

Learning new words

Methods

12 September 2006

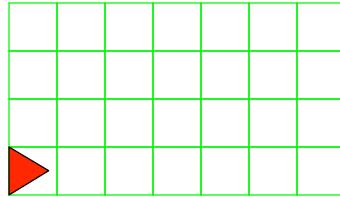


CS111 Computer Programming

Department of Computer Science
Wellesley College

Inventing new worlds*

```
public class PuppyWorld extends BoggleWorld
{
    public void run()
    {
        Boggle spot = new Boggle();
        spot.forward(5);
    } // Run spot run
}
```

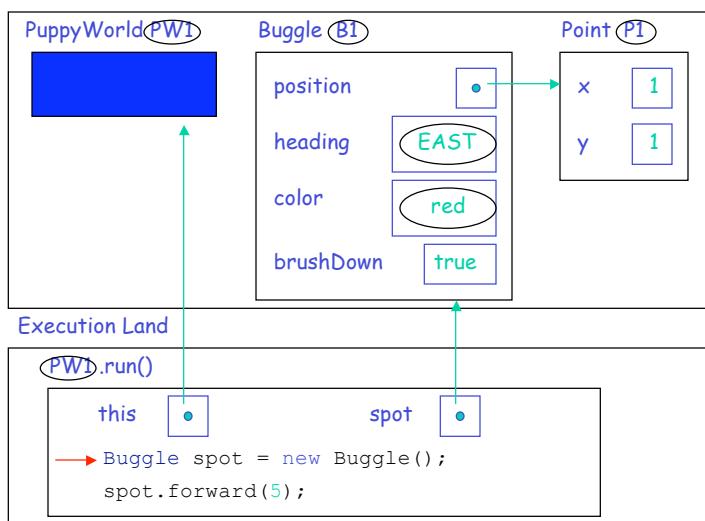


*BoggleWorld is itself a class.

Methods 1 2

The Java Execution Model (JEM)

Object Land



Methods 1 3

In the beginning

Object Land



Execution Land



Methods 1 4

The run() method is invoked

Object Land

PuppyWorld PW1



Execution Land

(PW1).run()

this

```
Bugle spot = new Bugle();
spot.forward(5);
```

Methods 1 5

Execution of run() begins

Object Land

PuppyWorld PW1



Bugle

position

Point

heading

x

1

color

y

1

brushDown

EAST

true

red

Execution Land

(PW1).run

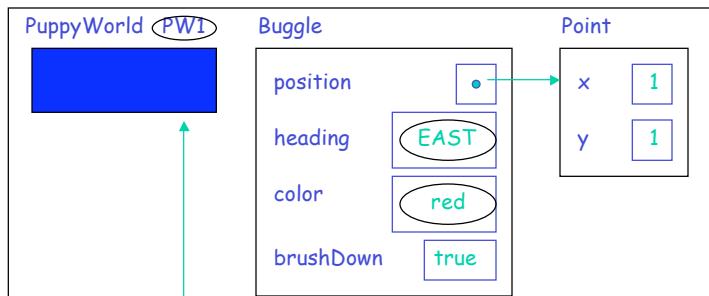
this

```
→ Bugle spot = new Bugle();
spot.forward(5);
```

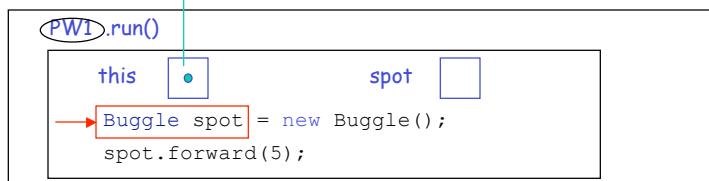
Methods 1 6

The Java Execution Model (JEM)

Object Land



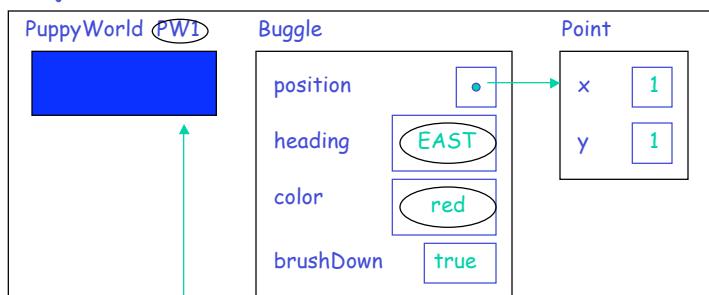
Execution Land



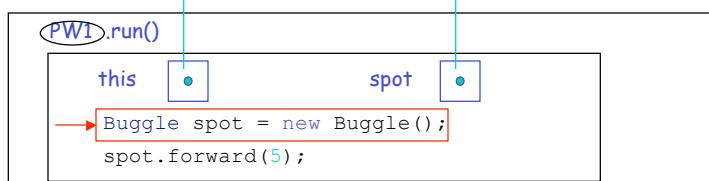
Methods 1 7

The assignment statement executes

Object Land



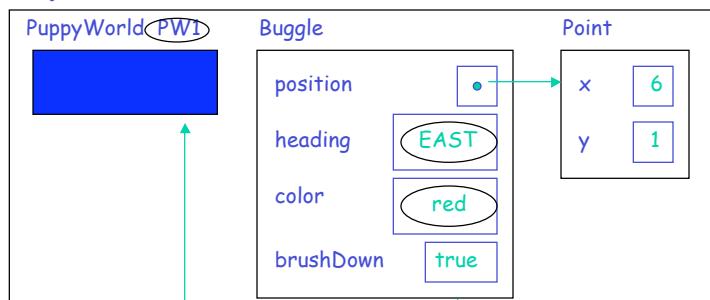
Execution Land



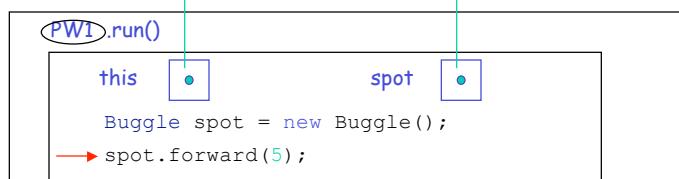
Methods 1 8

Spot runs away

Object Land



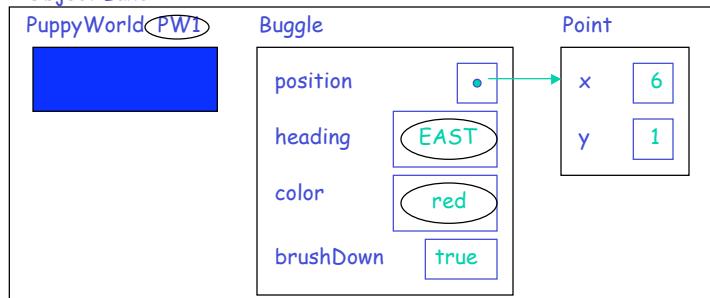
Execution Land



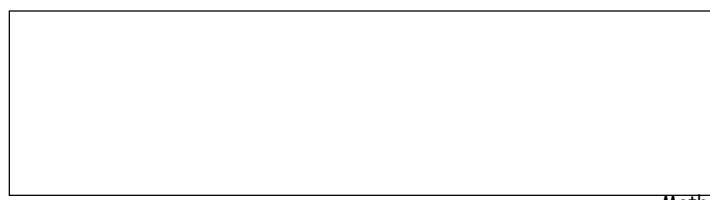
Methods 1 9

run () finishes execution

Object Land



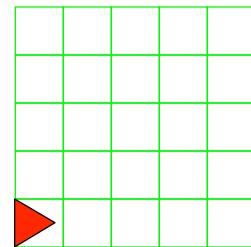
Execution Land



Methods 1 10

Spot does tricks

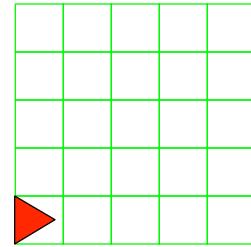
```
public class PuppyWorld extends BuggleWorld
{
    public void run()
    {
        Buggle spot = new Buggle();
        // roll over spot
        spot.left();
        spot.left();
        spot.left();
        spot.left();
        return;
    }
}
```



Methods 1 11

We would like to say ... *

```
public class PuppyWorld extends BuggleWorld
{
    public void run()
    {
        Buggle spot = new Buggle();
        spot.rollover();
    }
}
```

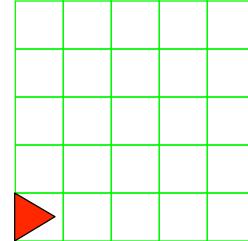


*But spot would not understand. Spot is a Buggle
and Buggles only understand what is in their contract.

Methods 1 12

A new breed of Boggle

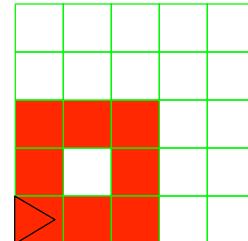
```
public class PuppyWorld extends BoggleWorld
{
    public void run()
    {
        PuppyBoggle spot = new PuppyBoggle();
        spot.rollover();
    }
}
class PuppyBoggle extends Boggle
{
    public void rollover()
    {
        this.left();
        this.left();
        this.left();
        this.left();
        return;
    }
}
```



Methods 1 13

Spot learns to draw

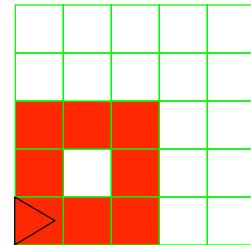
```
public class PuppyWorld extends BoggleWorld
{
    public void run()
    {
        PuppyBoggle spot = new PuppyBoggle();
        spot.dogHouse(); // draws 3x3 doghouse
    }
}
class PuppyBoggle extends Boggle
{
    public void dogHouse()
    {
        // code to draw doghouse
    }
}
```



Methods 1 14

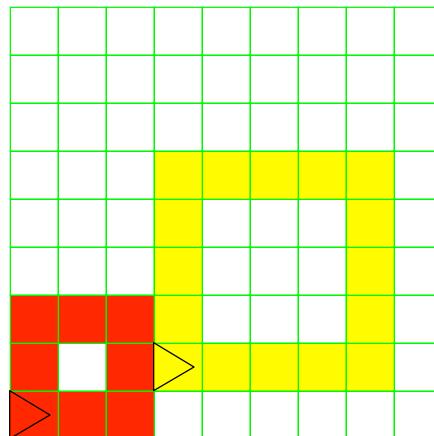
New code for PuppyBuggles

```
public class PuppyWorld extends BuggleWorld
{
    ...
    class PuppyBuggle extends Buggle
    {
        public void dogHouse()
        {
            this.forward(2);
            this.left();
            this.forward(2);
            this.left();
            this.forward(2);
            this.left();
            this.forward(2);
            this.left();
        }
    }
}
```



Methods 1 15

Lassie wants a dogHouse too*



*But collies are much bigger than terriers.

Methods 1 16

We could . . .

```
public class PuppyWorld extends BuggleWorld
{
    public void run()
    {
        PuppyBuggle spot = new PuppyBuggle();
        PuppyBuggle lassie = new PuppyBuggle();
        lassie.setLocation(new Point(4,2));
        lassie.setColor(Color.yellow);
        spot.dogHouse();
        lassie.bigDogHouse();
    }
}
class PuppyBuggle extends Buggle
{
    public void dogHouse() { ... }
    public void bigDogHouse() { ... }
}
```



Methods 1 17

A solution using parameters

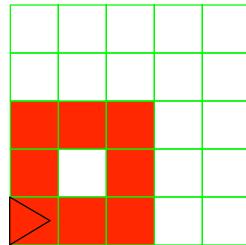
```
public class PuppyWorld extends BuggleWorld
{
    public void run()
    {
        PuppyBuggle spot = new PuppyBuggle();
        PuppyBuggle lassie = new PuppyBuggle();
        lassie.setLocation(new Point(4,2));
        lassie.setColor(Color.yellow);
        spot.dogHouse(3);
        lassie.dogHouse(5);
    }
}
class PuppyBuggle extends Buggle
{
    public void dogHouse(int n) { ... }
}
```



Methods 1 18

Methods with parameters

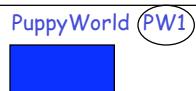
```
public class PuppyWorld extends BuggleWorld
{
    ...
    class PuppyBuggle extends Buggle
    {
        public void dogHouse(int n)
        {
            this.forward(n-1);
            this.left();
            this.forward(n-1);
            this.left();
            this.forward(n-1);
            this.left();
            this.forward(n-1);
            this.left();
        }
    }
}
```



Methods 1 19

JEM traces lassie and spot

Object Land



Execution Land

PW1.run()

```
PuppyBuggle spot = new PuppyBuggle();
PuppyBuggle lassie = new PuppyBuggle();
lassie.setLocation(new Point(4,2));
lassie.setColor(Color.yellow);
spot.dogHouse(3);
lassie.dogHouse(5);
```

Methods 1 20

`run()` begins to run

Object Land
PuppyWorld



Execution Land
`.run()`

`this`

```
PuppyBuggle spot = new PuppyBuggle();
PuppyBuggle lassie = new PuppyBuggle();
lassie.setLocation(new Point(4,2));
lassie.setColor(Color.yellow);
spot.dogHouse(3);
lassie.dogHouse(5);
```

Methods 1 21

A PuppyBuggle is born

Object Land
PuppyWorld



PuppyBuggle

position	(1, 1)
heading	EAST
color	red
brushDown	true

Execution Land
`.run()`

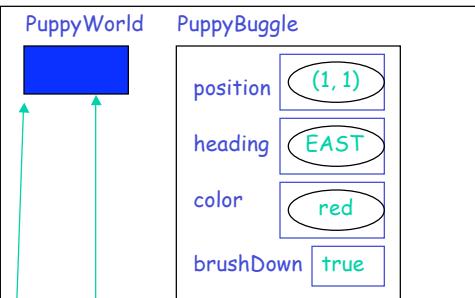
`this`

```
→ PuppyBuggle spot = new PuppyBuggle();
PuppyBuggle lassie = new PuppyBuggle();
lassie.setLocation(new Point(4,2));
lassie.setColor(Color.yellow);
spot.dogHouse(3);
lassie.DogHouse(5);
```

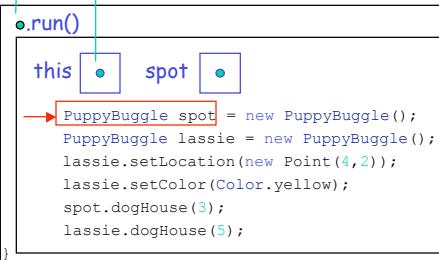
Methods 1 22

A name is chosen . . .

Object Land



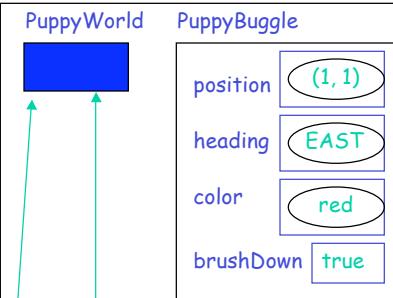
Execution Land



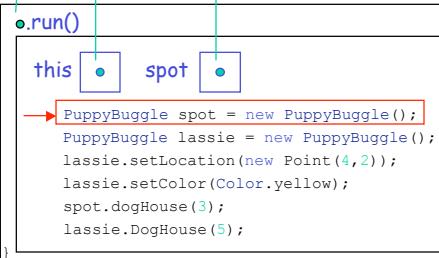
Methods 1 23

. . . and given to the new PuppyBuggle

Object Land

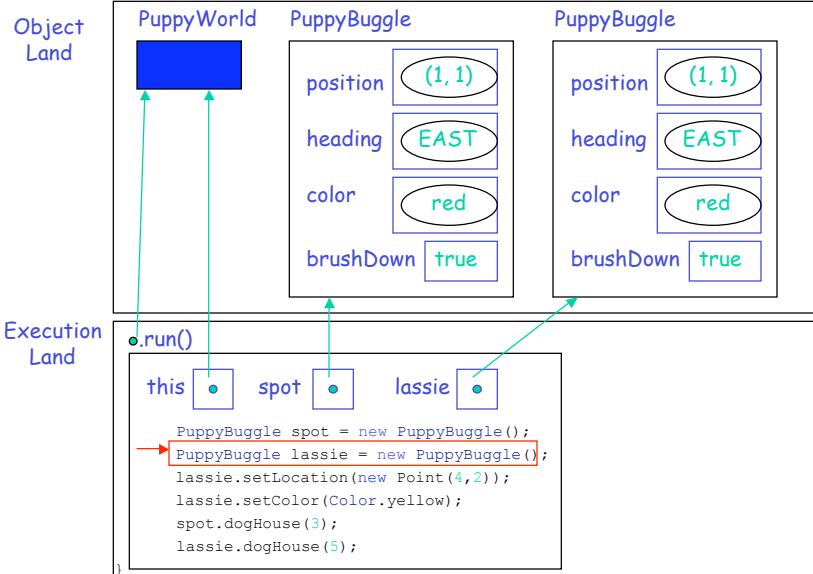


Execution Land



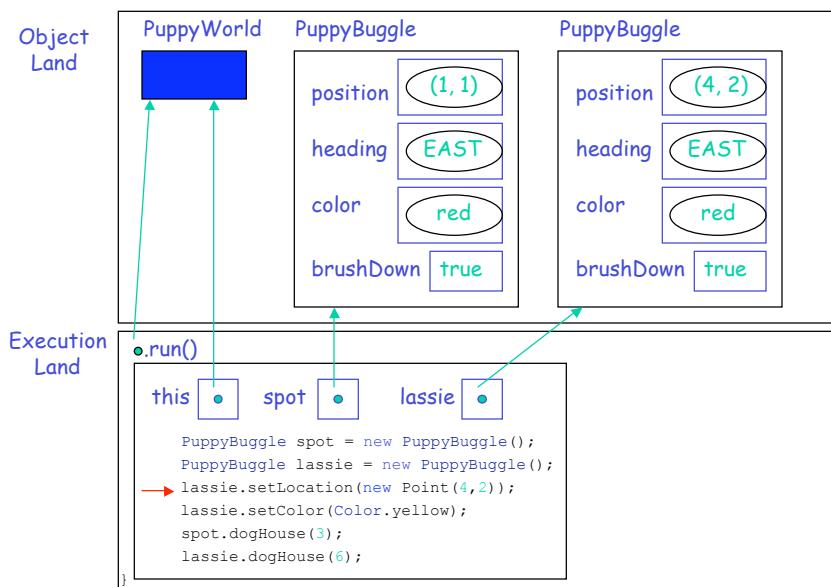
Methods 1 24

Lassie enters PuppyWorld in the same way



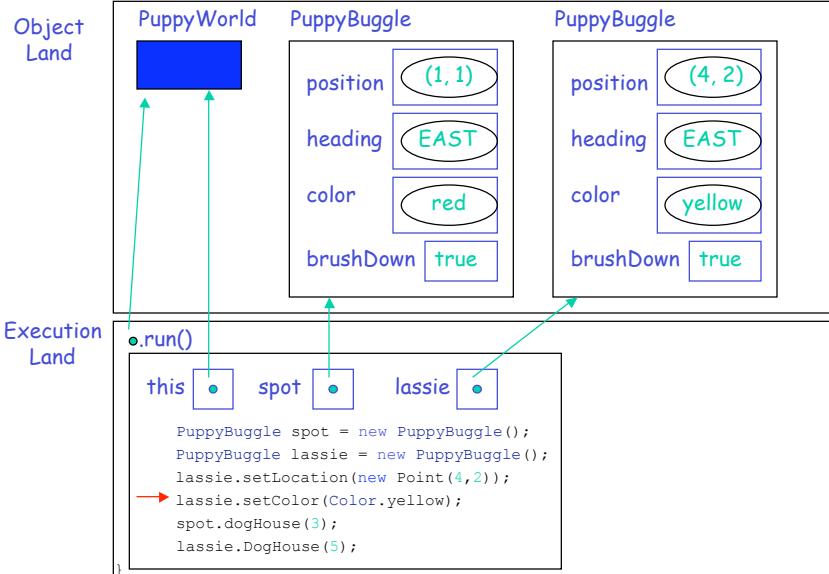
Methods 1 25

Lassie come home



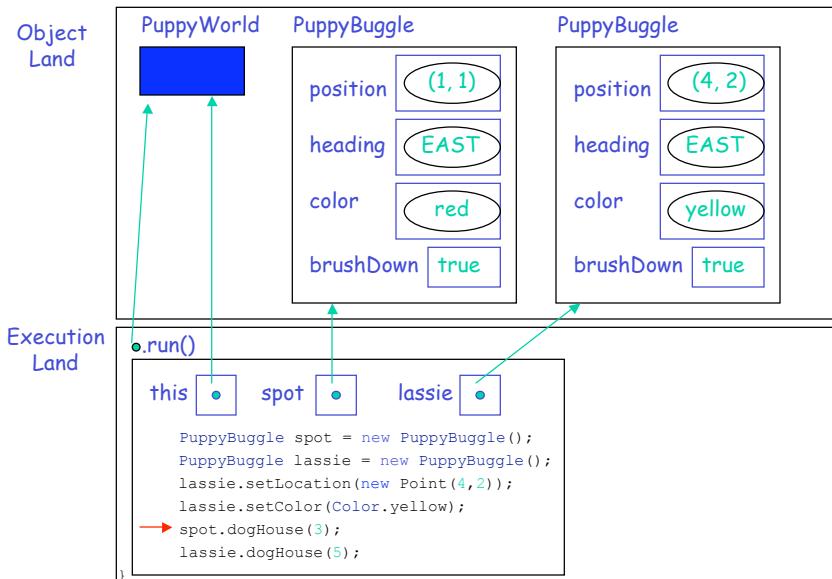
Methods 1 26

A new color for lassie



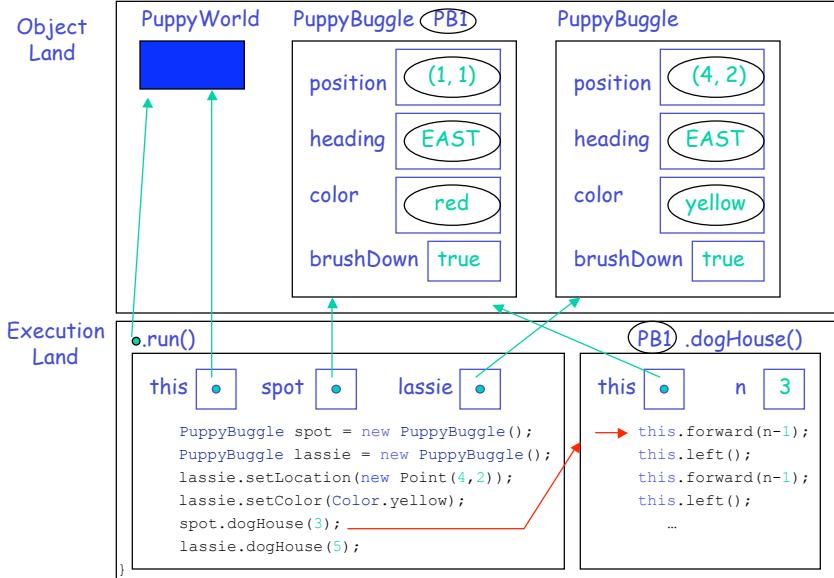
Methods 1 27

Spot is sent to the dogHouse



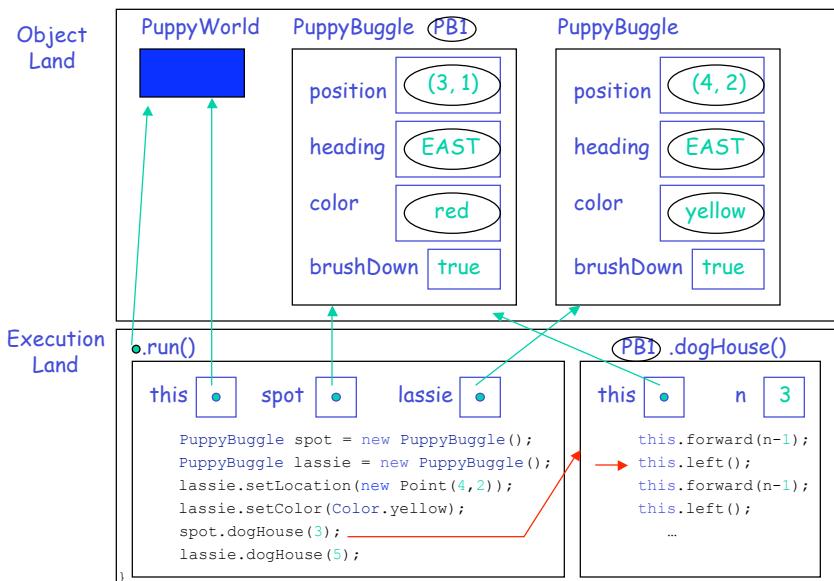
Methods 1 28

New a frame is created for dogHouse ()



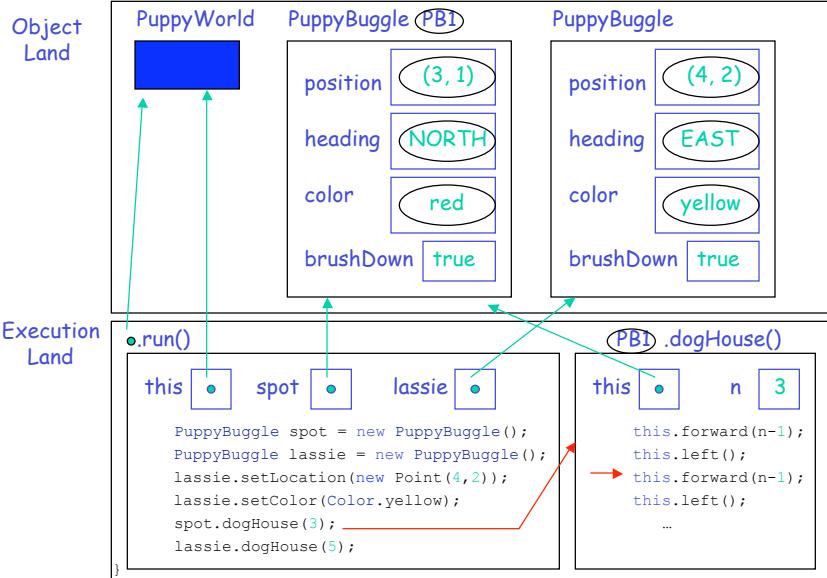
Methods 1 29

The value of n-1 is calculated forward(2) is executed



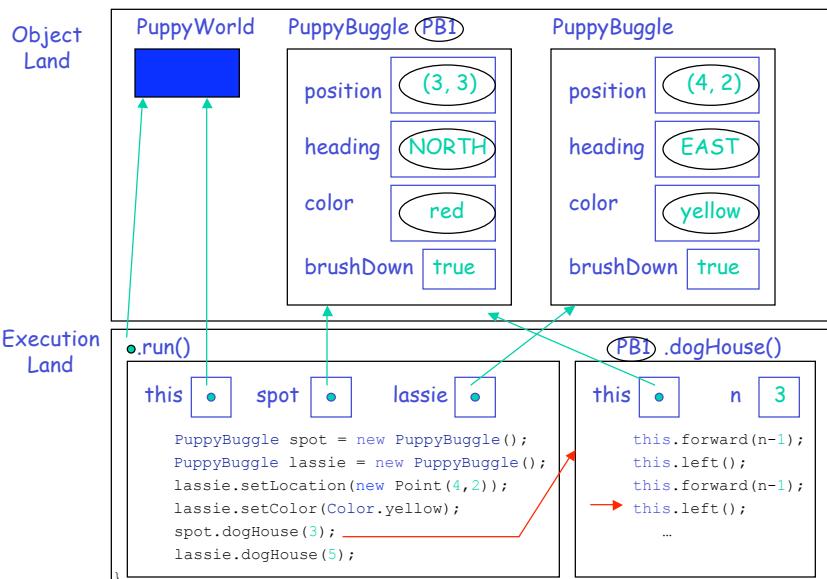
Methods 1 30

Spot turns left



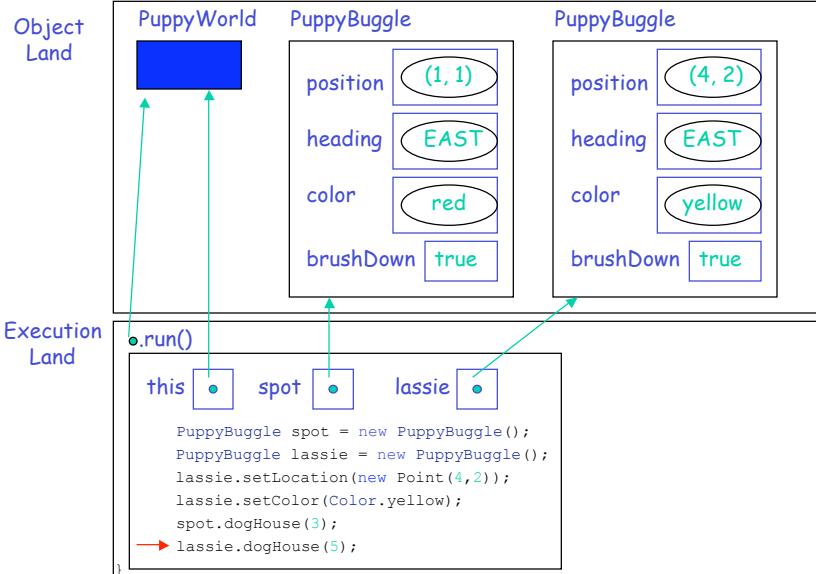
Methods 1 31

Spot and heads NORTH



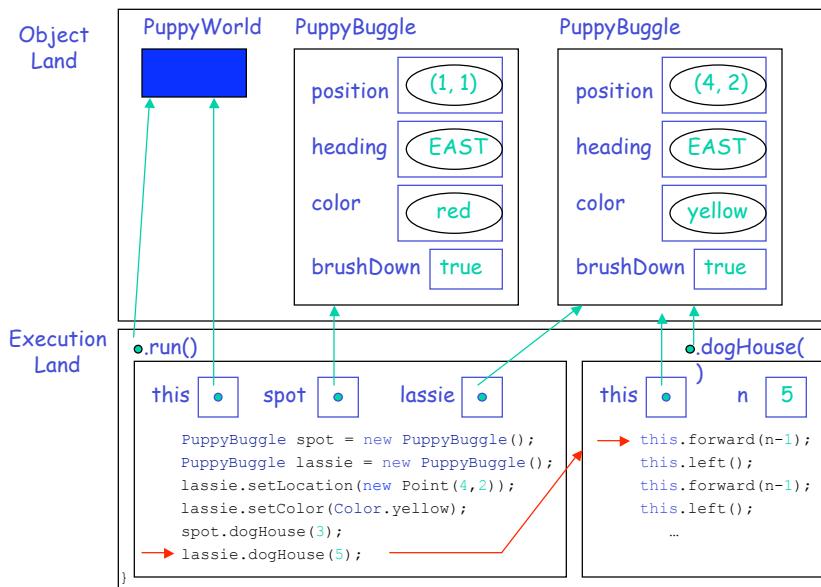
Methods 1 32

After a while spot.dogHouse(3) finishes



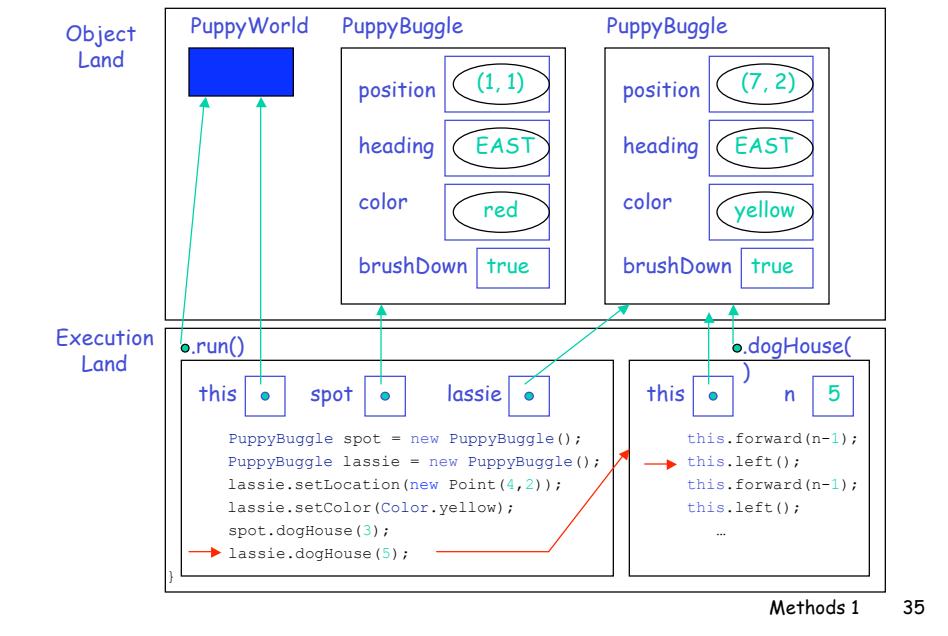
Methods 1 33

And lassie.dogHouse(3) begins

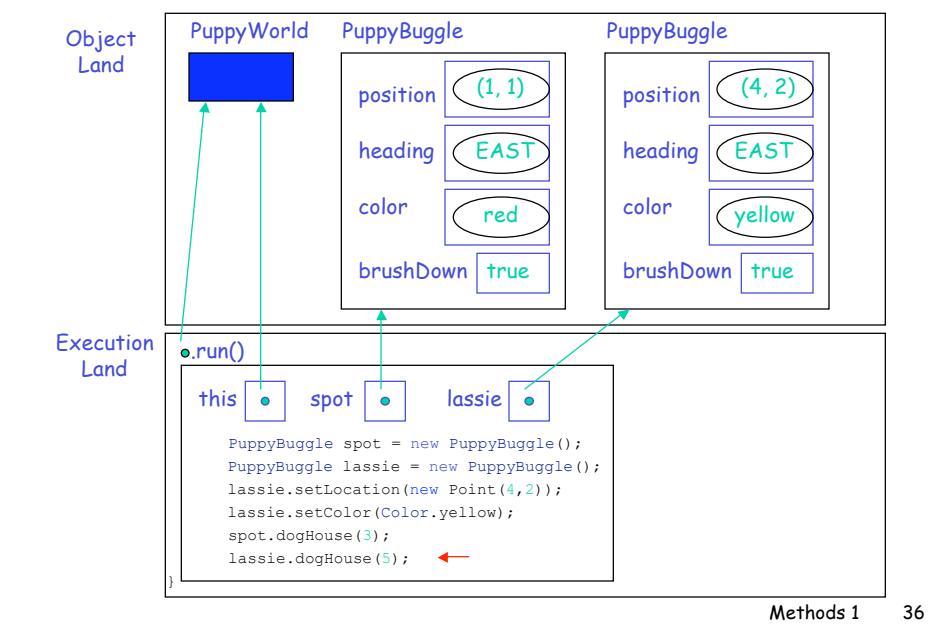


Methods 1 34

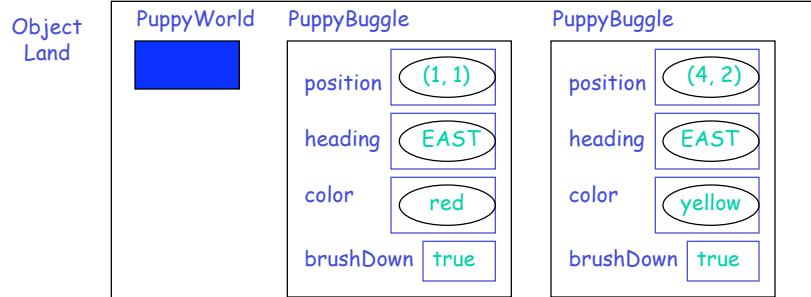
The show goes on ...



Until lassie finishes her dogHouse



run() finishes and goes away too



Methods 1 37

Eventually the garbage collector makes her rounds



Methods 1 38

By the way, we could keep both versions*

```
public class PuppyWorld extends BuggleWorld
{
    public void run()
    {
        PuppyBuggle spot = new PuppyBuggle();
        PuppyBuggle lassie = new PuppyBuggle();
        lassie.setLocation(new Point(4,2));
        lassie.setColor(Color.yellow);
        spot.dogHouse();
        lassie.dogHouse(5);
    }
}
class PuppyBuggle extends Buggle
{
    public void dogHouse() {...}          *This is known as overloading.
    public void dogHouse(int n) {...}
}
```

Methods 1 39