

Untangling Operations from Structure on Business Data

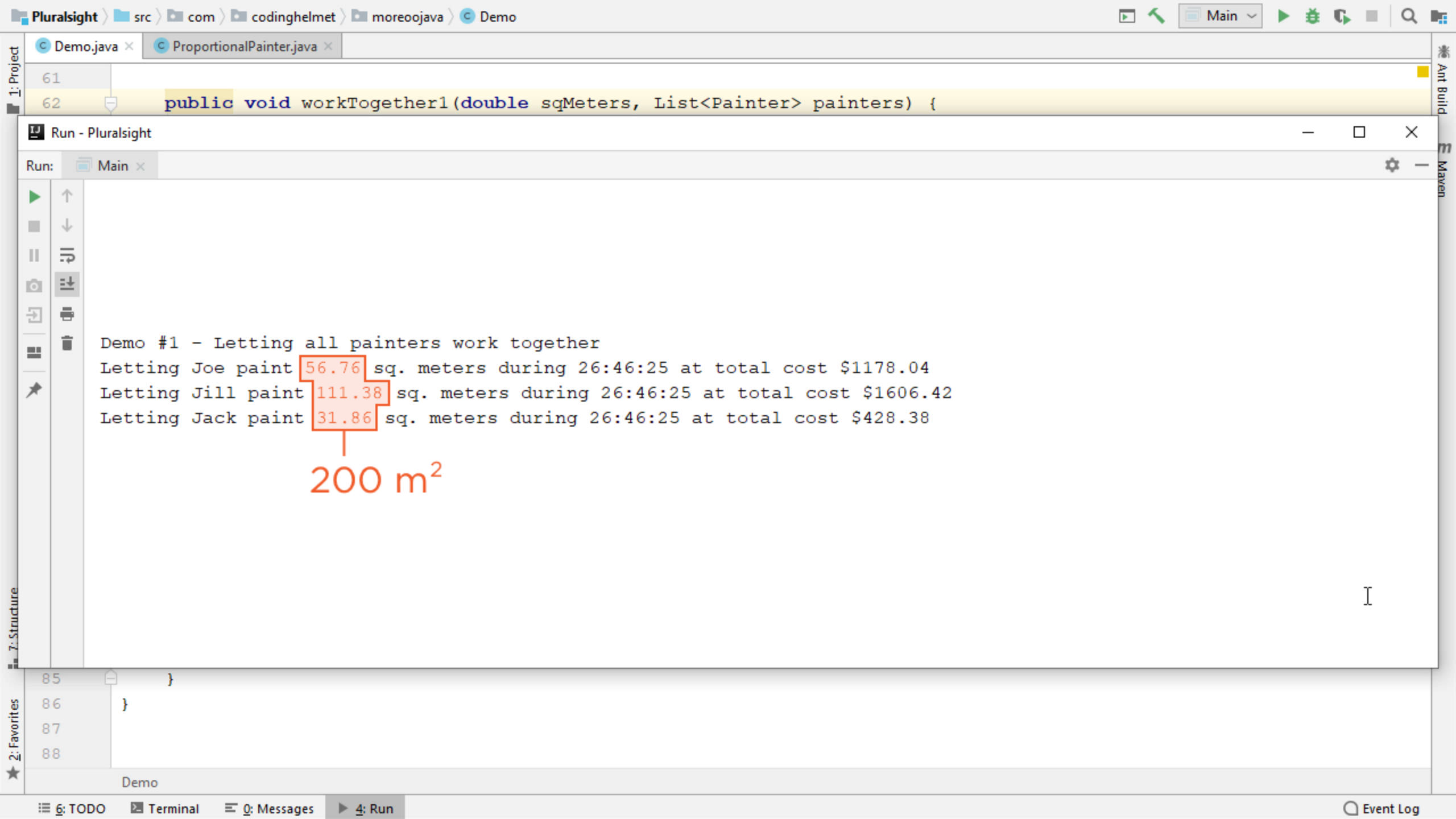


Zoran Horvat

CEO AT CODING HELMET

@zoranh75 <http://codinghelmet.com>





```
public void workTogether1(double sqMeters, List<Painter> painters) {
```

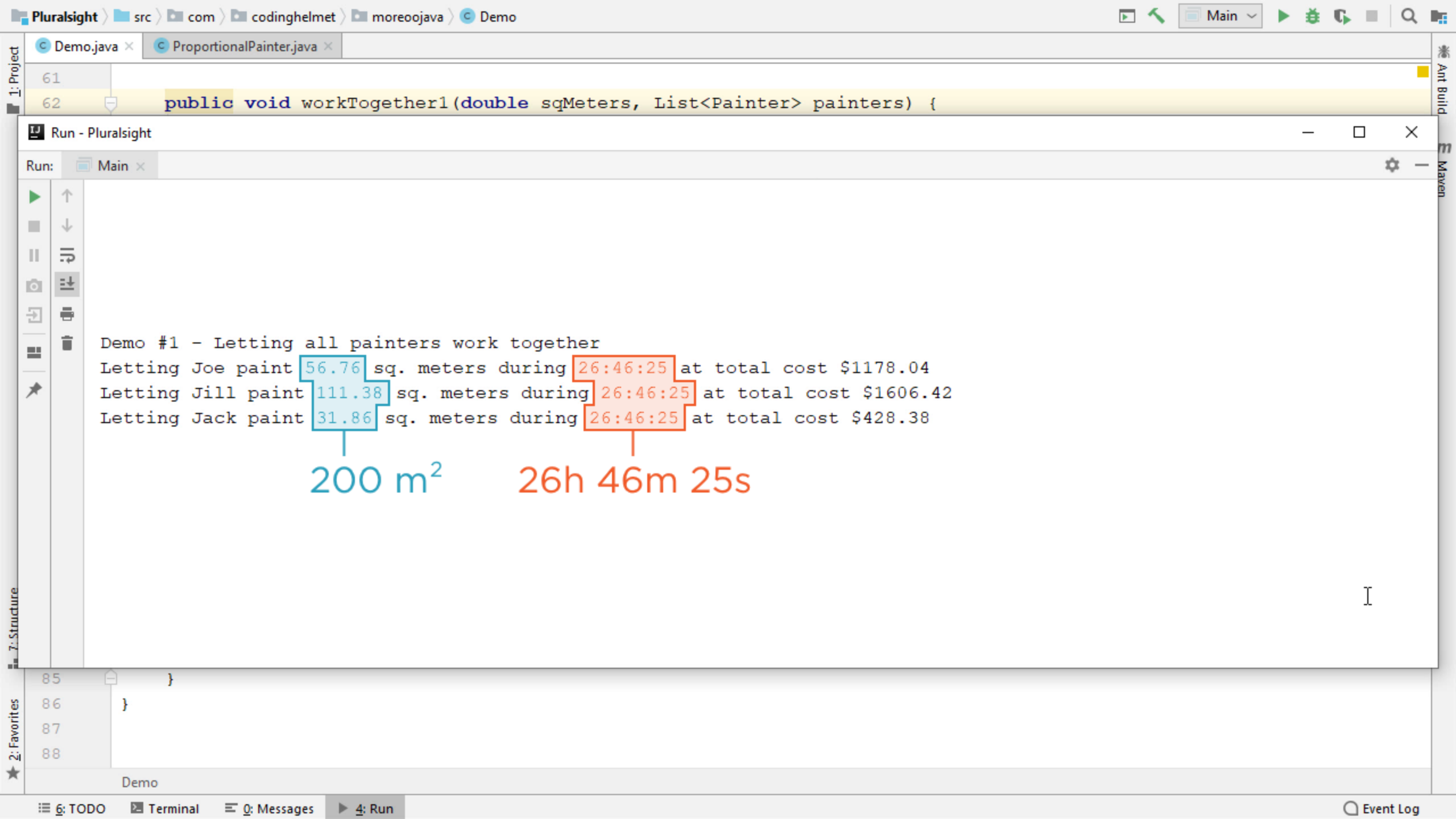
Demo #1 - Letting all painters work together

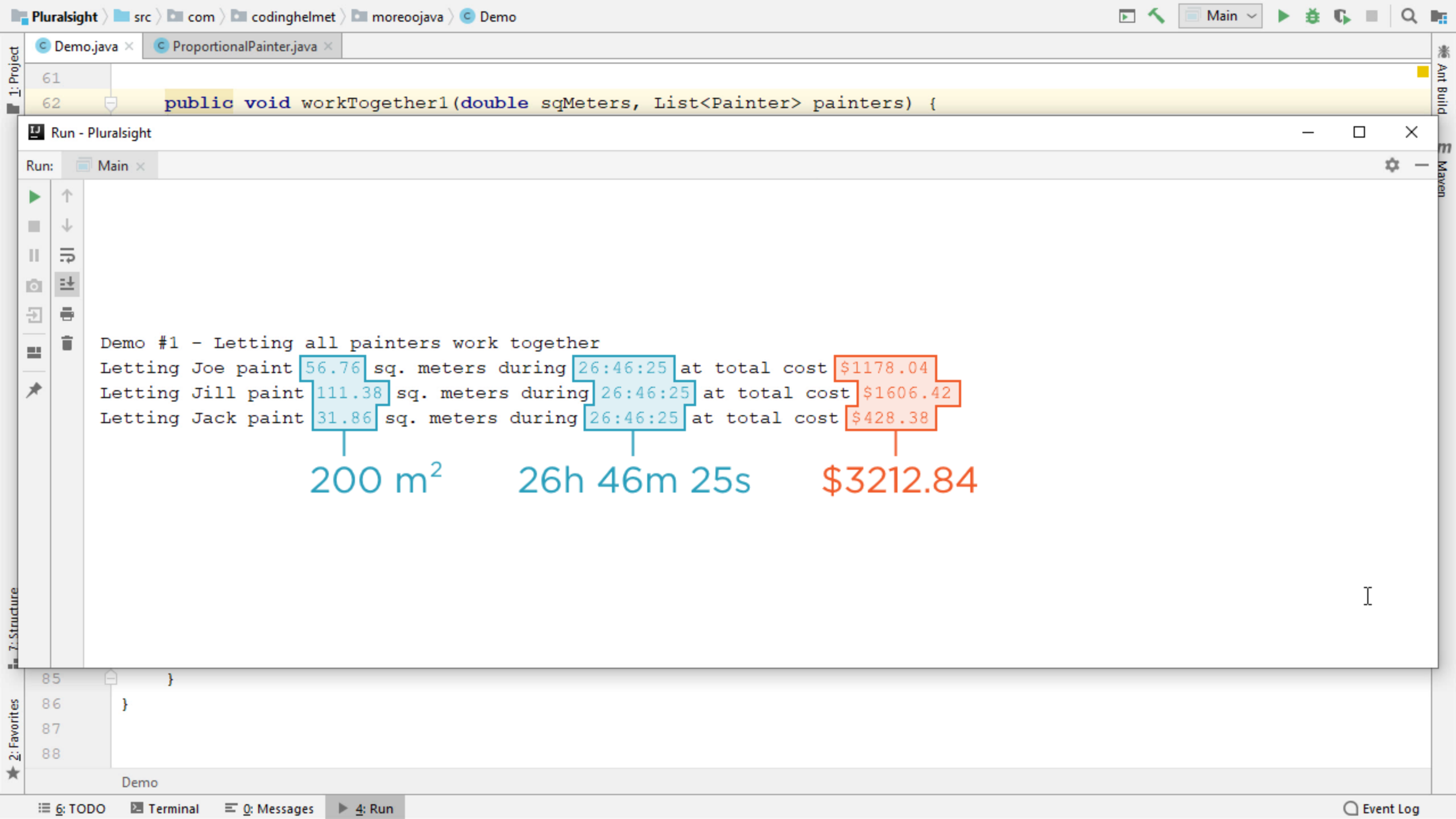
Letting Joe paint 56.76 sq. meters during 26:46:25 at total cost \$1178.04

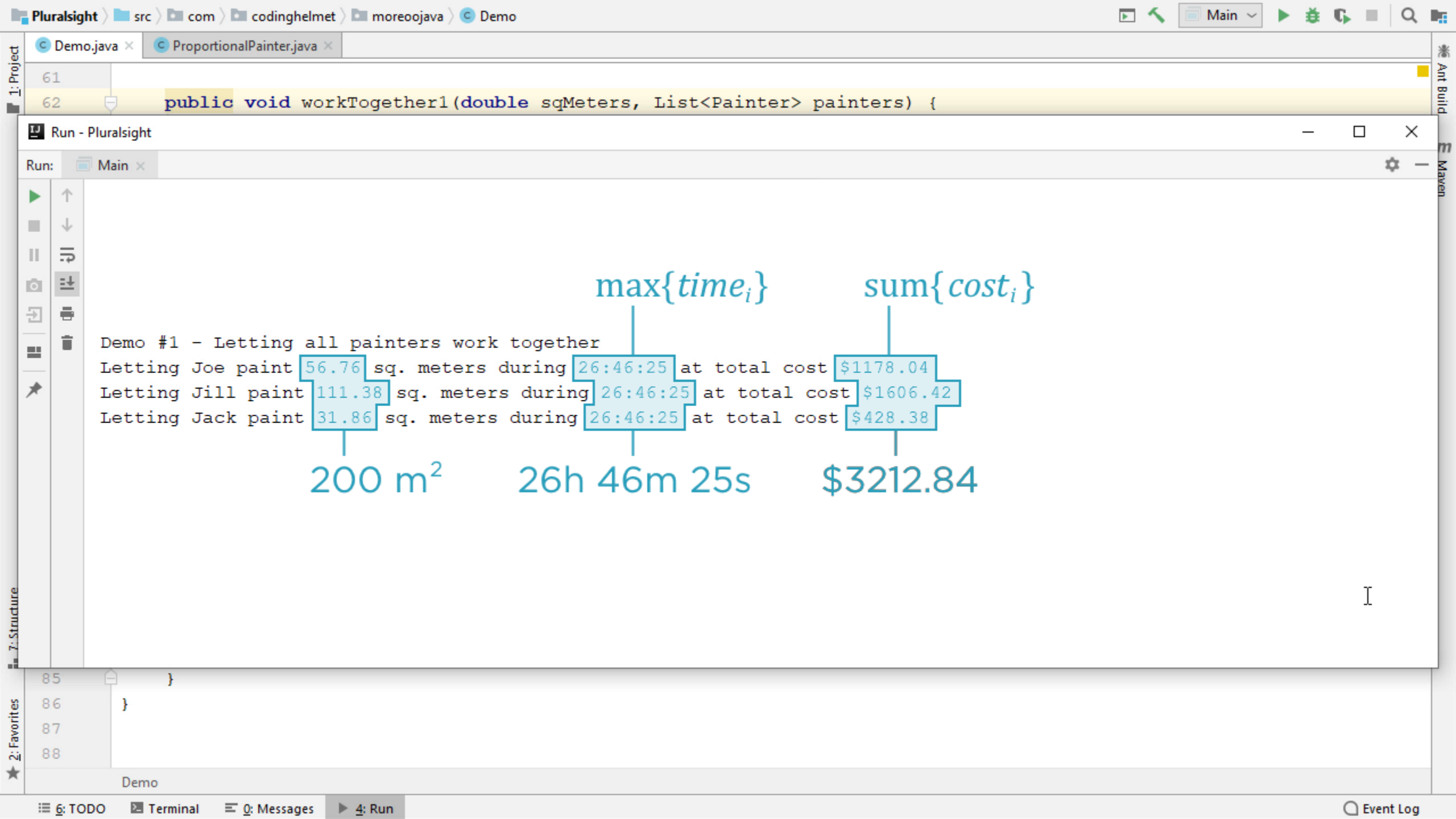
Letting Jill paint 111.38 sq. meters during 26:46:25 at total cost \$1606.42

Letting Jack paint 31.86 sq. meters during 26:46:25 at total cost \$428.38

200 m²







Pluralsight

src > com > codinghelmet > moreoojava > CompositePainter

Main

Ant Build Maven

Demo.java x Painter.java x CompositePainter.java x ProportionalPainter.java x

12

13 public CompositePainter(List<Painter> painters) {

14 this.painters = painters;

15 }

16

17 @Override

18 public boolean isAvailable() {

19 }

20

21 @Override

22 public Duration estimateTimeToPaint(double sqMeters) {

23 }

24

25 @Override

26 public Money estimateCompensation(double sqMeters) {

27 }

28

29 @Override

30 public String getName() {

31 return this.getPainterNames()

32 .collect(Collectors.joining(delimiter: ", ", prefix: "{ ", suffix: " }"));

33 }

34

35 private Stream<String> getPainterNames() {

36 return Painter.stream(this.painters).map(Painter::getName);

37 }

38 }

39 }

1: Project

2: Favorites

3: Structure

CompositePainter

6: TODO

Terminal

0: Messages

4: Run

Event Log

Composite

Split

Contained elements

Pluralsight

src>com>codinghelmet>moreoojava>CompositePainter

Main

Ant Build Maven

Demo.java

Painter.java

CompositePainter.java

ProportionalPainter.java

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

```
public CompositePainter(List<Painter> painters) {
    this.painters = painters;
}

@Override
public boolean isAvailable() {
}

@Override
public Duration estimateTimeToPaint(double sqMeters) {
}

@Override
public Money estimateCompensation(double sqMeters) {
}

@Override
public String getName() {
    return this.getPainterNames()
        .collect(Collectors.joining( delimiter: ", ", prefix: "{ ", suffix: " }"));
}

private Stream<String> getPainterNames() {
    return Painter.stream(this.painters).map(Painter::getName);
}
}
```

Composite

Collect

Contained elements

CompositePainter

6: TODO

Terminal

0: Messages

4: Run

Event Log

Pluralsight

src > com > codinghelmet > moreoojava > CompressorPainter

Main

1: Project

Demo.java x CompressorPainter.java x Painter.java x CompositePainter.java x ProportionalPainter.java x

7

8 public class CompressorPainter implements Painter {

9

10 private Duration fillTime;

11

12

13

14

15

16 public CompressorPainter(

17 String name, Duration fillTime,

18 double fillAfterSqMeters, Duration cleaningTime,

19 double sqMetersPerHour, MoneyRate rate) {

20

21 this.name = name;

22 this.fillTime = fillTime;

23 this.fillAfterSqMeters = fillAfterSqMeters;

24 this.cleaningTime = cleaningTime;

25 this.sqMetersPerHour = sqMetersPerHour;

26 this.rate = rate;

27 }

28

29 @Override

30 public String getName() { return this.name; }

31

32 @Override

33 public boolean isAvailable() { return true; }

34

2: Favorites

2: Structure

CompressorPainter

6: TODO

Terminal

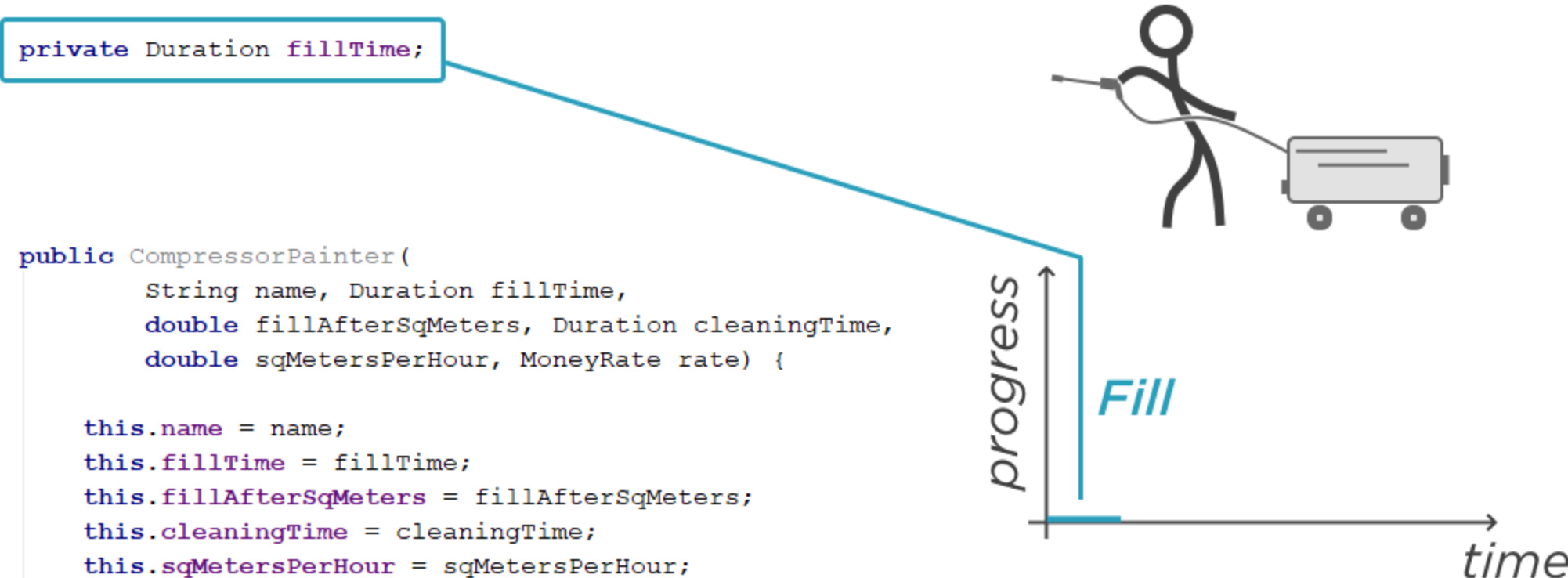
0: Messages

4: Run

Event Log

Ant Build

Maven



Pluralsight

src > com > codinghelmet > moreoojava > CompressorPainter

Main

Ant Build Maven

1: Project

2: Favorites

3: Structure

Demo.java

CompressorPainter.java

Painter.java

CompositePainter.java

ProportionalPainter.java

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

33

34

public class CompressorPainter implements Painter {

private double fillAfterSqMeters;

public CompressorPainter(
String name, Duration fillTime,
double fillAfterSqMeters, Duration cleaningTime,
double sqMetersPerHour, MoneyRate rate) {

this.name = name;
this.fillTime = fillTime;
this.fillAfterSqMeters = fillAfterSqMeters;
this.cleaningTime = cleaningTime;
this.sqMetersPerHour = sqMetersPerHour;
this.rate = rate;
}

@Override
public String getName() { return this.name; }

@Override
public boolean isAvailable() { return true; }
}

Refill

progress

time

CompressorPainter

6: TODO

Terminal

0: Messages

4: Run

Event Log

Pluralsight

src > com > codinghelmet > moreoojava > CompressorPainter

Main

Ant BuildMaven

1: Project

2: Favorites

3: Structure

Demo.java

CompressorPainter.java

Painter.java

CompositePainter.java

ProportionalPainter.java

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

33

34

public class CompressorPainter implements Painter {


private double fillAfterSqMeters;


public CompressorPainter(
String name, Duration fillTime,
double fillAfterSqMeters, Duration cleaningTime,
double sqMetersPerHour, MoneyRate rate) {

this.name = name;
this.fillTime = fillTime;
this.fillAfterSqMeters = fillAfterSqMeters;
this.cleaningTime = cleaningTime;
this.sqMetersPerHour = sqMetersPerHour;
this.rate = rate;
}

@Override
public String getName() { return this.name; }

@Override
public boolean isAvailable() { return true; }
}





CompressorPainter

6: TODO

Terminal

0: Messages

4: Run

Event Log

Pluralsight

src > com > codinghelmet > moreoojava > CompressorPainter

Main

Ant Build Maven

1: Project

2: Structure

2: Favorites

Demo.java

CompressorPainter.java

Painter.java

CompositePainter.java

ProportionalPainter.java

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

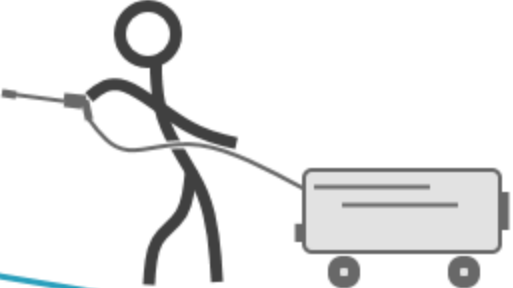
31

32

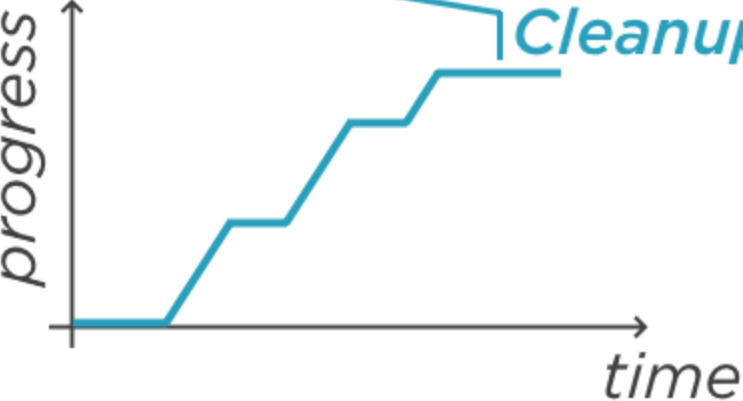
33

34

```
public class CompressorPainter implements Painter {  
  
    private Duration cleaningTime;  
  
    public CompressorPainter(  
        String name, Duration fillTime,  
        double fillAfterSqMeters, Duration cleaningTime,  
        double sqMetersPerHour, MoneyRate rate) {  
  
        this.name = name;  
        this.fillTime = fillTime;  
        this.fillAfterSqMeters = fillAfterSqMeters;  
        this.cleaningTime = cleaningTime;  
        this.sqMetersPerHour = sqMetersPerHour;  
        this.rate = rate;  
    }  
  
    @Override  
    public String getName() { return this.name; }  
  
    @Override  
    public boolean isAvailable() { return true; }  
}
```



Cleanup



CompressorPainter

6: TODO

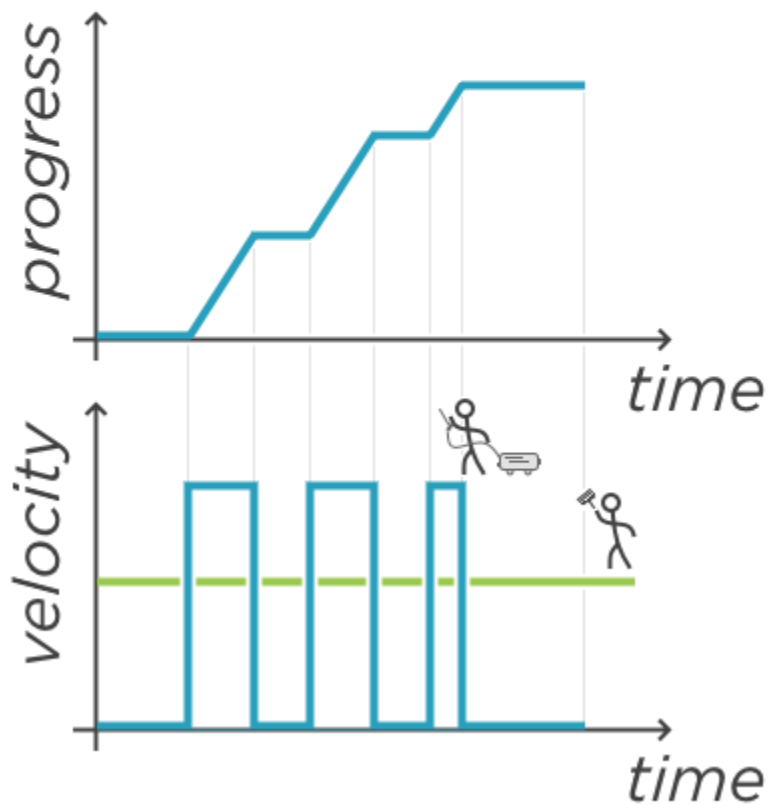
Terminal

0: Messages

4: Run

Event Log

```
7
8 public class CompressorPainter implements Painter {
9     private String name;
10    private Duration fillTime;
11    private double fillAfterSqMeters;
12    private Duration cleaningTime;
13    private double sqMetersPerHour;
14    private MoneyRate rate;
15
16    public CompressorPainter(
17        String name, Duration fillTime,
18        double fillAfterSqMeters, Duration cleaningTime,
19        double sqMetersPerHour, MoneyRate rate) {
20
21        this.name = name;
22        this.fillTime = fillTime;
23        this.fillAfterSqMeters = fillAfterSqMeters;
24        this.cleaningTime = cleaningTime;
25        this.sqMetersPerHour = sqMetersPerHour;
26        this.rate = rate;
27    }
28
29    @Override
30    public String getName() { return this.name; }
31
32    @Override
33    public boolean isAvailable() { return true; }
34
```



```
31
32 @Override
33 public Duration estimateTimeToPaint(double sqMeters) {
34     return this.estimateTimeToPaint(sqMeters, this.estimateTotalVelocity(sqMeters));
35 }
36
37 @
38 private Duration estimateTimeToPaint(double sqMeters, Velocity totalVelocity) {
39     return Painter.stream(this.painters) PaintersStream
40         .map(painter -> painter.estimateTimeToPaint(
41             sqMeters: sqMeters * painter.estimateVelocity(sqMeters).divideBy(totalVelocity))) Stream<Duration>
42         .max(Duration::compareTo) Optional<Duration>
43         .get();
44 }
45
46 private Velocity estimateTotalVelocity(double sqMeters) {
47     return Painter.stream(this.painters) PaintersStream
48         .map(painter -> painter.estimateVelocity(sqMeters)) Stream<Velocity>
49         .reduce(Velocity::add) Optional<Velocity>
50         .orElse(Velocity.ZERO);
51 }
52
53 @Override
54 public Money estimateCompensation(double sqMeters) {
55     return this.estimateCompensation(sqMeters, this.estimateTotalVelocity(sqMeters));
56 }
57
58 private Money estimateCompensation(double sqMeters, Velocity totalVelocity) {
59     return Painter.stream(this.painters) PaintersStream
```

Presumes constant velocity



Summary



Generalizing an algorithm

- Make it apply to different situations
- Either allow some code duplication...
- Or restructure the algorithm



Summary



Implementing the Strategy pattern

- Recognize universal parts
- Extract varying parts out
- Wrap varying parts into a class
- Inject a concrete strategy into the consumer



Summary



Consuming Strategies

- Compose algorithms from parts
- Algorithm outline delegates to strategies
- That makes the algorithm general
- When the problem changes, substitute the strategy object

Summary



Beyond deep domain model

- Developing domain-specific language
- Use the deep and complex model to develop even better designs



Summary



Next module:

Advancing to a
Domain-specific Language

