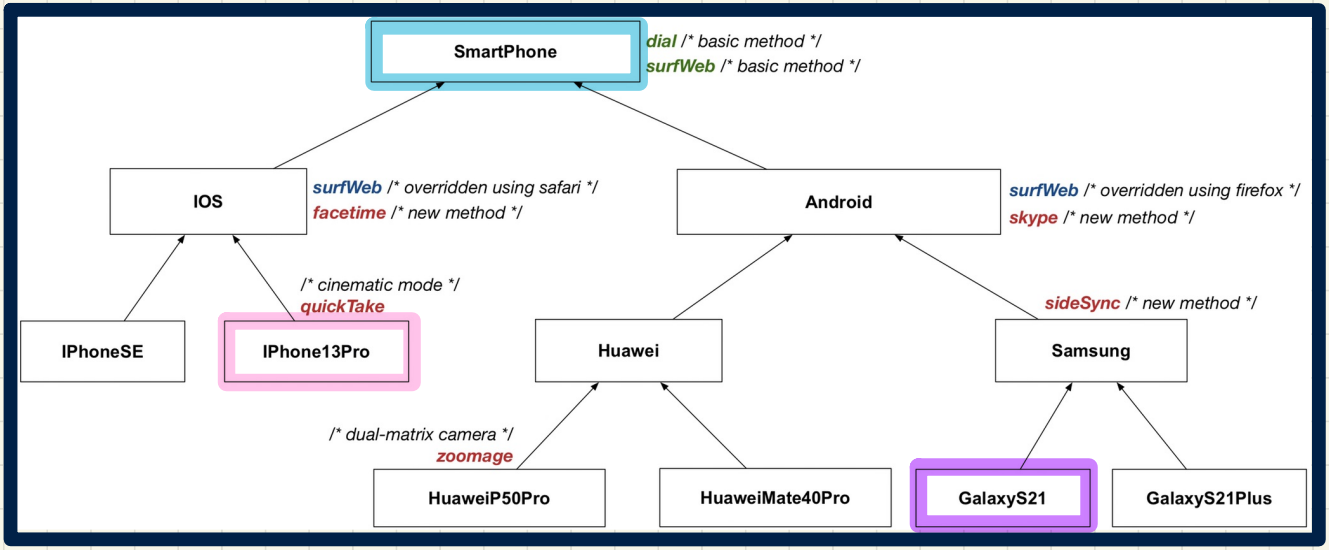
Exercise Compare the ranges of expectations of:

+ IPhone13Pro
+ HuaweiP50Pro
+ GalaxyS21Plus

Inheritance Forms a Type Hierarchy

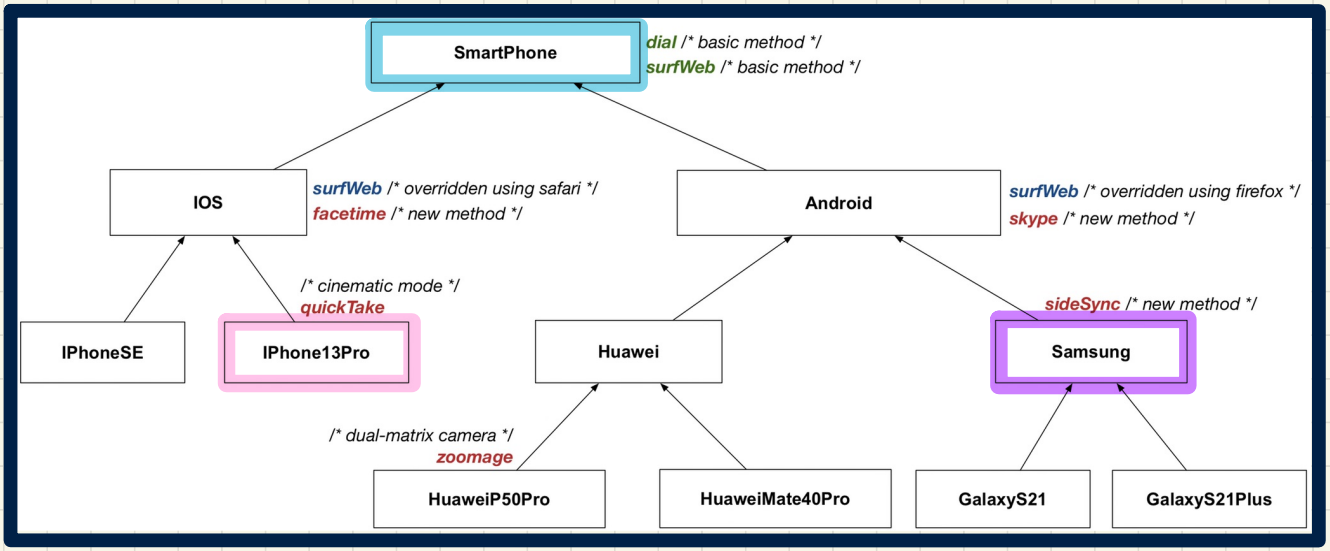


Inheritance Accumulates Code for Reuse



	ancestors	expectations	descendants

Inheritance Accumulates Code for Reuse

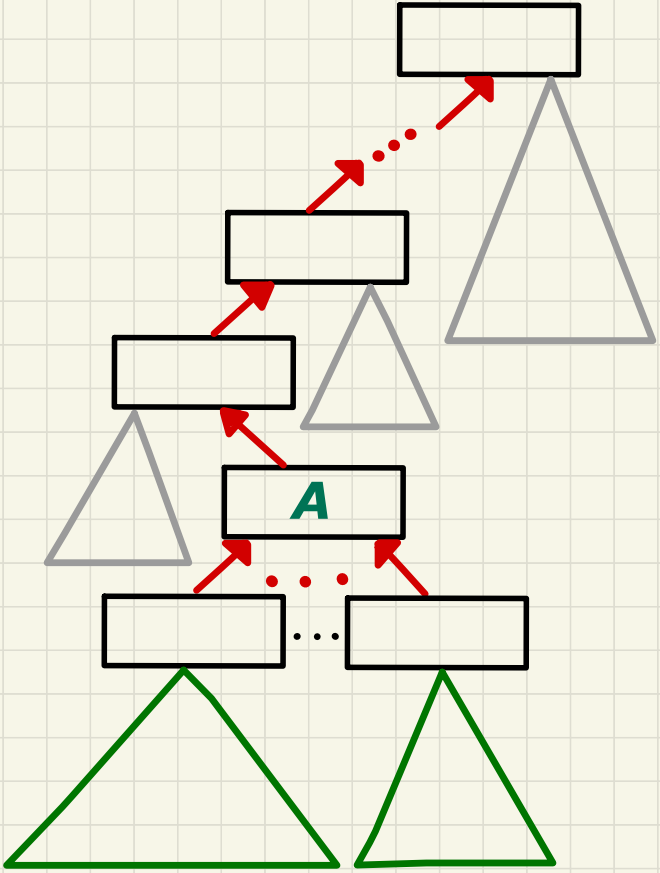



```
SmartPhone sp1;
IPhone13Pro sp2;
Samsung sp3;
```

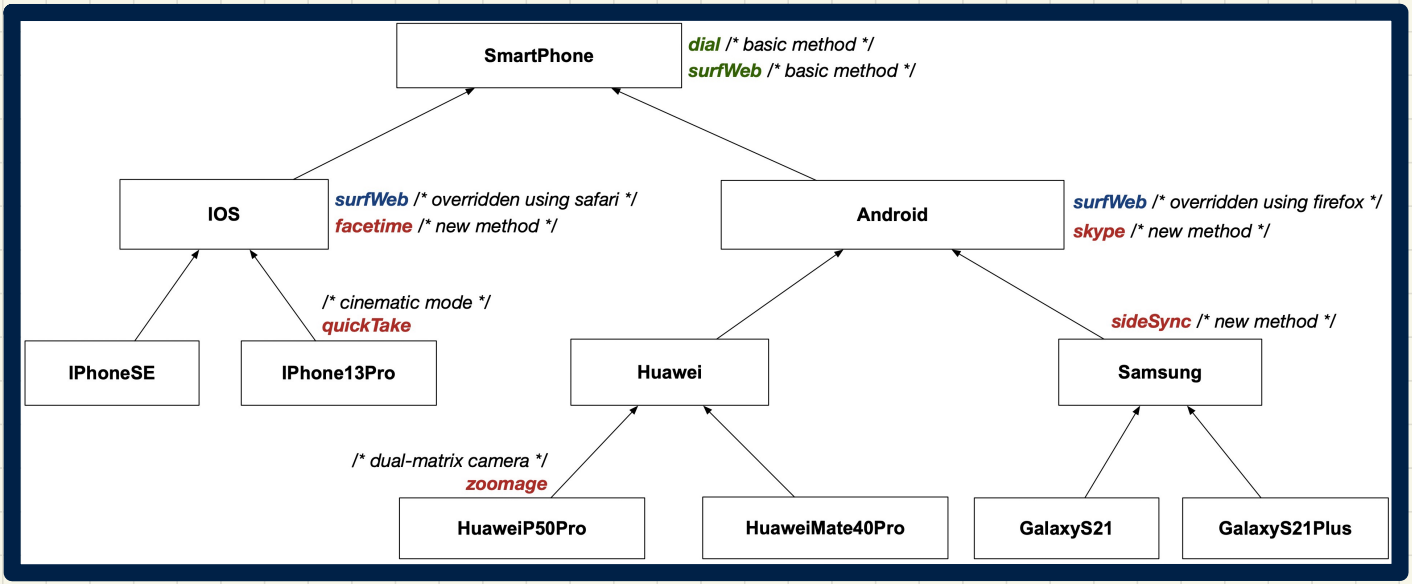
```
sp1 = ?;
sp2 = ?;
sp3 = ?;
```

Rules of Substitutions

```
A oa = ...;
? ob = ...;
oa = ob;
```



Rules of Substitutions (1)



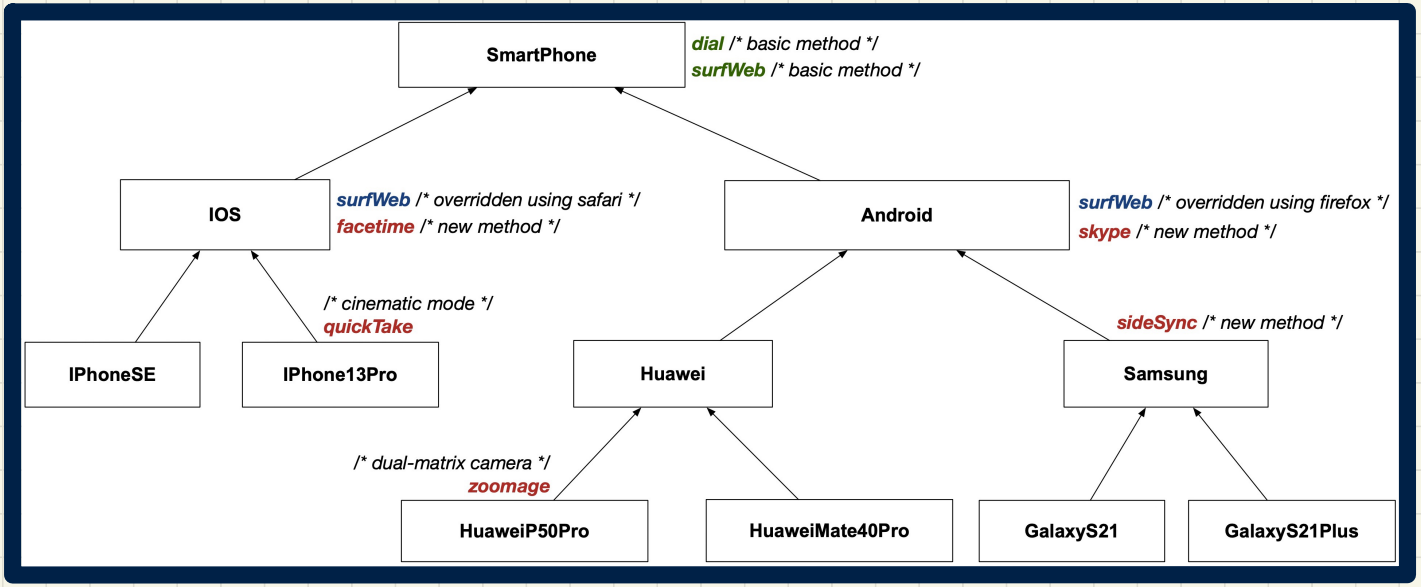
Declarations:

```
IOS sp1;
IPhoneSE sp2;
IPhone13Pro sp3;
```

Substitutions:

```
sp1 = sp2;
sp1 = sp3;
```

Rules of Substitutions (2)



Declarations:

```
IOS sp1;
SmartPhone sp2;
```

Substitutions:

```
sp1 = sp2;
```

Visualization: Static Type vs. Dynamic Type

Declaration:

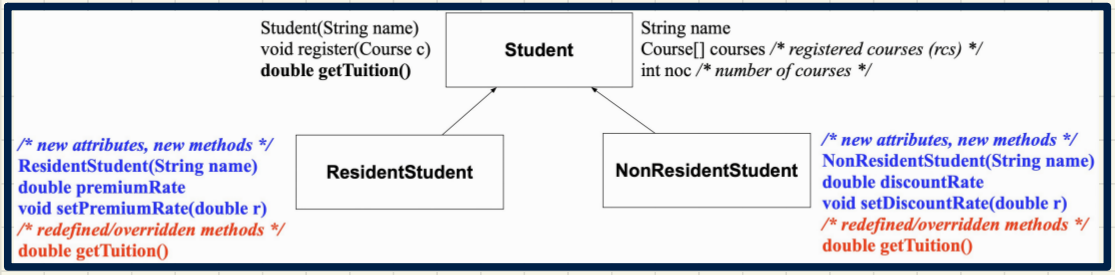
Student s;

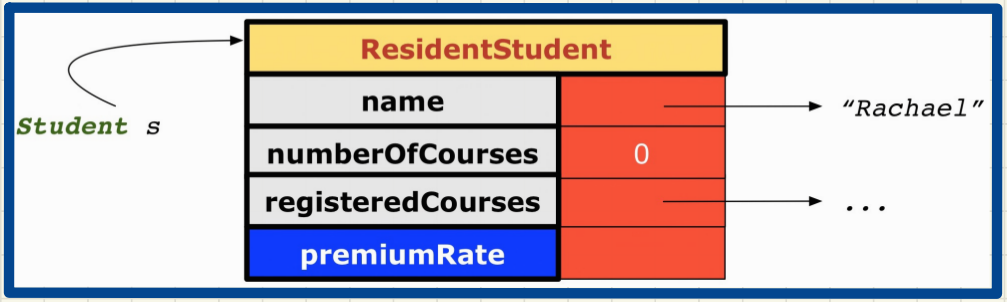
Substitution:

s = **new ResidentStudent**("Rachael");

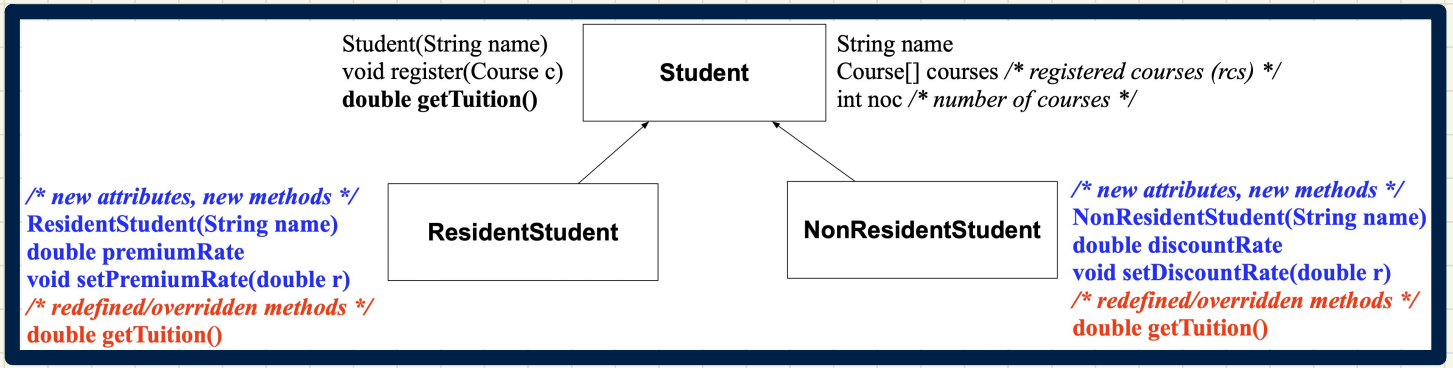
Static Type: Expectation

Dynamic Type: Accumulation of Code





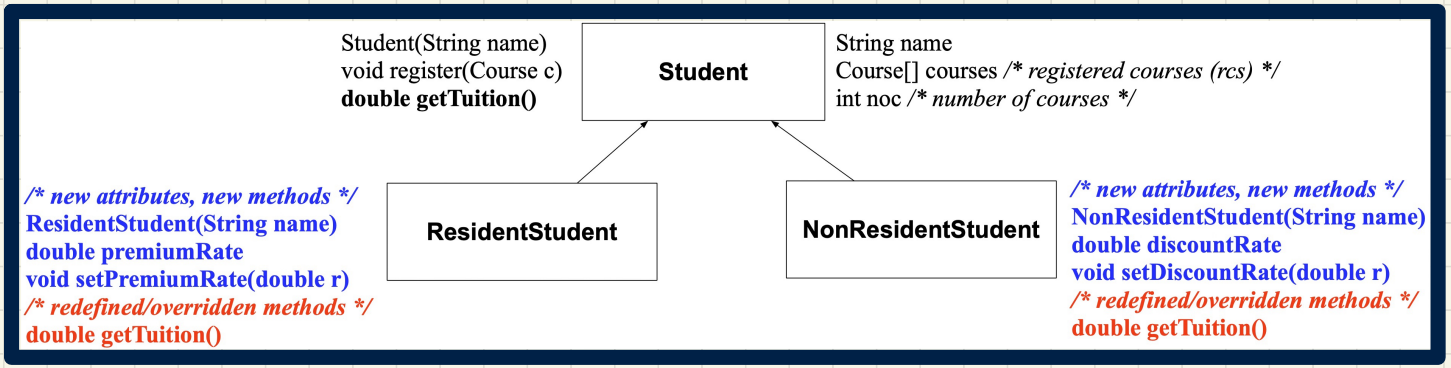
Change of Dynamic Type (1.1)



Example 1:

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

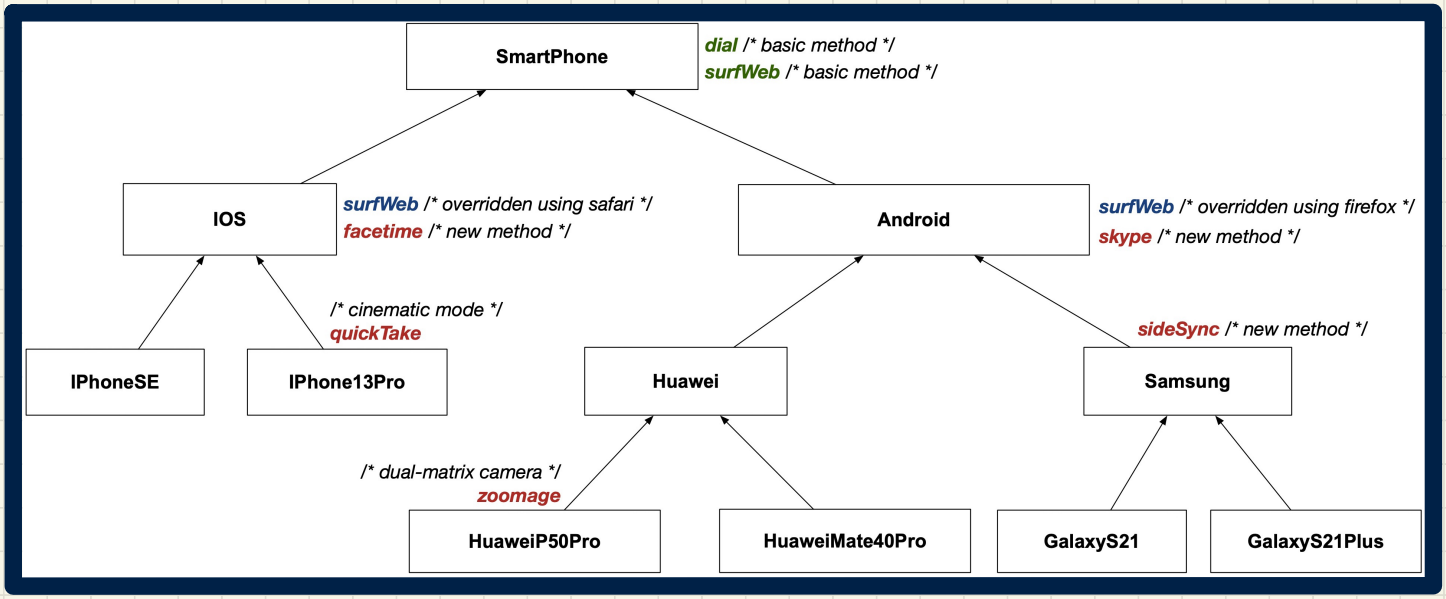
Change of Dynamic Type (1.2)



Example 2:

```
ResidentStudent jeremy = new Student(...);
```

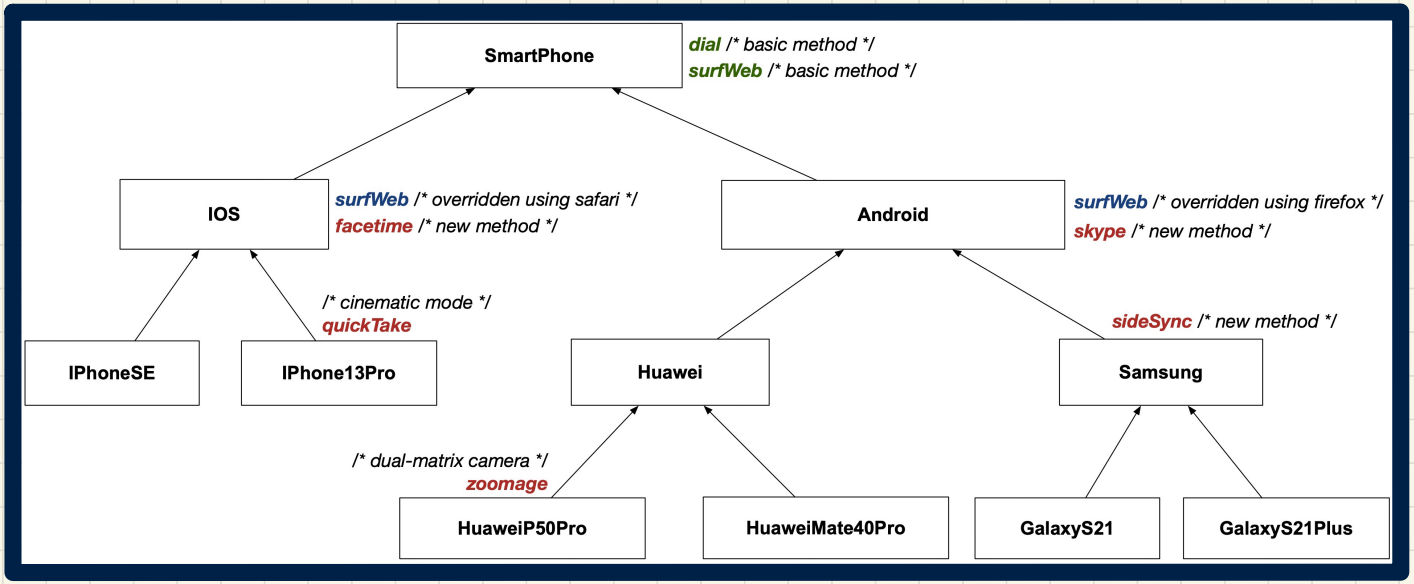
Change of Dynamic Type: Exercise (1)



Exercise 1:

```
Android myPhone = new HuaweiP50Pro(...);
myPhone = new GalaxyS21(...);
```

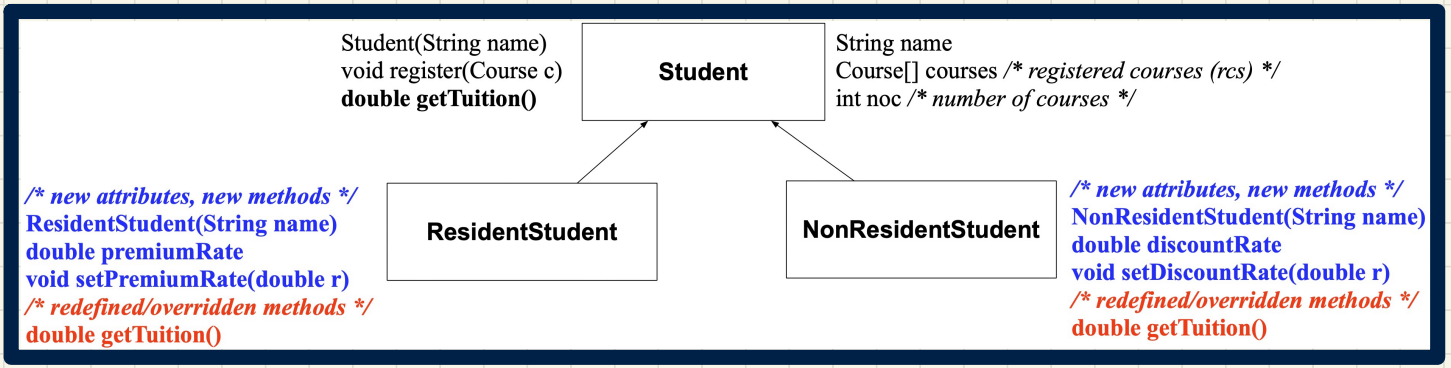
Change of Dynamic Type: Exercise (2)



Exercise 2:

```
IOS myPhone = new HuaweiP50Pro(...);
myPhone = new GalaxyS21(...);
```

Change of Dynamic Type (2.1)



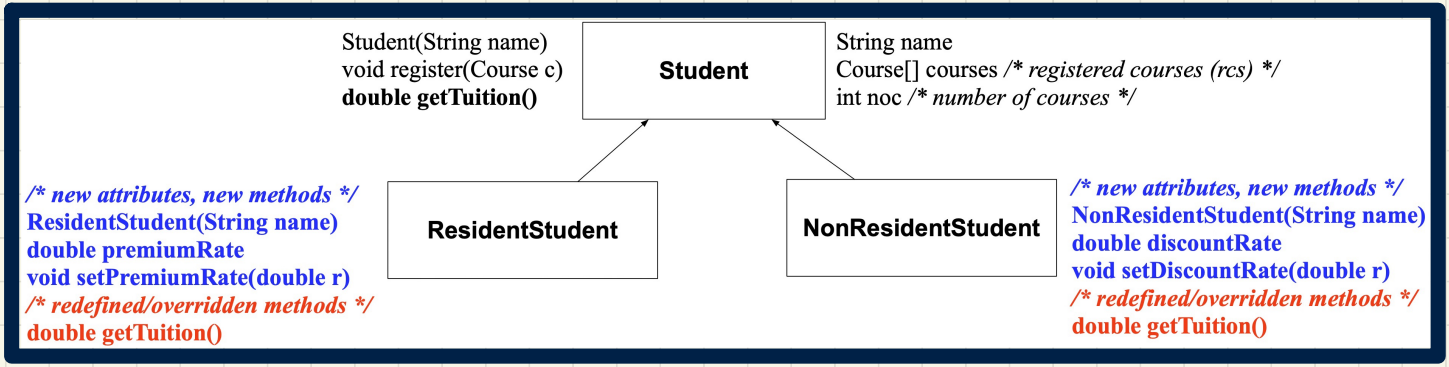
Given:

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

Example 1:

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

Change of Dynamic Type (2.2)



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());
```

Polymorphism and Dynamic Binding

Polymorphism:

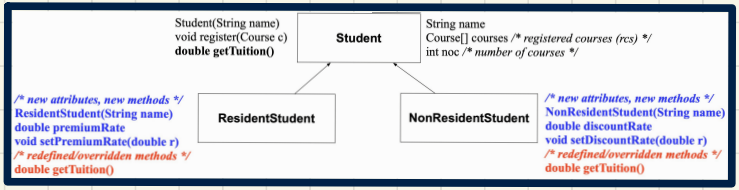
An object's **static type** may allow **multiple** possible **dynamic types**.

⇒ Each **dynamic type** has its **version** of method.

Dynamic Binding:

An object's **dynamic type** determines the **version** of method being invoked.

```
Student jim = new ResidentStudent(...);
jim.getTuition();
jim = new NonResidentStudent(...);
jim.getTuition();
```




```
SmartPhone sp1 = new IPhone13Pro(...);
SmartPhone sp2 = new GalaxyS21(...);
sp1.surfWeb();
sp1 = sp2;
sp1.surfWeb();
```