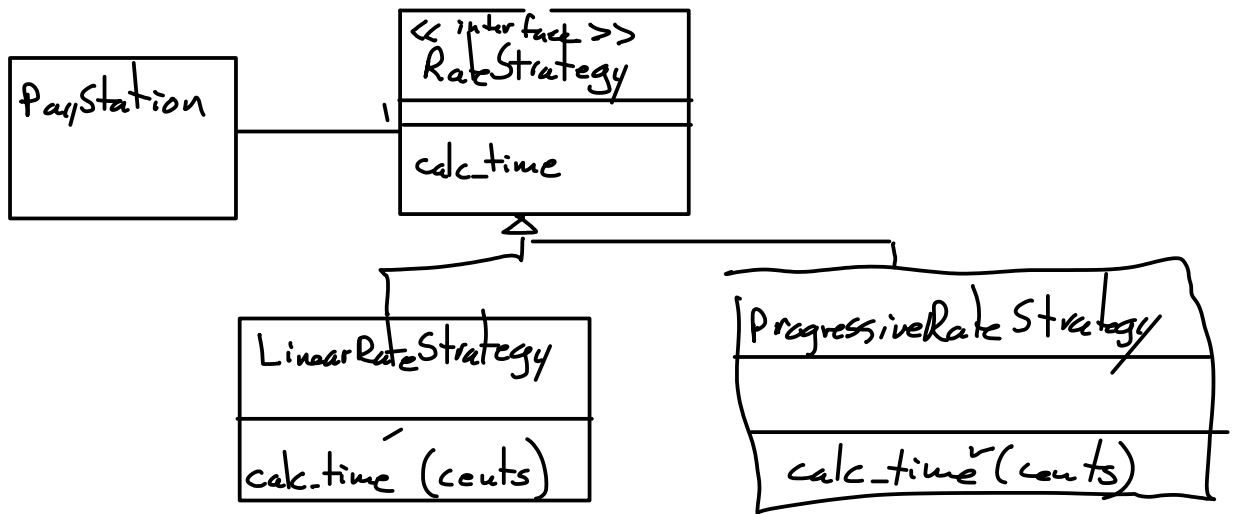


Recall: new requirements

Betatown - diff. rate structure (Progressive rate)

First hour    \$1.50    (.05 → 2 mins)  
second        \$2.00    (.05 → 1.5 mins)  
3 +            \$3.00    (.05 → 1 min)

## Compositional Design



Java: RateStrategy is an interface with multiple impl.

Association to PayStation means instance variable

```
class PayStation {  
    private RateStrategy rateStrategy;
```

type

variable name

← Declaring an instance var.

```
    public PayStation (RateStrategy rs) {  
        this.rateStrategy = rs;
```

constructor  
→ saves the  
RateStrategy into  
instance var.

```
    }  
    :  
}
```

client create correct paystation

PayStation alphaStation = new PayStation (new LinearRateStrategy());

Python: same design but...

- don't need interfaces
- don't need a class - just use a function.

---

Summarize Design idea

- \* identified point of variation
- \* assign responsibility to a new object with suitable interface
- \* delegate responsibility to that object

prog rate tests

~~1 hour~~ for 150

60 mins

120 mins for ~~2.00~~ 350

180 mins for 650

240 mins 950

within 1st hour partial

within 2nd hour partial

within 3rd hour

4+ partial