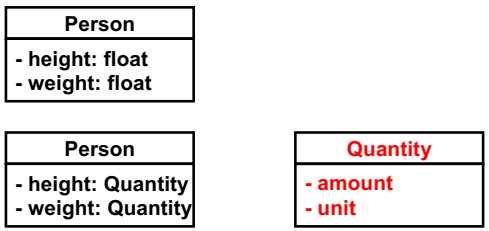
Quantity Patterns

Quantity Pattern

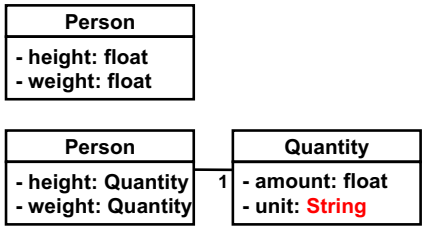
- Intent
  - Represents a value with its amount and unit.
    - Example values: 6 feet, 1.8 meters, 32 F, 0 C, etc.
    - Explicitly focuses on units.
  - Allows a value to be converted from one unit to another unambiguously.

Class Quantity

- Represents a value with its amount and unit.
  - Example values: 6 feet, 1.8 meters, 32 F, 0 C, etc.
  - Explicitly focuses on units.



- How to type unit?
- Let's not type it with String.



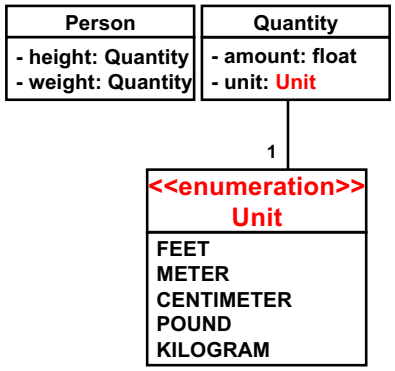

```
if( unit.equals("feet") ){
     // do something
}else if ( f.equals("meter") ){
     // do something else
}
```

```
new Quantity(6, "fet");
```

You need to write error-handling code carefully. In the worst case, errors may not be detected at runtime.

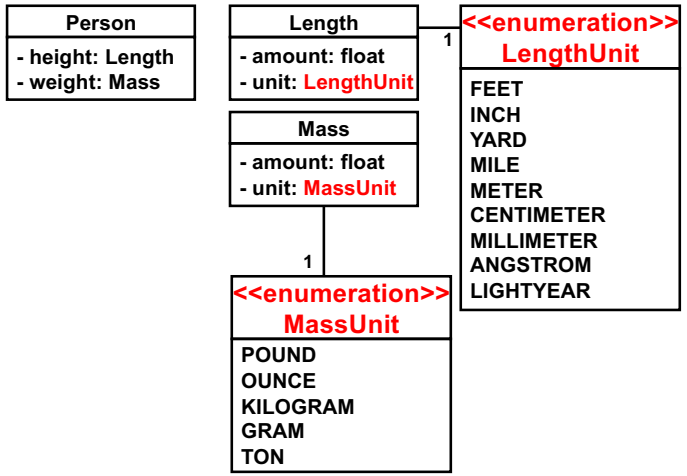
You want to catch as many errors as possible at compile-time. Have your complier work harder!

Use an Enumeration

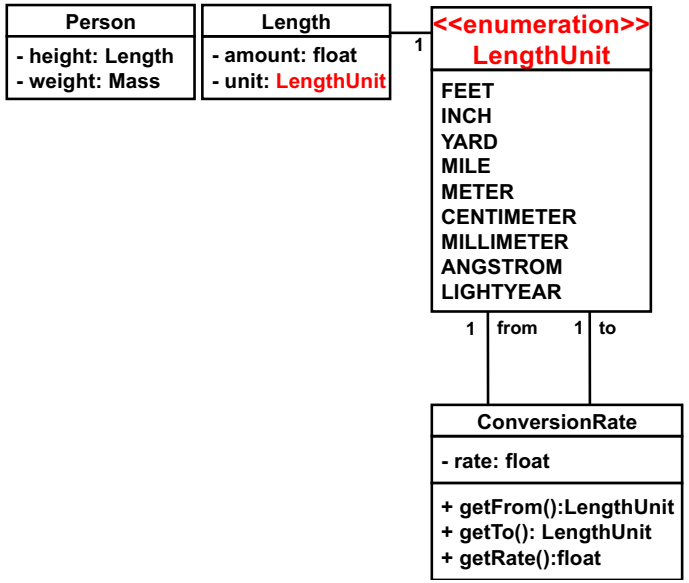



```
• if( unit.equals("feet") ){
      // do something
  }else if ( unit.equals("meter") ){
      // do something else
  }

• if( unit.equals(Unit.FEET) ){
  // do something
  }
  else if( unit.equals(Unit.METER) ){
  // do something else
  }

• new Quantity(6, "fet");
```

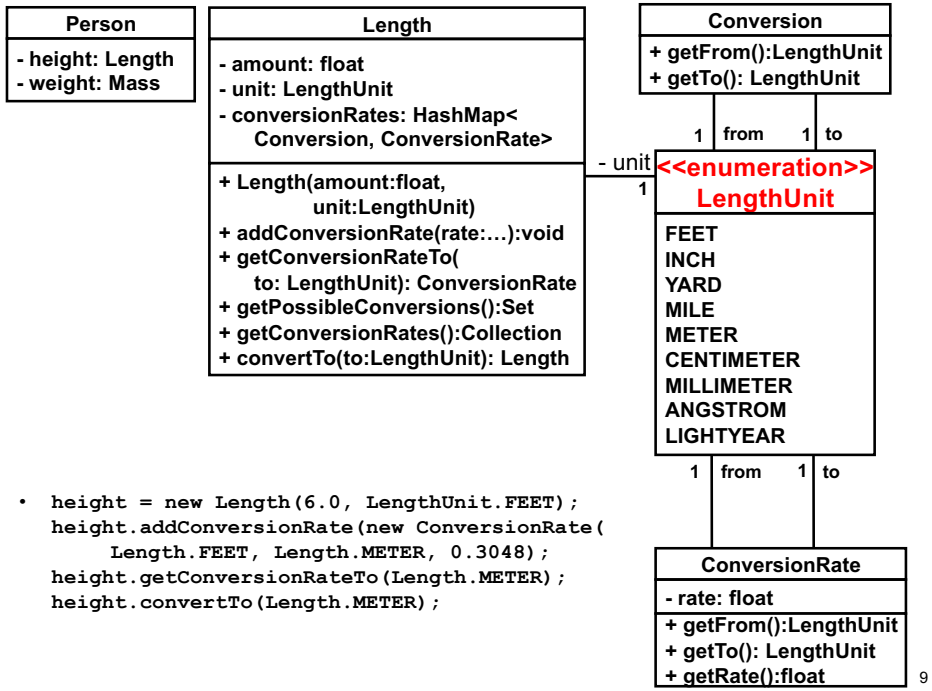

- You might want to categorize units as the number of units grows.



Unit Conversion

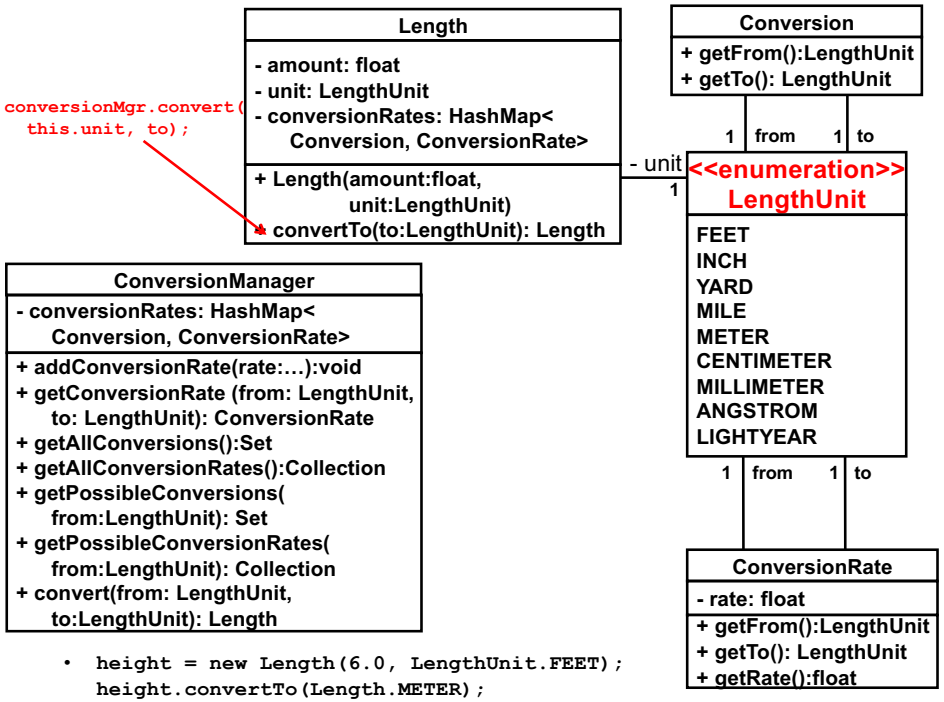


- Each instance of ConversionRate takes care of a pair of two particular units.
  - An instance: Conversion from LengthUnit.FEET to LengthUnit.METER.
  - Another instance: Conversion from LengthUnit.METER to LengthUnit.FEET.
- Who manages these ConversionRate instances?



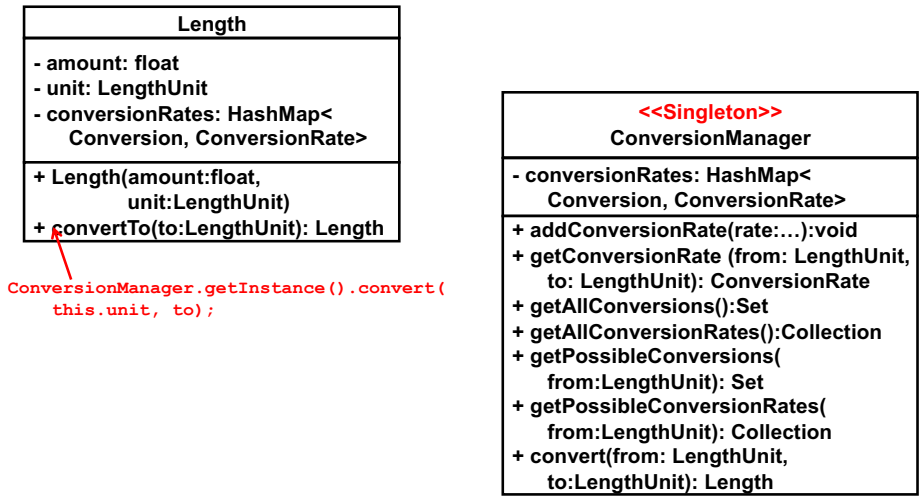

```
height = new Length(6.0, LengthUnit.FEET);
height.addConversionRate(new ConversionRate(
     Length.FEET, Length.METER, 0.3048);
height.getConversionRateTo(Length.METER);
height.convertTo(Length.METER);
```


- Conversion-related methods scatter in all quantity classes such as Length, Mass, etc.
  - addConversionRate(), getConversionRateTo(), getPossibleConversions(), etc. etc.
- You can consolidate them to a single class, say ConversionManager.

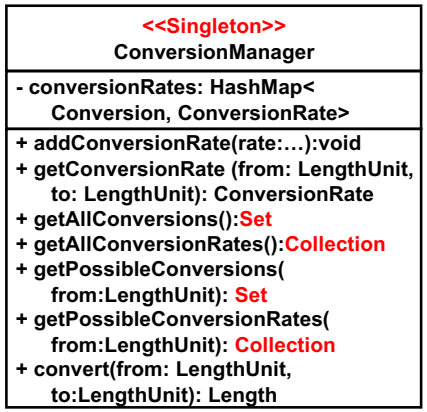



```
height = new Length(6.0, LengthUnit.FEET);
height.convertTo(Length.METER);
```

Is ConversionManager a Singleton?

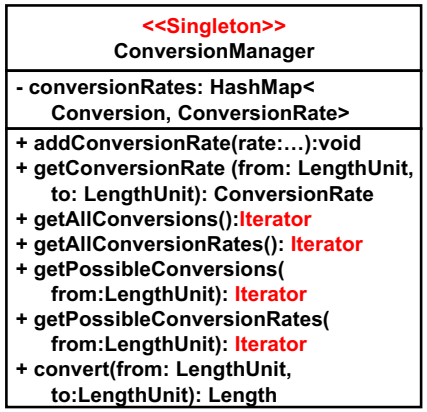


What Types to return in ConversionManager's Methods?



- Users of ConversionManager do not (do not want to) really care the types to represent keys and values.
- They would simply traverse keys and values with iterators.

```
Set<Conversion> conversions =
    ConversionManager.getInstance().getAllConversions();
Iterator<Conversion> iterator = conversion.iterator();
while ( iterator.hasNext() ) {
     Conversion c = iterator.next();
     ...;}
```

- Return an Iterator rather than a Set or Collection.
  - If you are an API designer for ConversionManager, have your API users program to an interface (Iterator), not implementations (Set and Colleciton)

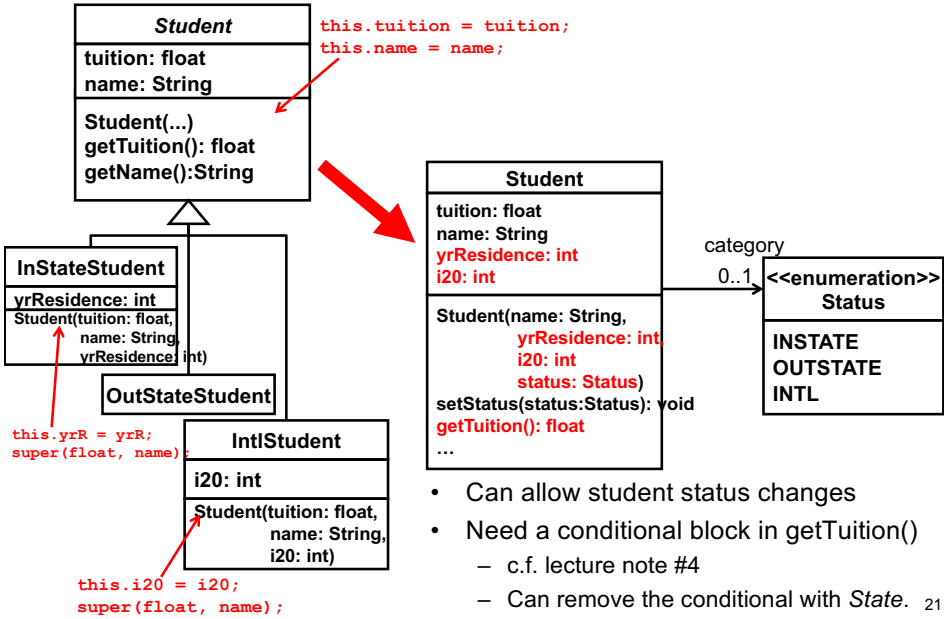
```
Iterator<Conversion> iterator =
    ConversionManager.getInstance().getAllConversions();
while ( iterator.hasNext() ) {
     Conversion c = iterator.next();
     ...;}
```


State Design Pattern

State Design Pattern

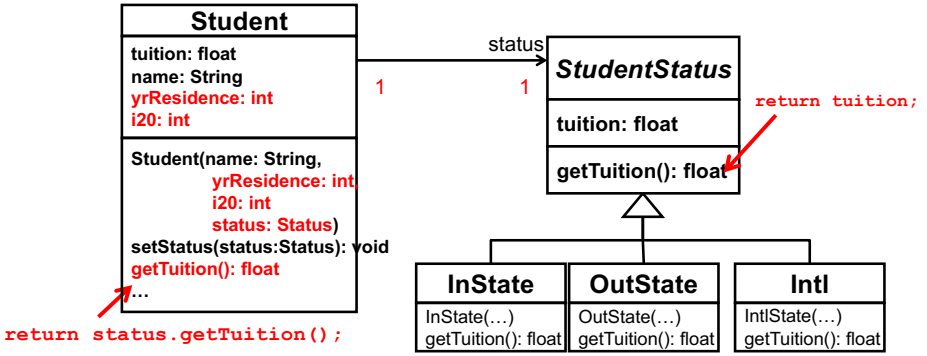
- Intent
  - Allow an object to change its behavior according to its state.

Eliminating Class Inheritance



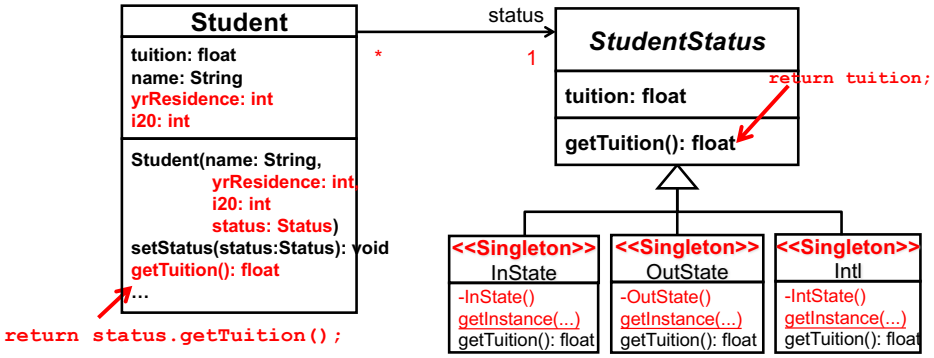
- Can allow student status changes
- Need a conditional block in getTuition()
  - c.f. lecture note #4
  - Can remove the conditional with *State*.

Using *State*



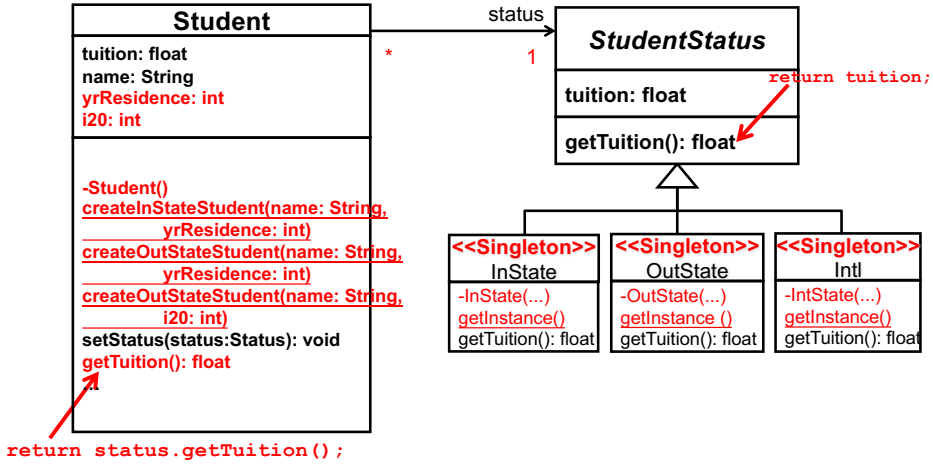

```
Student s1 = new Student( ..., new InState(...) );
s1.getTuition();
```


State Classes as Singleton




```
Student s1 = new Student( ..., InState.getInstance(...) );
s1.getTuition();
```


Adding Static Factory Methods

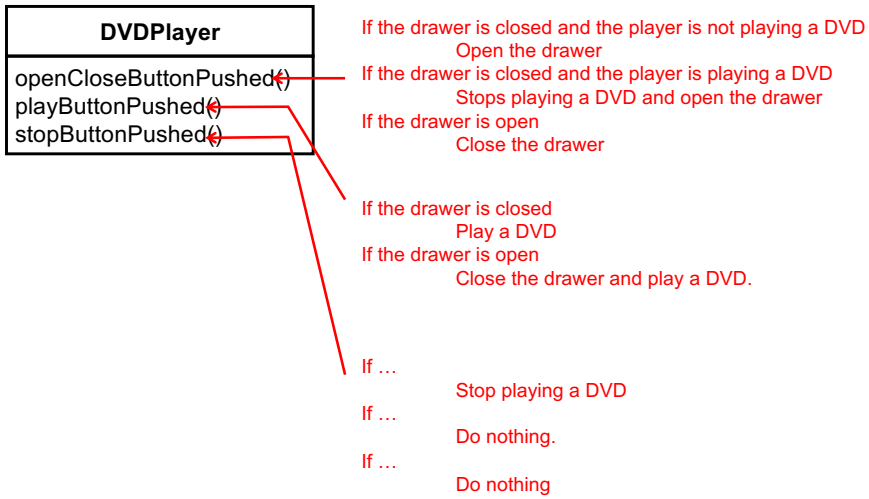



```
Student s1 = Student.createInStateStudent( "John Smith", 18 );
s1.getTuition();
```

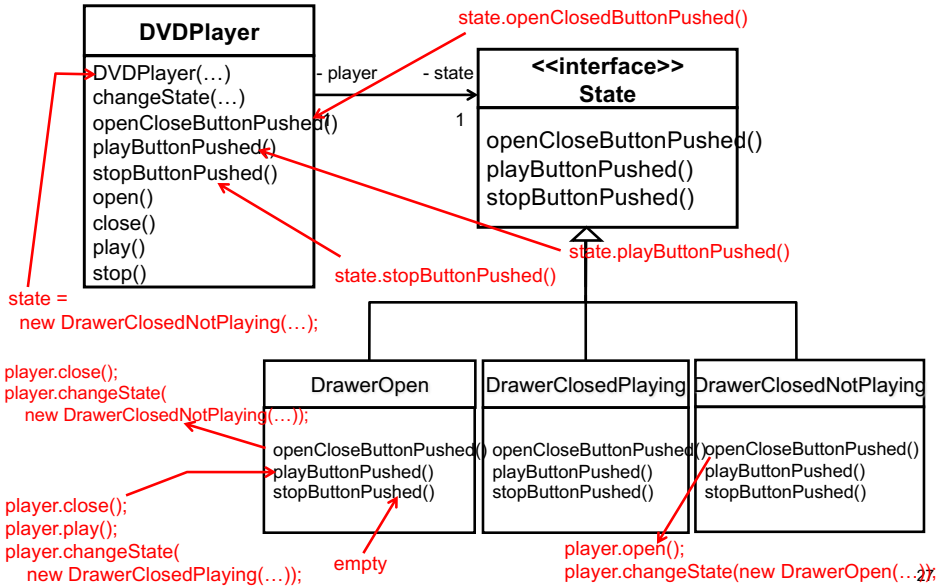
Another Example: DVD Player

- When the “open/close” button pushed,
  - Opens the drawer
    - If the drawer is closed and the player is not playing a DVD.
  - Stops playing a DVD and opens the drawer
    - if the drawer is closed and the player is playing a DVD.
  - Closes the drawer
    - if the drawer is open.
- When the “play” button pushed,
  - Plays a DVD
    - If the drawer is closed.
      - Displays an error message if the drawer is empty.
  - Closes the drawer and plays a DVD
    - If the drawer is open.
      - Displays an error message if the drawer is empty.

- When the “stop” button pushed
  - Stops playing a DVD
    - If the drawer is closed and the player is playing a DVD
  - Does nothing.
    - If the drawer is closed and the player is not playing a DVD.
  - Does nothing
    - If the drawer is open.



Defining States as Classes



State Classes as Singleton