

Section 10 **Mixins & Subclasses**

Spring 2020

Dispatch Overview

Dispatch is the *runtime* procedure for looking up which function to call based on the parameters given:

- Ruby (and Java) use *Single Dispatch* on the implicit **self** (or "this") parameter
 - Uses runtime class of **self** to lookup the method when a call is made
 - · This is what you learned in CSE 143
- Double Dispatch uses the runtime classes of both self and a single method parameter
 - · Ruby/Java do not have this, but we can emulate it
 - This is what you will do in HW7

2

4

 $\boldsymbol{\cdot}$ You can dispatch on any number of the parameters and the general term $% \boldsymbol{\cdot}$ for this is *Multiple Dispatch* or *Multimethods*

Emulating Double Dispatch

1

Player

- To emulate double dispatch in Ruby (on HW7) just use the built-in single dispatch procedure twice!
 - · Have the principal method immediately call another method on its first parameter, passing self as an argument
 - The second call will implicitly know the class of the self parameter
 - It will also know the class of the *first parameter* of the principal method, because of Single Dispatch
- ·There are other ways to emulate double dispatch
 - · Found as an idiom in SML by using case expressions

Double Dispatch Example: RPS

- Suppose we wanted to code up a game of "Rock-Paper-Scissors":
 - o A game that is played in rounds with 2 players.
 - o Each player gets to pick a weapon: one of "Rock", "Paper", or
- Each combination results in a winner/loser (except when both are the same):
 - o Rock beats Scissors
 - Paper beats Rock
 - o Scissors beats Paper

3

Double Dispatch Example: RPS

- What are the different combinations of games?
 - o Player 1 fights Player 2 with a tool, and Player 2 responds, which determines the outcome.

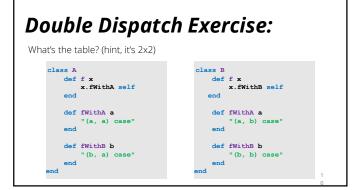
	T tayer 1		
	Rock	Paper	Scissors
Rock	Tie	Paper wins	Rock wins
Paper	Paper wins	Tie	Scissor wins
Scissors	Rock wins	Scissor wins	Tie

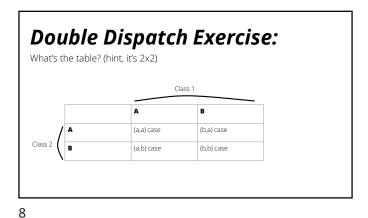
Double Dispatch Example: RPS

- How could we represent this in an OOP way?
 - How does "Class 1" fight "Class 2"? How do we encode the "tool"? How do we encode the "outcome"?

		Rock	Paper	Scissors
Class 2	Rock	Tie	Paper wins	Rock wins
	Paper	Paper wins	Tie	Scissor wins
,	Scissors	Rock wins	Scissor wins	Tie

5 6





Extending RPS

7

What if we wanted to extend our game to add an action to convert each of the tools to strings?

• What would we have to change so that we could still play this game, but with another action?

	Rock	Paper	Scissors
Rock	Tie	Paper wins	Rock wins
Paper	Paper wins	Tie	Scissor wins
Scissors	Rock wins	Scissor wins	Tie
toString*	Rock	Paper	Scissors

* note: not a Class, but a method, because it only operates on 1 class, not 2.

Mixins

- Collection of methods
 - o Unlike class, you cannot instantiate it
- Can include any number of mixins
- Provides powerful extensions to the class with little

9 10

Mixins

- It's just "copy and paste the code into the class"
 - o Will override existing code
 - o Have access to instance functions
 - o Have access to instance variables

Mixins Example

```
odule
Doubler def
double
self + self # assume included in classes w/
 class String
    Doubler
end anotherPt
attr accessor:x,
:y include Doubler
def + other
ans = AnotherPt.new
ans.x = self.x +
other.x ans.y = self.y
+ other.y ans
```

12 11

2

Method Lookup Rules

- 1. Current class
- 2. Current class's mixins
 - a. Latest included mixin
 - b.
 - c. Earliest included mixin
- 3. Current class's super class
- 4. Current class's super class's mixins

Comparable

It provides you methods to compute

<, >, ==, !=, >=, <=

What's needed?

- Define function <=> (spaceship operator)
 - · Return negative, 0 or positive number

Very similar to Java Comparable interface which requires compareTo

13

14

Enumerable

It provides you methods to iterate over the object -> supports map, find!

What's needed?

- Define function each
 - each will either call each of other object or will yield result

Very similar to Java Iterable interface

Java Subtyping

Arrays should work just like records in terms of depth subtyping

• But in Java, if t1 <: t2, then t1[] <: t2[]

• So this code type-checks, surprisingly

```
class Point { ... }
class ColorPoint extends Point { ... }
void m1(Point[] pt_arr) {
  pt_arr[0] = new Point(3,4);
      ring m2(int x) {
ColorPoint[] cpt_arr = new ColorPoint[x];
for (int i=0; i < x; i++)
    cpt_arr[i] = new ColorPoint(0,0,"green");
m1(cpt_arr; // !
return cpt_arr[0].color; // !</pre>
```

15

17

16

Why?

More flexible type system allows more programs but prevents fewer errors

• Seemed especially important before Java/C# had generics

Good news: despite this "inappropriate" depth subtyping

- e.color will never fail due to there being no color field
- Array reads e1[e2] always return a (subtype of) t if e1 is a t[]

Bad news: to get the good news

- e1[e2]=e3 can fail even if e1 has type t[] and e3 has type t
- Array stores check the run-time class of e1's elements and do not allow storing a supertype
- No type-system help to avoid such bugs / performance cost

wat

18

https://www.destroyallsoftware.com/talks/wat

Thank you for a great quarter!

Take care of yourself and eachother ∜

19