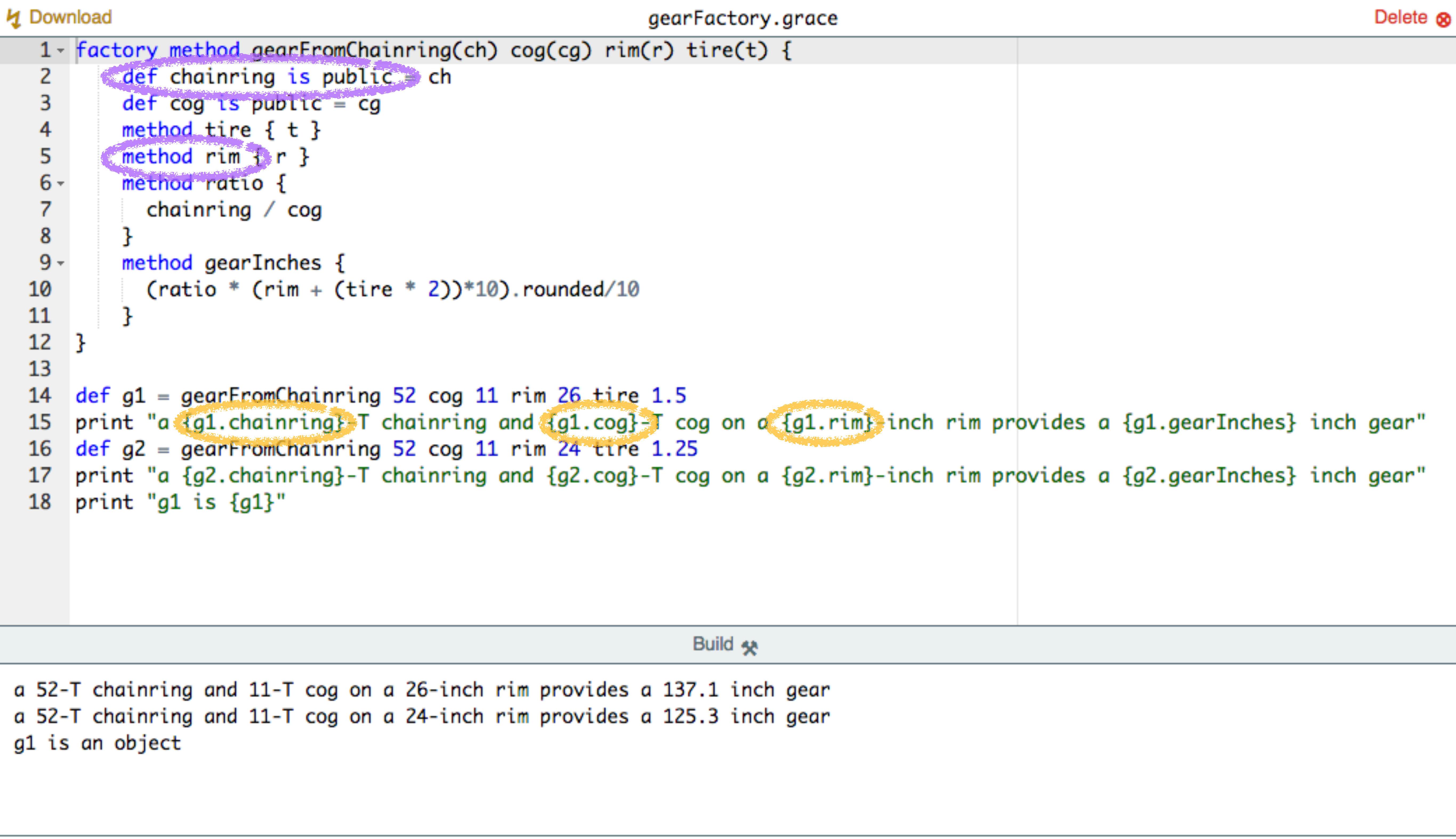
Grace version



```
factory method gearFromChainring(ch) cog(cg) rim(r) tire(t) {
    def chainring is public = ch
    def cog is public = cg
    method tire { t }
    method rim { r }
    method ratio {
        chainring / cog
    }
    method gearInches {
        (ratio * (rim + (tire * 2))*10).rounded/10
    }
}

def g1 = gearFromChainring 52 cog 11 rim 26 tire 1.5
print "a {g1.chainring}-T chainring and {g1.cog}-T cog on a {g1.rim}-inch rim provides a {g1.gearInches} inch gear"
def g2 = gearFromChainring 52 cog 11 rim 24 tire 1.25
print "a {g2.chainring}-T chainring and {g2.cog}-T cog on a {g2.rim}-inch rim provides a {g2.gearInches} inch gear"
print "g1 is {g1}"
```

```
a 52-T chainring and 11-T cog on a 26-inch rim provides a 137.1 inch gear
a 52-T chainring and 11-T cog on a 24-inch rim provides a 125.3 inch gear
g1 is an object
```

Single Responsibility?

- Not really!
  - since when does a gear have a a *tire* and a *rim*?
  - mixed up with other bits of *bicycle*
- Does it matter?
  - *Maybe!*

Arguments to “Leave it be”

- Code is *Transparent* and *Reasonable*
  - Consequence of a change should be *Transparent*
  - Cost of change *proportional* to benefits
- Why? Because there *are no dependents*
- *How* should we improve it?
  - We don't yet know — but the moment that we *acquire some dependents*, we will

➡ Wait until that time

Portland State UNIVERSITY

Arguments for Change

- Code is neither (re)usable, nor exemplary
  - multiple responsibilities ⇒ can't be used for other *gear*s
  - not a pattern to be emulated

Portland State UNIVERSITY

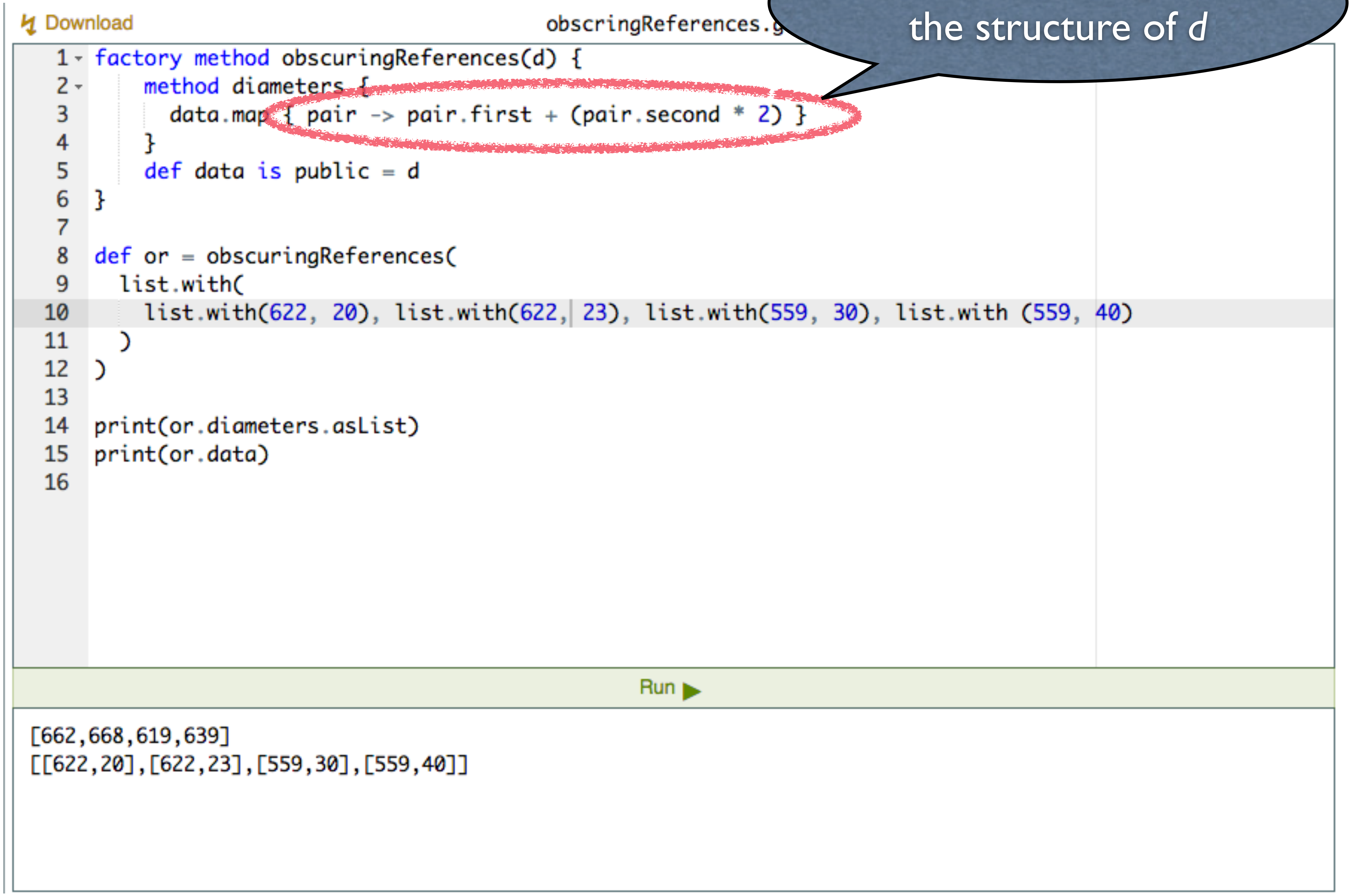
Improve it now *vs.* improve it later

- This tension always exists!
  - designs are never perfect
  - the designer has to weigh the costs and benefits of a change

Portland State UNIVERSITY

Embracing Change

- Some basic rules that will help, regardless of what change happens:
- Depend on Behaviour, not data
  - encapsulate instance variables
    - Grace gives us this one for free
  - encapsulate data
    - e.g., don't expose an array of pairs of numbers



```
factory method obscuringReferences(d) {
    method diameters {
      data.map { pair -> pair.first + (pair.second * 2) }
    }
    def data is public = d
}

def or = obscuringReferences(
  list.with(
    list.with(622, 20), list.with(622, 23), list.with(559, 30), list.with (559, 40)
  )
)

print(or.diameters.asList)
print(or.data)
```

```
[662,668,619,639]
[[622,20],[622,23],[559,30],[559,40]]
```

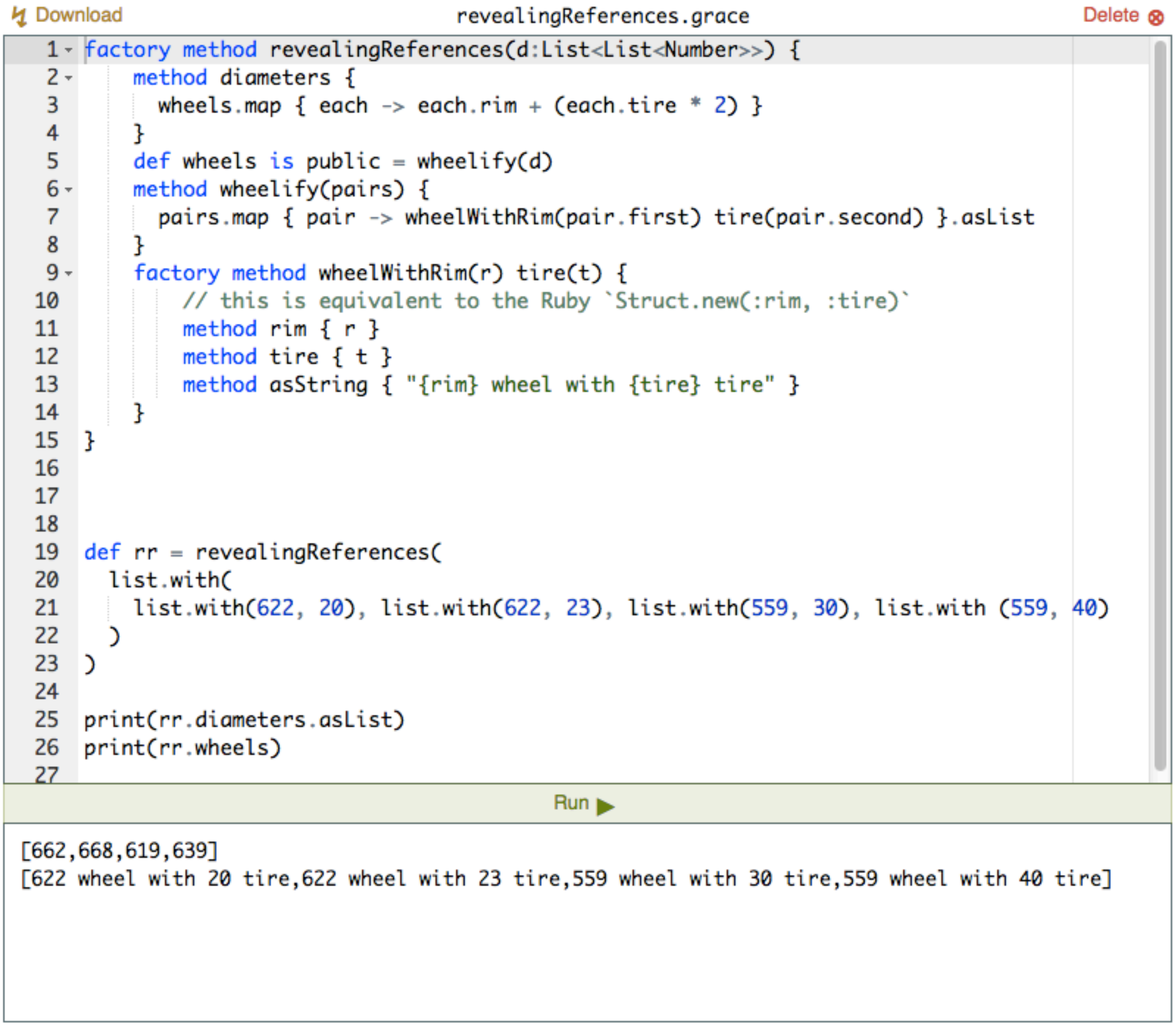
method knows all about the structure of *d*

Separate Structure from Meaning

- If you need a table of wheel and tire sizes, make it contain *objects*, not lists
- Metz uses a Ruby Struct to create a transparent object.
- In Grace:

```
factory method wheelWithRim(r) tire(t) {
    // this is equivalent to the Ruby `Struct.new(:rim, :tire)`
    method rim { r }
    method tire { t }
    method asString { "{rim} wheel with {tire} tire" }
}
```

Portland State UNIVERSITY



Download revealingReferences.grace Delete

```
factory method revealingReferences(d:List<List<Number>>) {
    method diameters {
      wheels.map { each -> each.rim + (each.tire * 2) }
    }
    def wheels is public = wheelify(d)
    method wheelify(pairs) {
      pairs.map { pair -> wheelWithRim(pair.first) tire(pair.second) }.asList
    }
    factory method wheelWithRim(r) tire(t) {
        // this is equivalent to the Ruby `Struct.new(:rim, :tire)`
        method rim { r }
        method tire { t }
        method asString { "{rim} wheel with {tire} tire" }
    }
}



def rr = revealingReferences(
  list.with(
    list.with(622, 20), list.with(622, 23), list.with(559, 30), list.with (559, 40)
  )
)

print(rr.diameters.asList)
print(rr.wheels)

```

Run ▶

```
[662,668,619,639]
[622 wheel with 20 tire,622 wheel with 23 tire,559 wheel with 30 tire,559 wheel with 40 tire]
```

Embracing Change

- Enforce Single Responsibility Everywhere
  - Extract extra responsibilities from methods
  - Isolate responsibilities in classes
    - Grace lets you create “local” factory methods

Portland State UNIVERSITY

The Real Wheel

- The customer tells you that she has need for computing wheel circumference.
- This tells you that your “bicycle calculator app” needs to model wheels.
- So let’s move *wheel* out of *gear*.