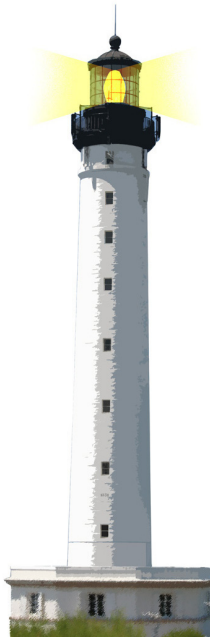
Conditionals to dispatch

A simple case

S. Ducasse



Pharo

Goals

- Think about messages
- Eliminate condition for message
- Understand double dispatch will make you understand message passing!
- Another example of **Don’t Ask, Tell**

About train prices

```
testPriceOfPersonInFirstClass
    self
      assert: (FirstClassTrainTicket new priceFor: Person new)
      equals: 100
```

FirstClassTrainTicket

```
FirstClassTrainTicket >> priceFor: aPerson
    ^ 100
```

Adding kids

```
testPriceOfChildInFirstClass
    self
      assert: (TrainTicketFirstClass new priceFor: Child new)
      equals: 45
```

```
FirstClassTrainTicket >> priceFor: aPerson

    | pClass |
    pClass := aPerson class.
    ^ pClass = Child
        ifTrue: [ 45 ]
        ifFalse: [ 100 ]
```

Introducing Granny prices

```
testPriceOfGranmaInFirstClass
    self
      assert: (FirstClassTrainTicket new priceFor: Granny new)
      equals: 55
```

FirstClassTrainTicket

```
priceFor: aPerson

    | pClass |
    pClass := aPerson class.
    ^ pClass = Granny
        ifTrue: [ 55 ]
        ifFalse: [
            pClass = Child
              ifTrue: [ 45 ]
              ifFalse: [ 100 ] ]
```

Entering second class tickets

```
testPriceOfChildSecond
    self assert: (SecondClassTrainTicket new priceFor: Child new) equals: 35
```

```
testPriceOfGranmaSecond
    self assert: (SecondClassTrainTicket new priceFor: Granny new) equals: 45
```

Entering second class tickets

```
SecondClassTrainTicket >> priceFor: aPerson

    | pClass |
    pClass := aPerson class.
    ^ pClass = Granny
        ifTrue: [ 45 ]
        ifFalse: [
          pClass = Child
            ifTrue: [ 35 ]
            ifFalse: [ 80 ] ]
```

Well....

And

- Dog
- Bicycle
- Couple
- Sky, windsurf

And third class...

We can do better

Propose a solution without any test!

Let us get started

```
TrainTicketFirstClass >> priceFor: aPerson

    ^ ...
```

```
Person >> priceForFirstClass
    ^ 100
```

```
Child >> priceForFirstClass
    ^ 45
```

Let us get started

```
FirstClassTrainTicket >> priceFor: aPerson
    ^ aPerson priceForFirstClass
```

```
Person >> priceForFirstClass
    ^ 100
```

```
Child >> priceForFirstClass
    ^ 45
```

```
Granny >> priceForFirstClass
    ^ 55
```

Let us get started

```
SecondClassTrainTicket >> priceFor: aPerson
    ^ aPerson priceForSecondClass
```

```
Person >> priceForSecondClass
    ^ 100
```

```
Child >> priceForSecond
    ^ 35
```

```
Child >> priceForSecond
    ^ 45
```

Adding Bicycle... easy

```
Bicycle >> priceForFirstClass
    ^ 70
```

```
Bicycle >>  priceForSecond
    ^ 40
```

Adding WindSurf... easy

```
WindSurf >> priceForFirstClass
    ^ 65
```

```
WindSurf >>  priceForSecond
    ^ 40
```

Analysis

- No need to get complex conditionals more complex
- Modular: no need to touch existing code!
- No questions! Just orders
- Don’t ask, tell

Conclusion

Messages support

- Modular
- Extensible design

A course by

S. Ducasse, G. Polito, and Pablo Tesone