More on Is-A (1/2)

```
1  class CSIEProfessor{ String name; int office_num; }
2  class HTLin extends CSIEProfessor{
3    //HTLin is a CSIEProfessor
4    HTLin(){ name = "HTLin"; office_num = 314; }
5  }
6  class CLChen extends CSIEProfessor{
7    //CLChen is a CSIEProfessor
8    String title;
9    CLChen(){
10     name = "CLChen"; office_num = 500; title = "Vice␣Chair";
11   }
12 }
13 class YDLyuu extends CSIEProfessor{
14   //YDLyuu is a CSIEProfessor
15   String title;
16   YDLyuu(){
17     name = "YDLyuu"; office_num = 200; title = "Chair";
18   }
19 }
```

- over-use of **extend** for is-a: one class describes one instance
- usual goal of OOP: one class, **many** instances

More on Is-A (2/2)

```
1  class CSIEProfessor{
2    String name; String title;
3    int office_num;
4    CSIEProfessor(String name, String title, int office_num){ }
5    bool is_chair(){ return title.equals("Chair"); }
6    bool decide_budget(){ if (is_chair()){ /*...*/ } }
7  }
8
9  class CSIEProfessorDemo{
10   public static void main(String[] argv){
11     CSIEProfessor  = new CSIEProfessor("YDLyuu", "Chair",
         200);
12     YDLyuu.decide_budget();
13   }
14 }
```

- under-use of **extend** for is-a: overly complicated class CSIEProfessor
- potential for bugs/hacks
  what if `new CSIEProfessor("YAOMMENT", "Chair", 0)`?

More on Is-A: Key Point

```
class ViceChair extends Professor
```

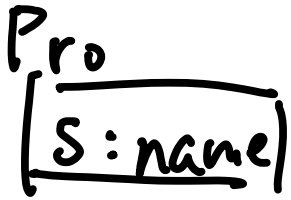
is a kind (type) of

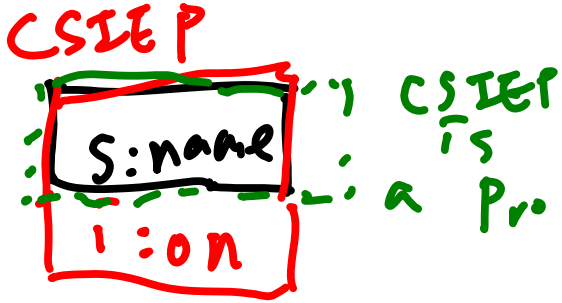
```
Professor HTLin = new Professor()
```

is an instance of

Instance Variables and Inheritance (1/3)

```
1  class Professor{
2    public String name;
3
4    public String get_name(){
5      return name;
6    }
7  }
8  class CSIEProfessor extends Professor{
9    public int office_number;
10   public int get_office_number(){
11     return office_number;
12   }
13 }
```





- `CSIEProfessor`: two instance variables;
  `Professor`: one instance variable

Instance Variables and Inheritance (2/3)

```
1  class Professor{
2    public String name;
3    public String get_name(){ return name;}
4  }
5  class CSIEProfessor extends Professor{
6    public String name; //?!
7    public int office_number;
8    public int get_office_number(){ return office_number;}
9    public String get_this_name(){ return name;}
10 }
11 /* CSIEProfessor HTLin = new CSIEProfessor();
12    Professor HTLin = new CSIEProfessor();
13    Professor HTLin = new Professor(); */
14 /* System.out.println(HTLin.get_name());
15    System.out.println(HTLin.get_this_name());
16    System.out.println(HTLin.name); */
```

- `CSIEProfessor`: three instance variables;
  `Professor`: one instance variable
- which name will we get?

Instance Variables and Inheritance (3/3)

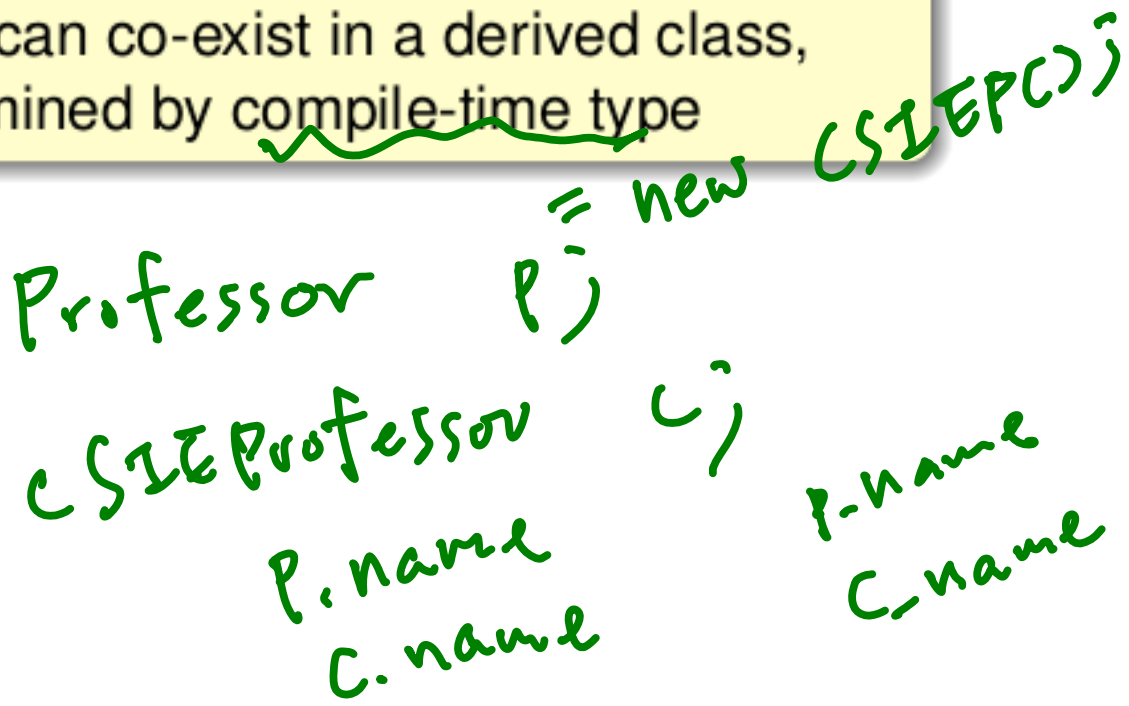
```
1  class Professor{
2    public String p_name;
3    public String get_name(){ return p_name; }
4  }
5  class CSIEProfessor extends Professor{
6    public String c_name;
7    public int office_number;
8    public int get_office_number(){ return office_number;}
9    public String get_this_name(){ return c_name;}
10 }
```

- an (almost) equivalent view of what the compiler sees

(Prof) HTLin.name

Instance Variables and Inheritance: Key Point

- instance variable binding: determined at compile time
- same "name" can co-exist in a derived class, binding determined by compile-time type



Instance Methods and Inheritance (1/2)

```
1  class Professor{
2    public void say_hello(){
3      System.out.println("Hello!");
4    }
5  }
6  class CSIEProfessor extends Professor{
7    public void play_BBS(){
8      System.out.println("Fun!");
9    }
10 }
```

- `CSIEProfessor`: two instance methods;
  `Professor`: one instance method

Instance Methods and Inheritance (2/2)

```
1  class Professor{
2    public void say_hello(){
3      System.out.println("Hello!");
4    }
5  }
6  class CSIEProfessor extends Professor{
7    public void say_hello(){
8      System.out.println("May␣the␣OOP␣course␣be␣with␣you!");
9    }
10 }
11 /* alternatives
12 CSIEProfessor HTLin = new CSIEProfessor();
13 Professor HTLin = new CSIEProfessor();
14 Professor HTLin = new Professor();
15 */
16 /* calls
17 HTLin.say_hello();
18 */
```

- which say_hello() will be called?

Instance Methods and Inheritance: Key Point

```
Professor p = new CSIEProfessor();
interact_with_professor(p);

void interact_with_professor(Professor p){
        p.say_hello();
}
```

instance method binding: dynamic, depending on run-time instance types

static binding

dynamic binding

"dynamic"