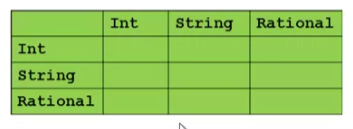
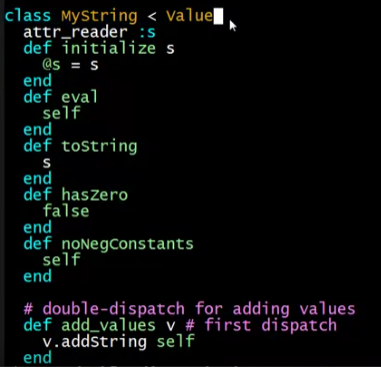
**Example –** Addition using OOP

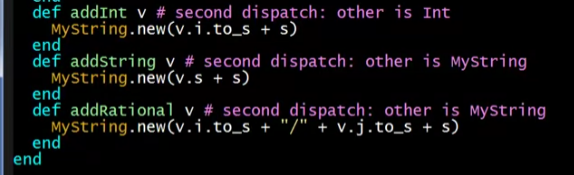
* Include variants String and Rational
* (Re)define Add to work on ANY pair of Int, String, Rational
  + Concatenation if either argument a String, else math

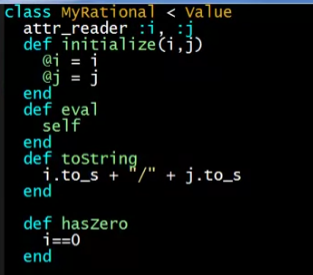
Now just defining the addition operation is a different 2D grid:

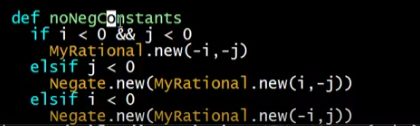


Add String and Rational class first

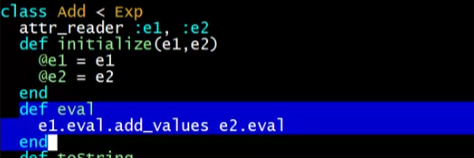








Add function

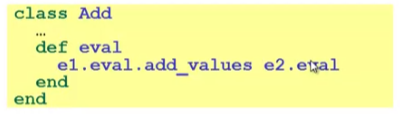


* e1.eval should have add\_values method

**What about OOP?**

Start promising:

* Use OOP to call method *add\_values* to one value with other value as result



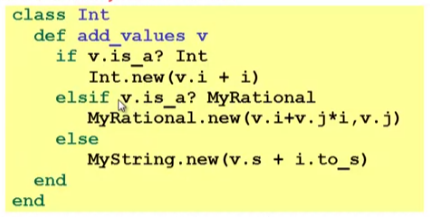
Classes *Int*, *MyString*, *MyRational* then all implement

* Each handling 3 of the 9 cases: “add self to argument”



**First Try (Hybrid)**

* This approach is common, but is “not as OOP”
  + So do not do it on your homework



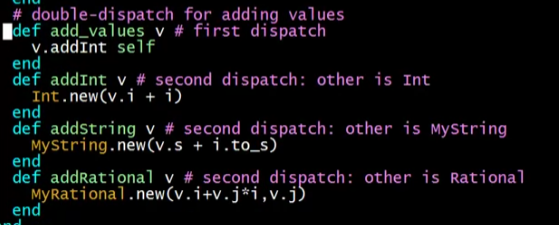
* A “hybrid” style where we used dynamic dispatch on 1 argument and then switched to Racket-style type tests for other argument
  + Definitely not “full OOP”

**Another way…**

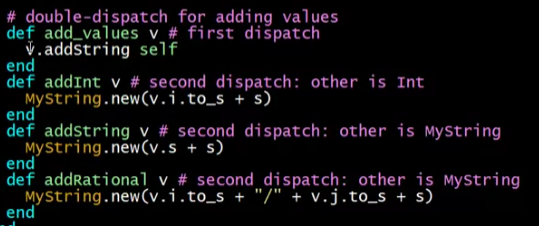
* *add\_values* method in *Int* needs “what kind of thing” *v* has
  + Same problem in *MyRational* and *MyString*
* In OOP, “always” solve this by calling a method on v instead!
* But now we need to “tell” v “what kind of thing” self is
  + We know what!
  + “Tell” v by calling different methods on v, passing self
* Use a “programming trick” (?) called ***double-dispatch***

9 different methods (3 per class) equivalent from 9 cases in ML

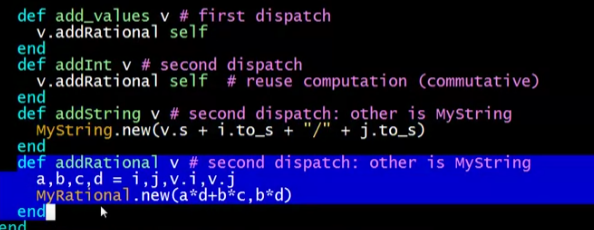
In class *Int*



In class *MyString*



In class MyRational



**Double-dispatch “trick”**

* Int, MyString, and MyRational each define all of addInt, addString, and addRational
  + For example, *String’s addInt* is for adding concatenating an integer argument to the string in *self*
  + 9 total methods, one for each case of addition
* Add’s *eval* method calls *e1.eval.add\_values e2.eval*, which dispatches to *add\_values* in *Int, String, or Rational*
  + Int’s add\_values: v.addInt self
  + MyString’s add\_values: v.addString self
  + MyRational’s add\_values: v.addRational self

So add\_values performs “2nd dispatch” to the correct case of 9!

**Why showing you this**

* Honestly, partly to belittle full commitment to OOP
* To understand dynamic dispatch via a sophisticated idiom
* Because required for homework
* To contrast with *multimethods* (optional)

Optional Note: Double-dispatch in Java

