

# Advancing to a Domain-specific Language

---



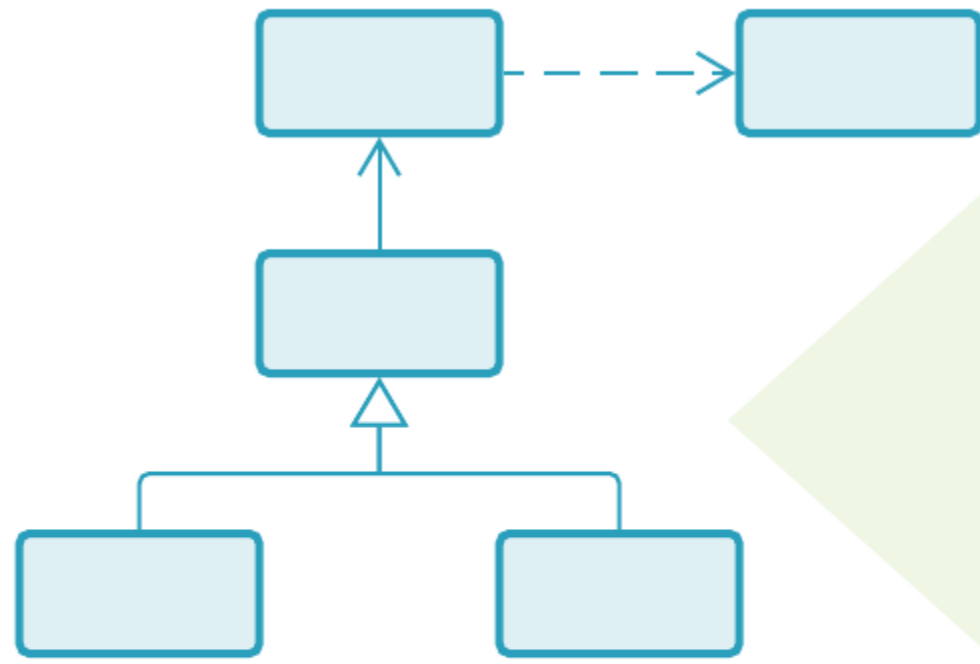
**Zoran Horvat**

CEO AT CODING HELMET

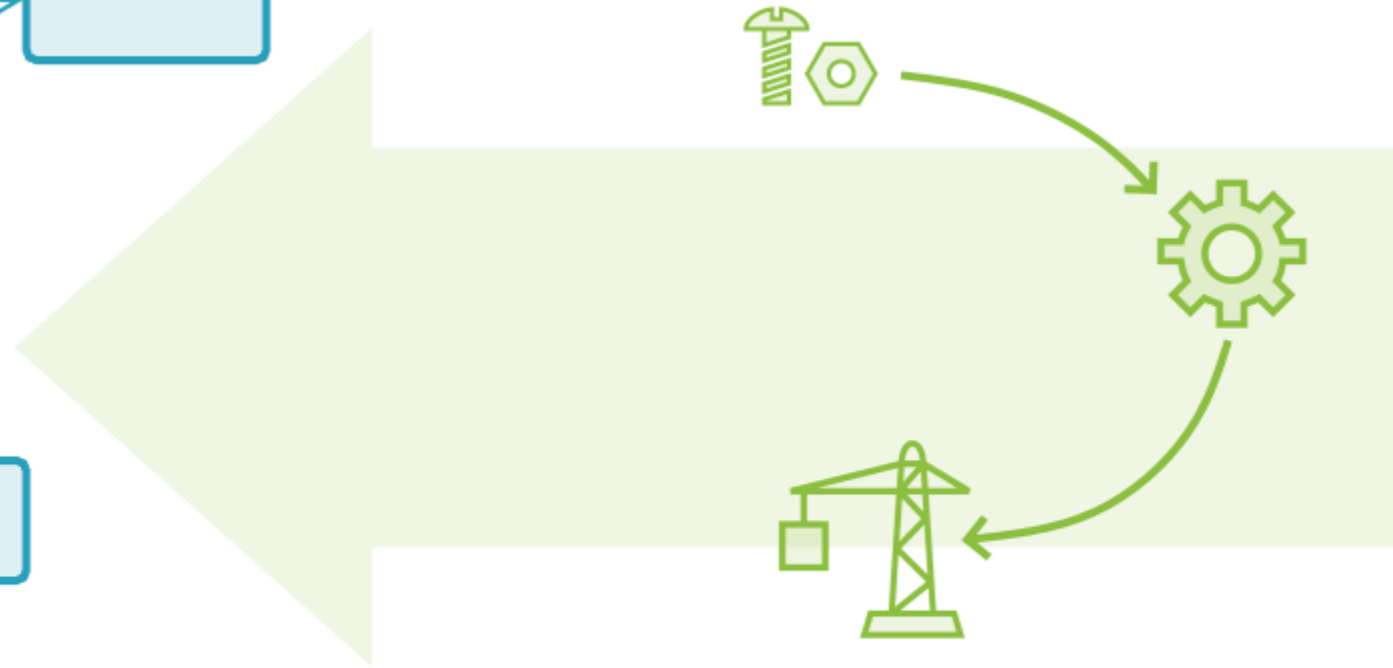
@zoranh75 <http://codinghelmet.com>



# Unifying the Languages

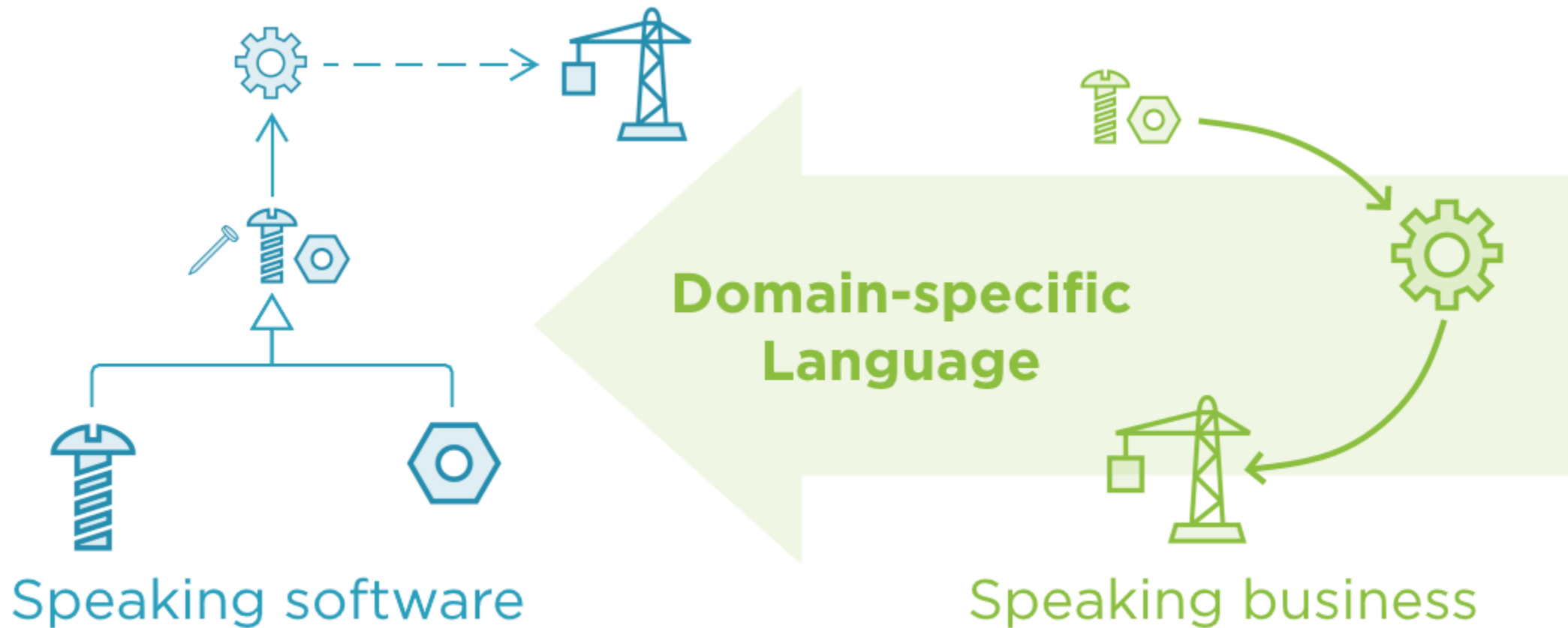


Speaking software

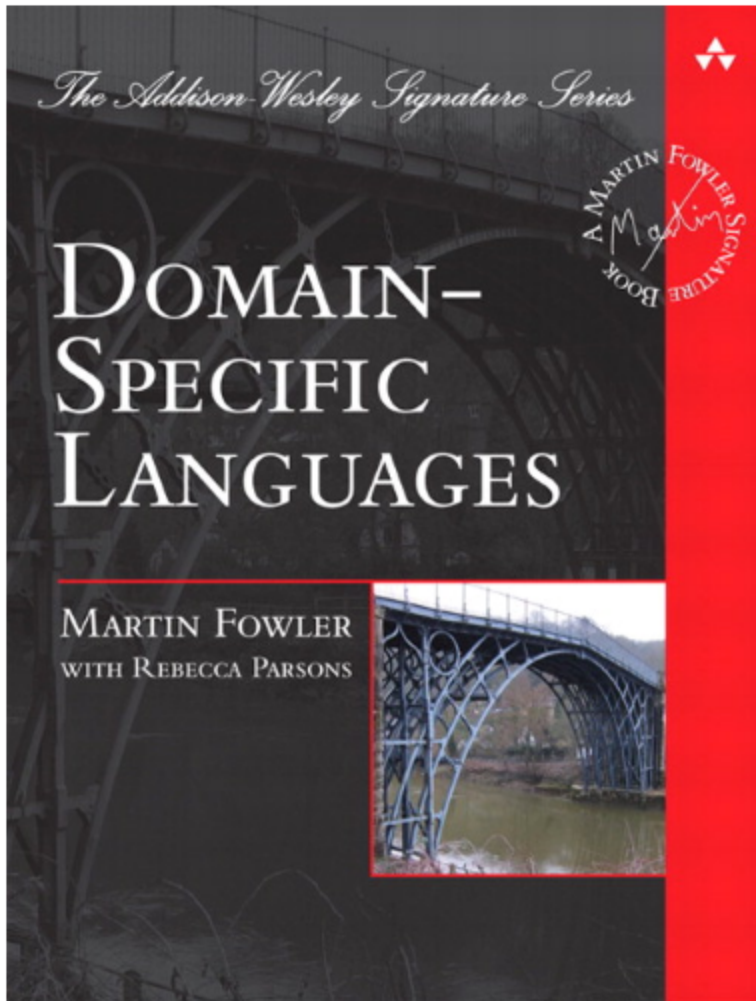


Speaking business

# Unifying the Languages



# Introducing Domain-specific Languages



Martin Fowler

## **Domain-specific Languages**

(September 2010)



# Introducing Domain-specific Languages

**Compose** elements of code the same way  
we compose sentences in a spoken language

```
painters  
    .thoseAvailable()  
    .findFastestOne()  
    .paint(area)
```

# Introducing Domain-specific Languages

**Compose** elements of code the same way  
we compose sentences in a spoken language

```
painters  
  .available()  
  .fastest()  
  .paint(area)  
  .orElse(report::noWorkDone)
```

|  
Internal DSL

# Introducing Domain-specific Languages

**Compose** elements of code the same way  
we compose sentences in a spoken language

```
painters  
  .available()  
  .fastest()  
  .paint(area)  
  .orElse(report::noWorkDone)
```

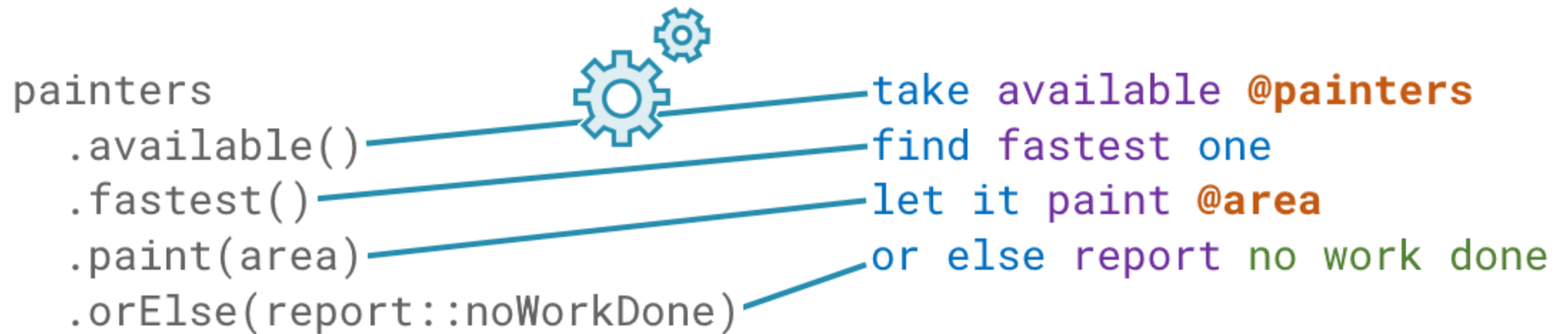
|  
Internal DSL

```
take available @painters  
find fastest one  
let it paint @area  
or else report no work done
```

|  
External DSL

# Introducing Domain-specific Languages

**Compose** elements of code the same way we compose sentences in a spoken language





# Introducing Domain-specific Languages

**Compose** elements of code the same way we compose sentences in a spoken language

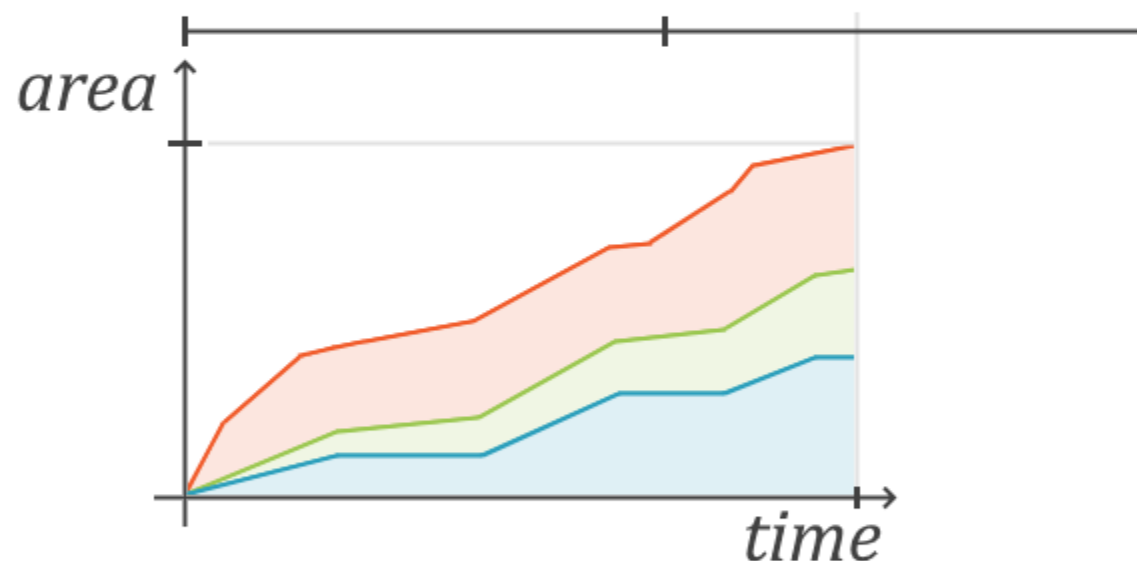
```
painters  
  .available()  
  .fastest()  
  .paint(area)  
  .orElse(report::noWorkDone)
```

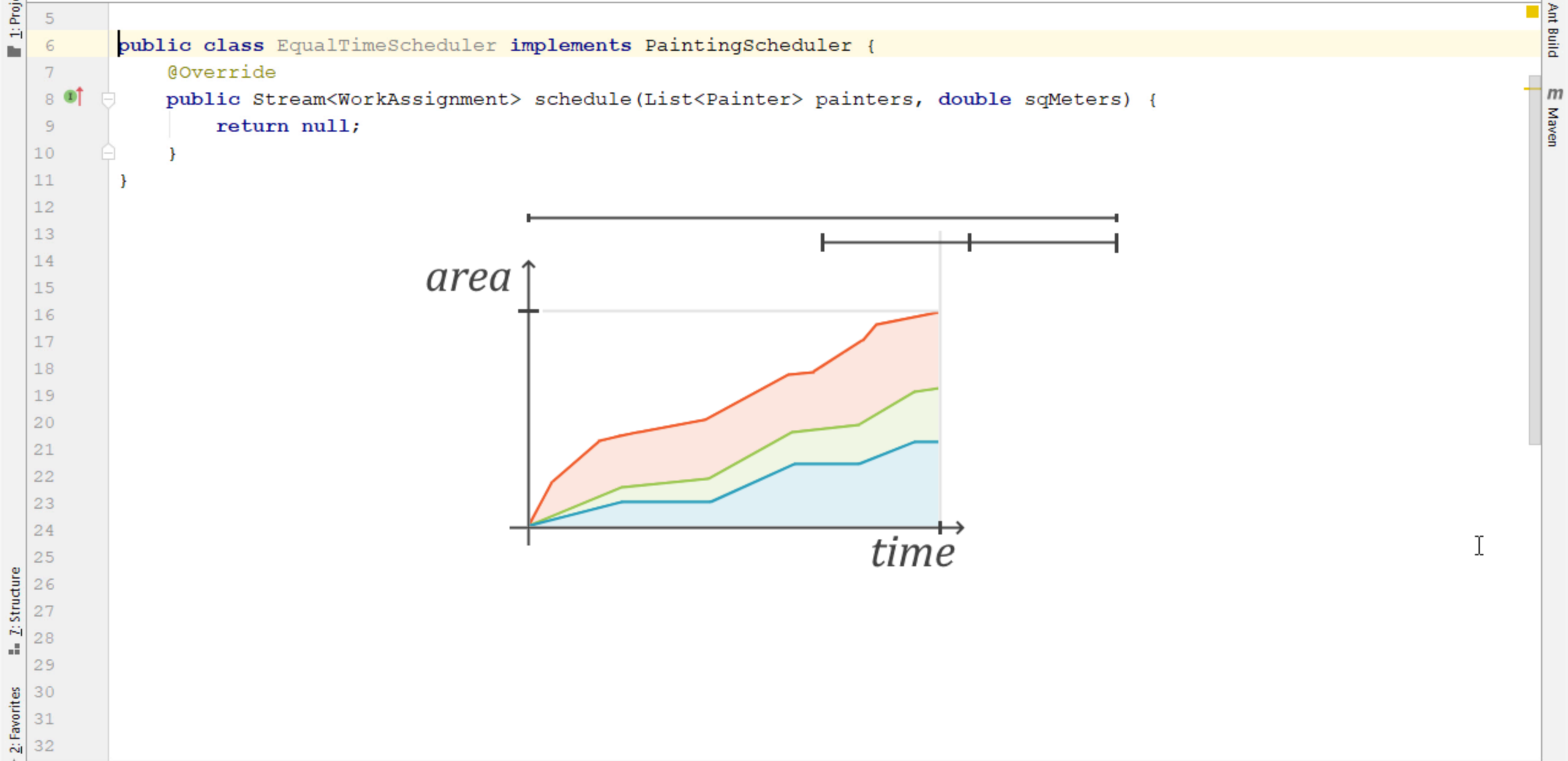
Used by developers

```
take available @painters  
find fastest one  
let it paint @area  
or else report no work done
```

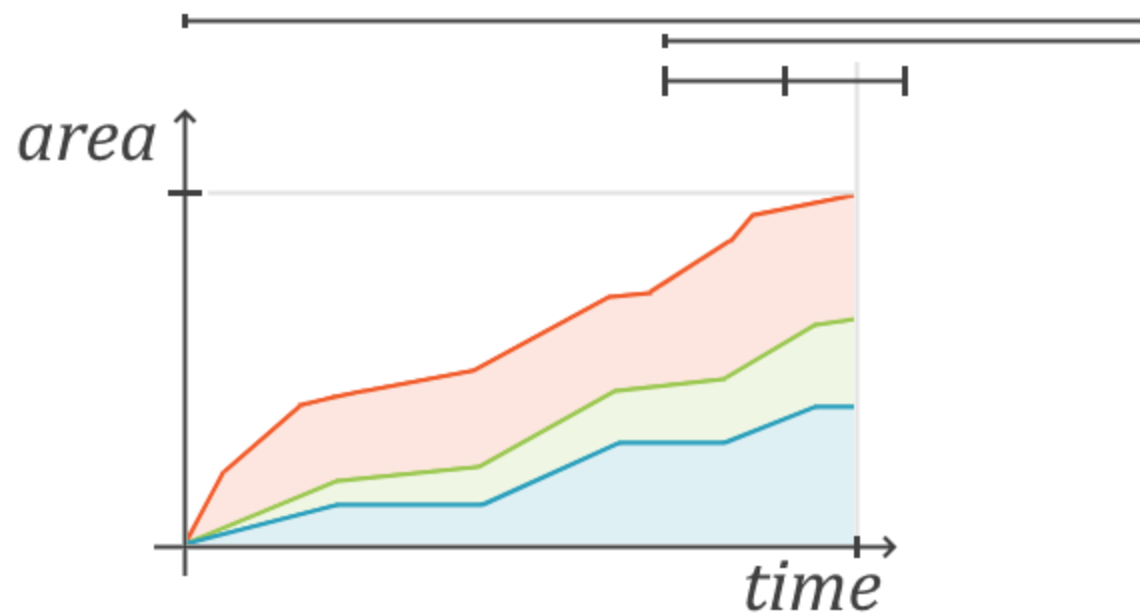
Used by customers

```
5  
6 public class EqualTimeScheduler implements PaintingScheduler {  
7     @Override  
8     public Stream<WorkAssignment> schedule(List<Painter> painters, double sqMeters) {  
9         return null;  
10    }  
11 }  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32
```





```
5  
6 public class EqualTimeScheduler implements PaintingScheduler {  
7     @Override  
8     public Stream<WorkAssignment> schedule(List<Painter> painters, double sqMeters) {  
9         return null;  
10    }  
11 }
```



★ 2: Favorites   ■ ■ 7: Structure



★ 2: Favorites

Ant Build Maven

★ 2: Favorites

### EqualTimeScheduler

Event Log

1: Project

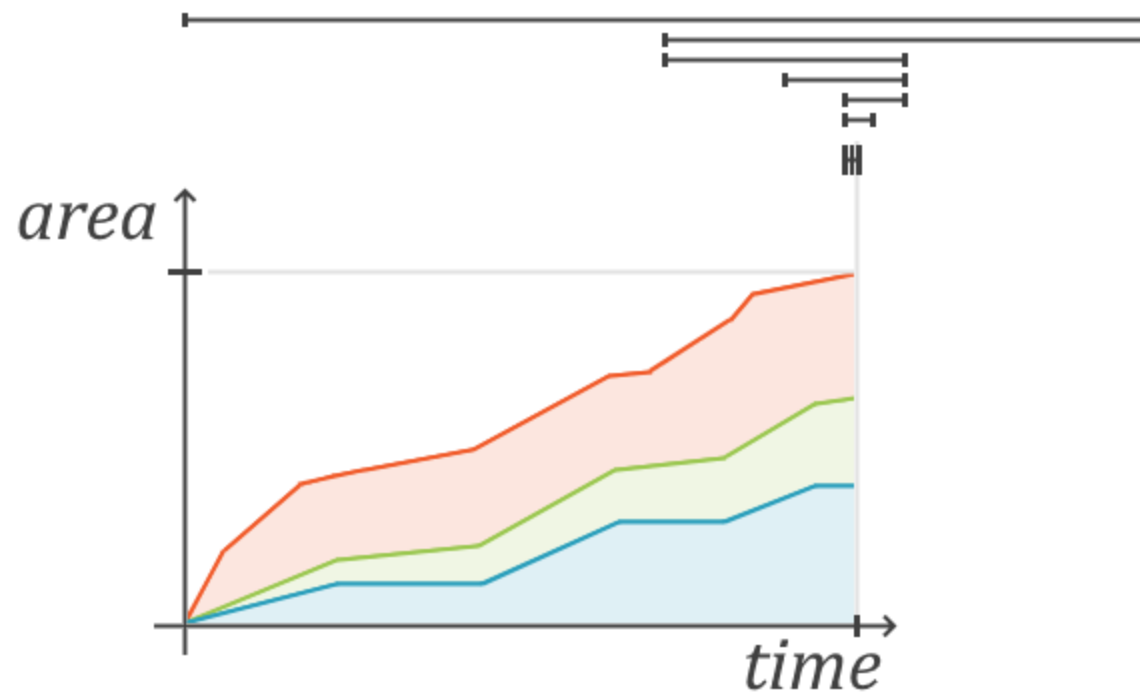
2: Structure

3: Favorites

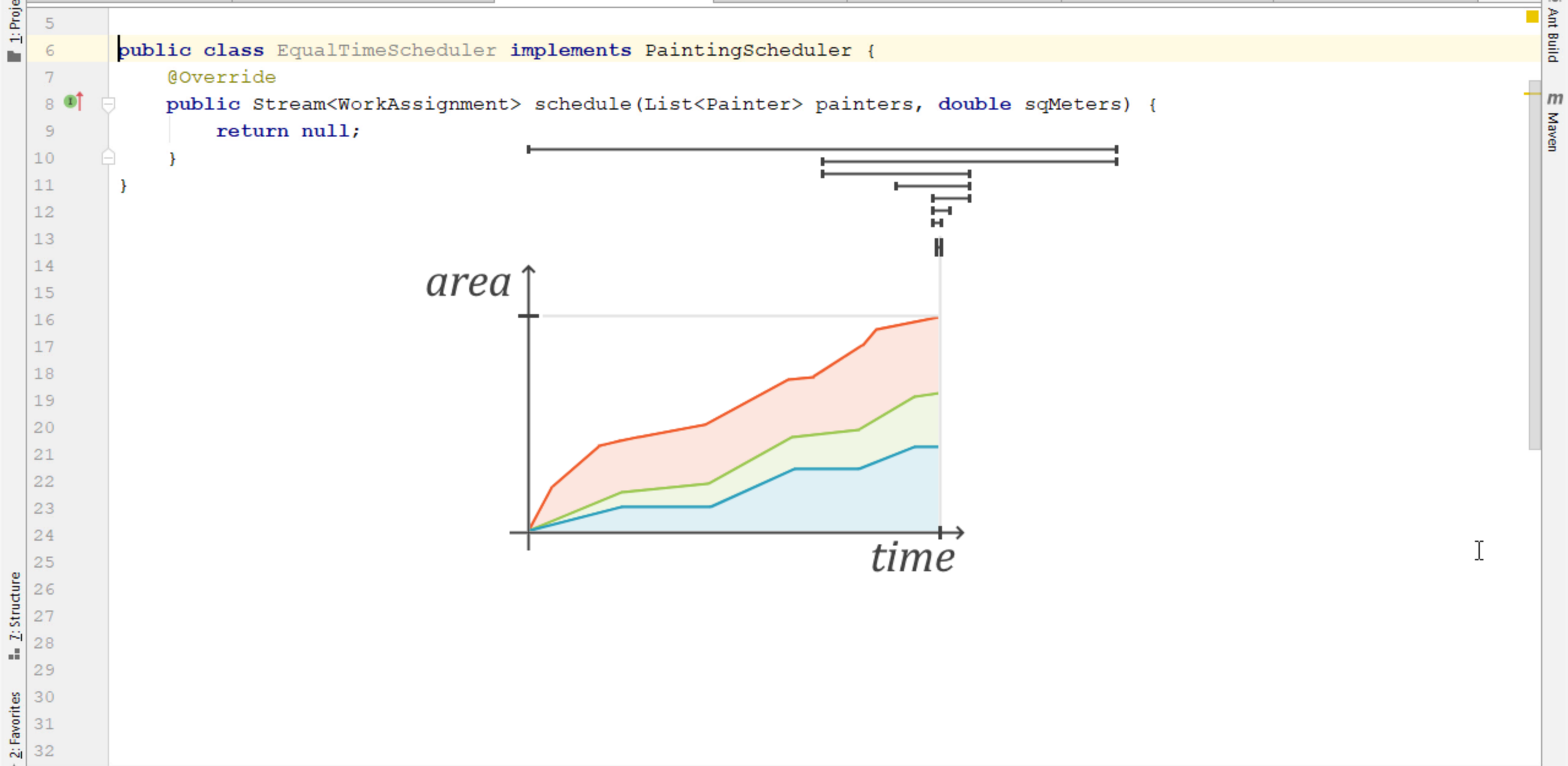
Ant Build  
Maven

PaintingScheduler.java x ConstantVelocityScheduler.java x EqualTimeScheduler.java x Painter.java x ProportionalPainter.java x CompressorPainter.java x CompositePainter.java x

```
5  
6 public class EqualTimeScheduler implements PaintingScheduler {  
7     @Override  
8     public Stream<WorkAssignment> schedule(List<Painter> painters, double sqMeters) {  
9         return null;  
10    }  
11 }  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32
```







1: Project

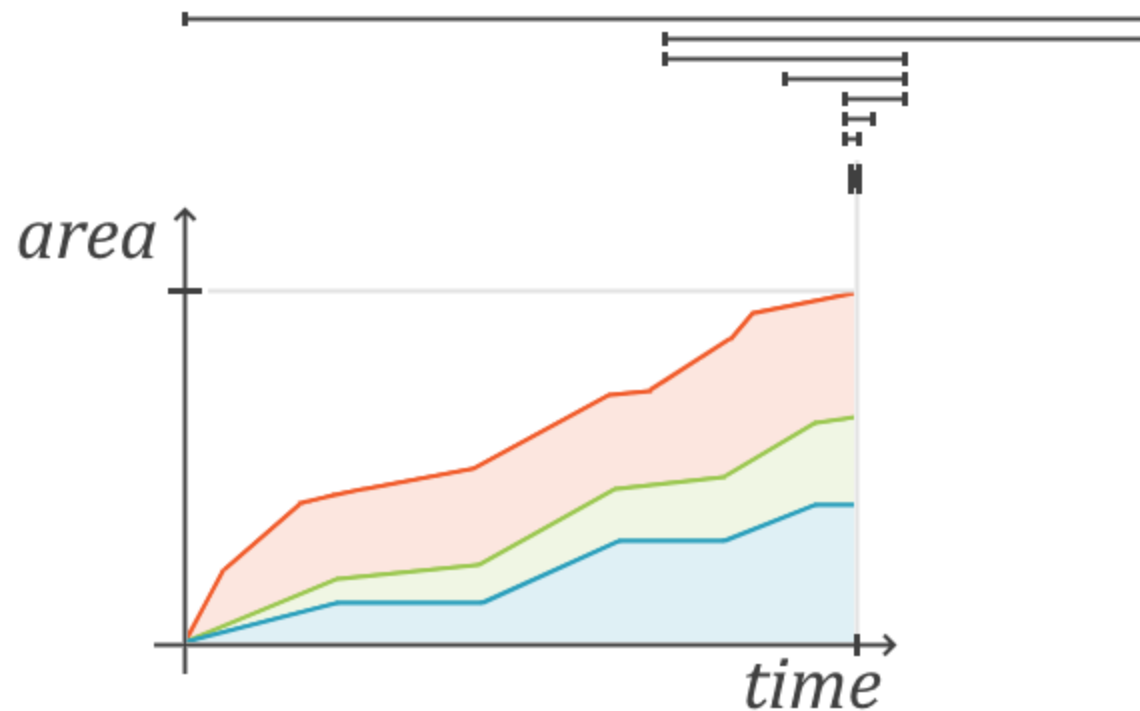
2: Structure

3: Favorites

Ant Build  
Maven

PaintingScheduler.java x ConstantVelocityScheduler.java x EqualTimeScheduler.java x Painter.java x ProportionalPainter.java x CompressorPainter.java x CompositePainter.java x

```
5  
6 public class EqualTimeScheduler implements PaintingScheduler {  
7     @Override  
8     public Stream<WorkAssignment> schedule(List<Painter> painters, double sqMeters) {  
9         return null;  
10    }  
11 }  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32
```



```
5
6 public class EqualTimeScheduler implements PaintingScheduler {
7     @Override
8     public Stream<WorkAssignment> schedule(List<Painter> painters, double sqMeters) {
9         return null;
10    }
11 }
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```



The diagram illustrates the EqualTimeScheduler algorithm. The top part shows a Gantt chart with a long horizontal bar at the top, representing the total work, and several smaller bars below it, representing the execution of multiple painters. The bottom part shows a graph with 'area' on the y-axis and 'time' on the x-axis. Three lines (red, green, blue) represent the cumulative work assigned to three different painters over time. The red line is the highest, followed by the green line, and then the blue line. The area under the red line is shaded red, the area under the green line is shaded green, and the area under the blue line is shaded blue. The total area under the red line represents the total work assigned to the painters over time.

PluralsightsrccomcodinghelmetmoreoojavaEqualTimeScheduler

PaintingScheduler.javaxConstantVelocityScheduler.javaxEqualTimeScheduler.javaxPainter.javaxProportionalPainter.javaxCompressorPainter.javaxCompositePainter.javax

1: Project

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

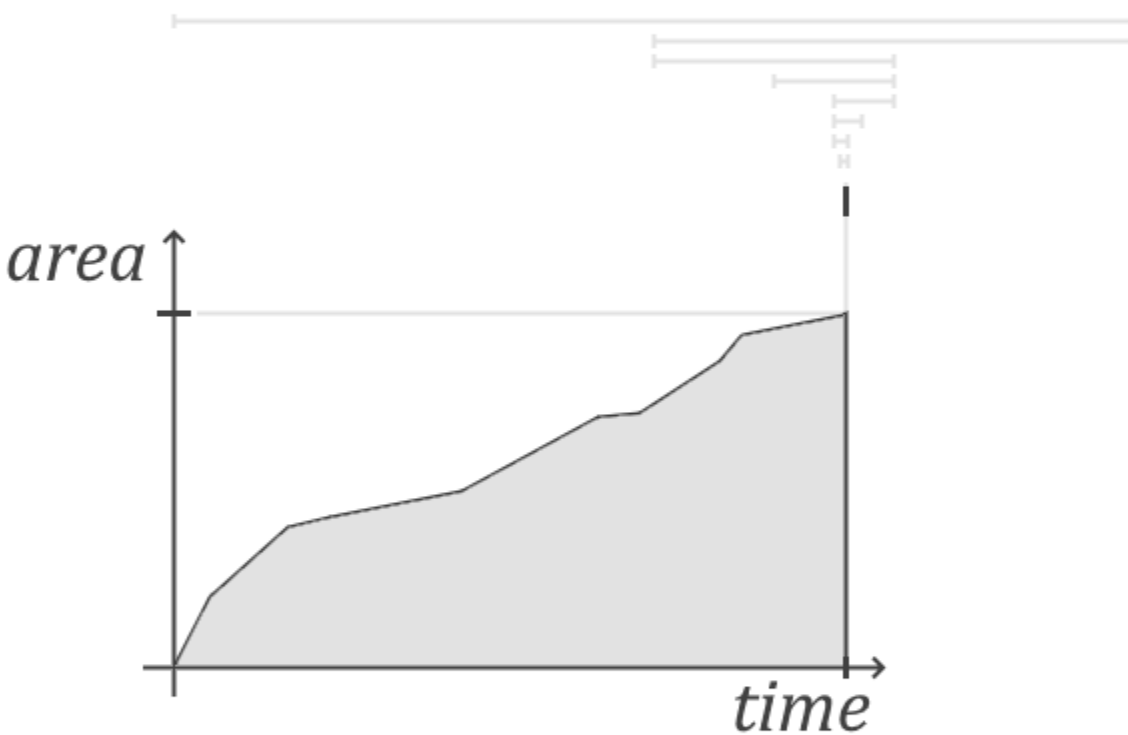
31

32

Ant Build

Maven

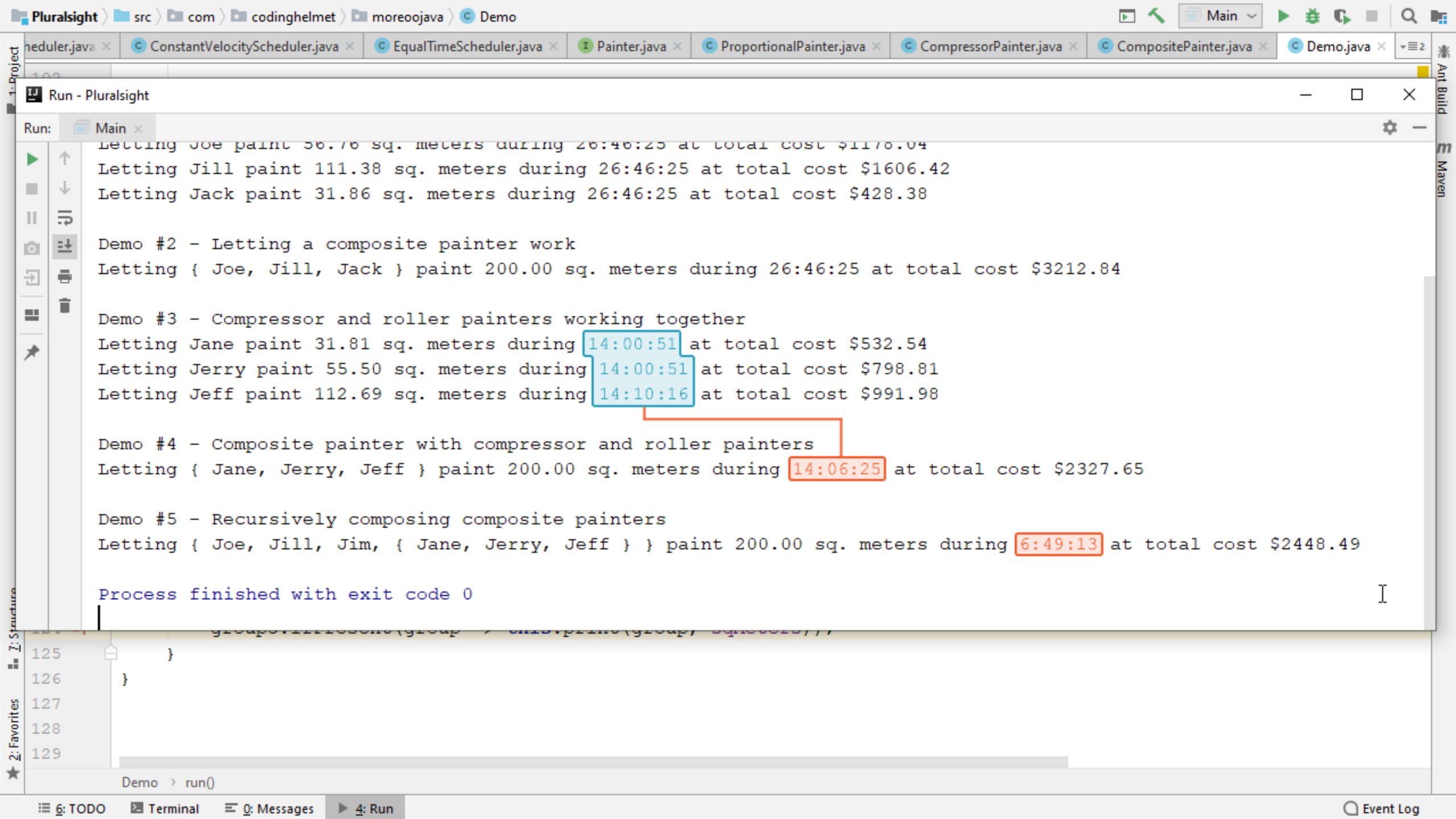
```
public class EqualTimeScheduler implements PaintingScheduler {  
    @Override  
    public Stream<WorkAssignment> schedule(List<Painter> painters, double sqMeters) {  
        return null;  
    }  
}
```



EqualTimeScheduler

6: TODOTerminal4: Run

Event Log



```
Letting Joe paint 56.76 sq. meters during 26:46:25 at total cost $1176.04
Letting Jill paint 111.38 sq. meters during 26:46:25 at total cost $1606.42
Letting Jack paint 31.86 sq. meters during 26:46:25 at total cost $428.38

Demo #2 - Letting a composite painter work
Letting { Joe, Jill, Jack } paint 200.00 sq. meters during 26:46:25 at total cost $3212.84

Demo #3 - Compressor and roller painters working together
Letting Jane paint 31.81 sq. meters during 14:00:51 at total cost $532.54
Letting Jerry paint 55.50 sq. meters during 14:00:51 at total cost $798.81
Letting Jeff paint 112.69 sq. meters during 14:10:16 at total cost $991.98

Demo #4 - Composite painter with compressor and roller painters
Letting { Jane, Jerry, Jeff } paint 200.00 sq. meters during 14:06:25 at total cost $2327.65

Demo #5 - Recursively composing composite painters
Letting { Joe, Jill, Jim, { Jane, Jerry, Jeff } } paint 200.00 sq. meters during 6:49:13 at total cost $2448.49

Process finished with exit code 0
```

# Summary



## Expanding the deep domain model

- Inventing a domain-specific language
- Demonstrated an internal DSL
- Expressed in the programming language

# Summary



## Consuming an internal DSL

- Used by programmers on the project
- Lets them chain atomic transforms
- Leads to more expressive code
- Code is intention-revealing by design

# Summary



## Preconditions to developing a DSL

- Objects and methods must be composable
- Each operation is small and isolated
- Operations return composable objects
- Chaining atomic transforms to build complex behavior





# Summary

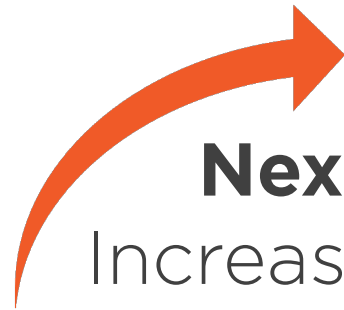


## Implementing composability

- Avoiding mutable methods
- Methods construct new objects
- Heavy use of immutable objects



# Summary



**Next module:**  
Increasing Flexibility by Removing  
Enums and Switch Statements

