Learning New Words - Methods

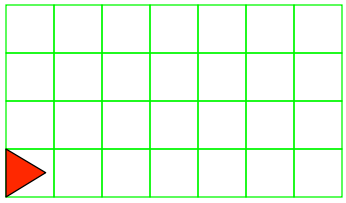
Tuesday, September 11, 2007

CS111 Computer Programming

Department of Computer Science
Wellesley College

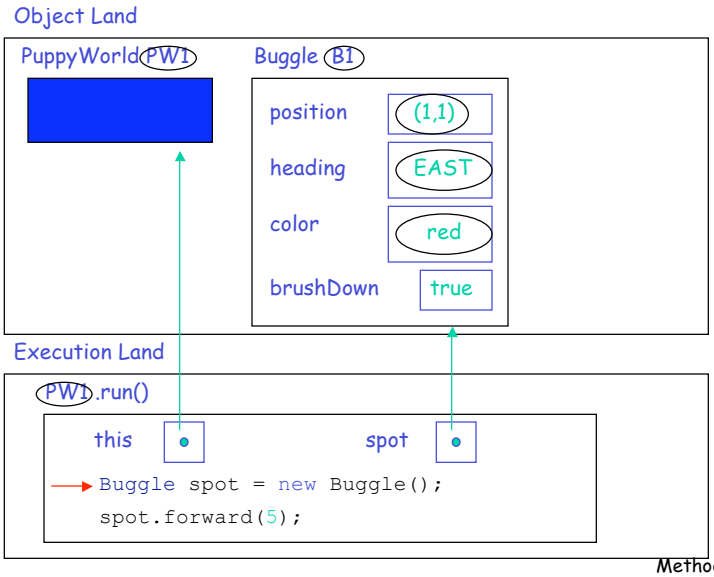
Inventing new worlds*

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



* `BuggleWorld` is itself a class.

A JEM Snapshot

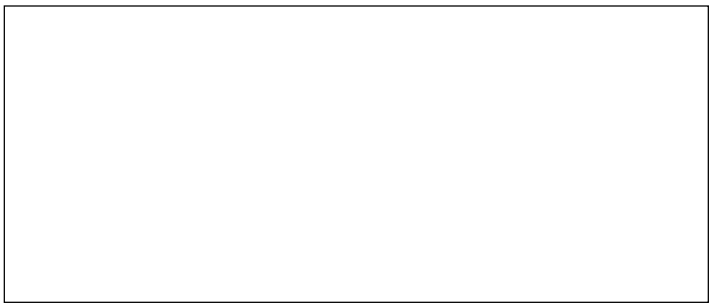


The Java Execution Model (JEM)

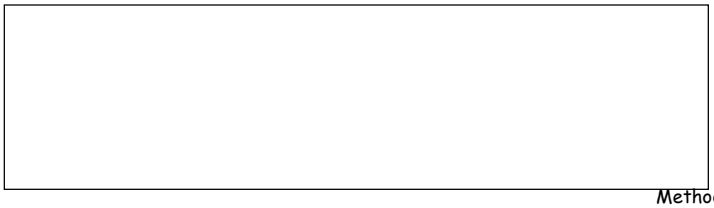
- The Java Execution Model (JEM) is a way to visualize what is going on when a piece of code executes.
- The JEM is divided into two regions: Object Land, where objects live, and Execution Land, where the code is executed.
- Execution Land displays the evolution of a piece of code as it executes. It is inhabited by execution frames, one for each method that is currently running. Every execution frame has two parts: a place for variables used in the method and the sequence of statements that make up the body of the method.

In the beginning

Object Land

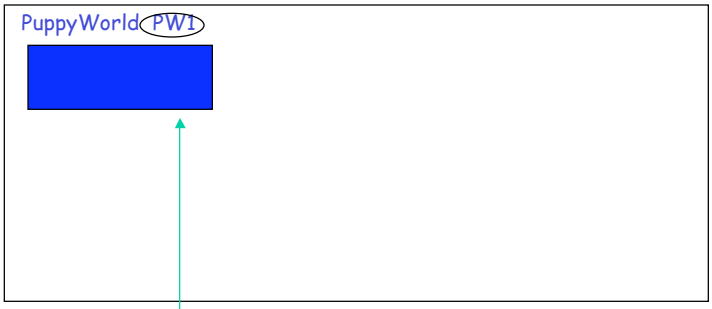


Execution Land

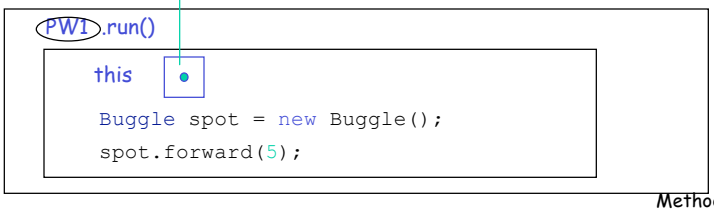


The `run()` method is invoked

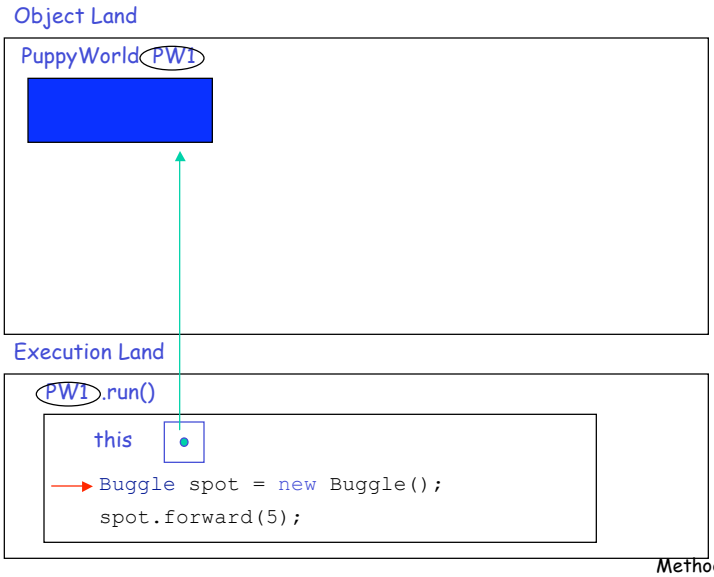
Object Land



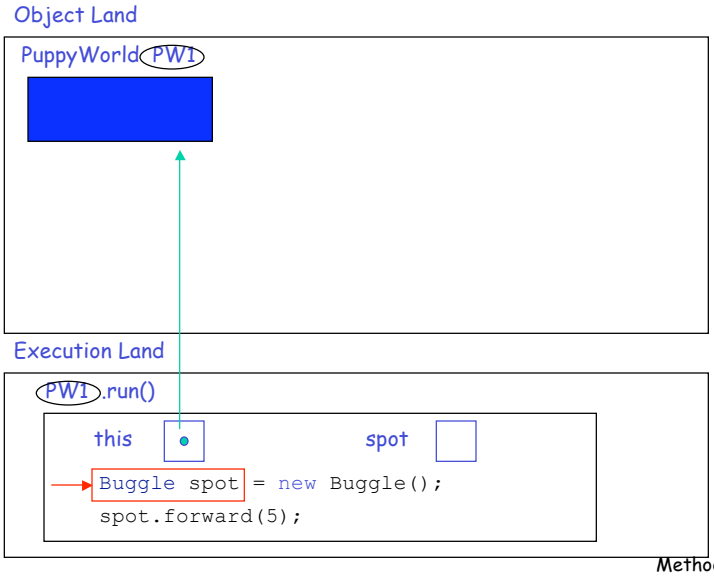
Execution Land



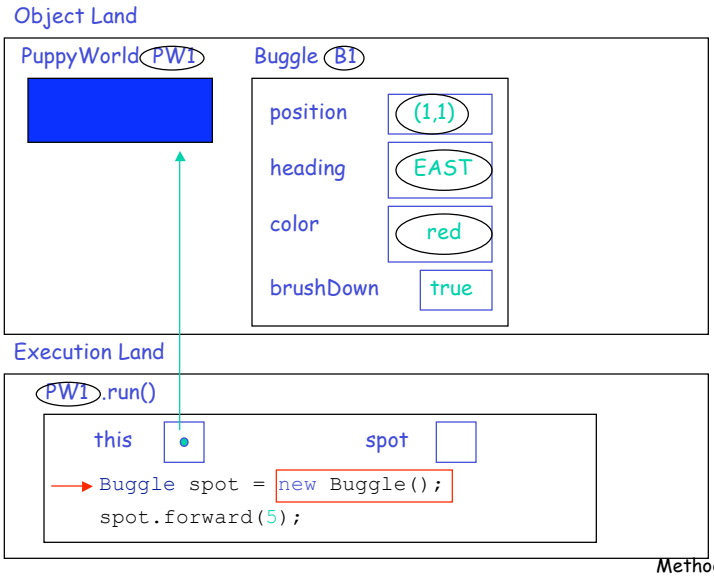
Execution of `run()` begins



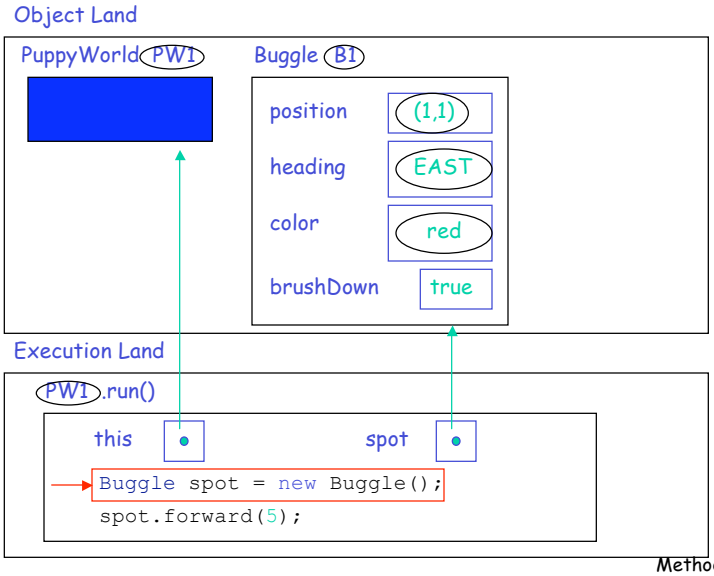
A local variable is created



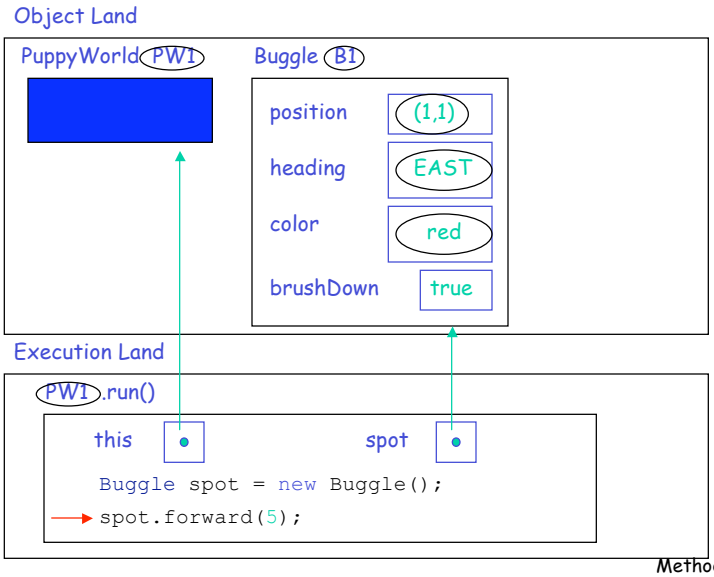
A new Buggle object is constructed



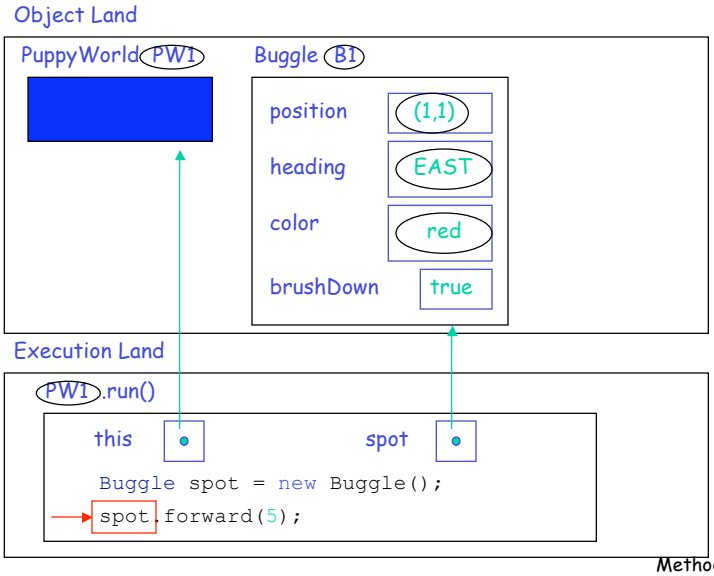
The assignment statement executes



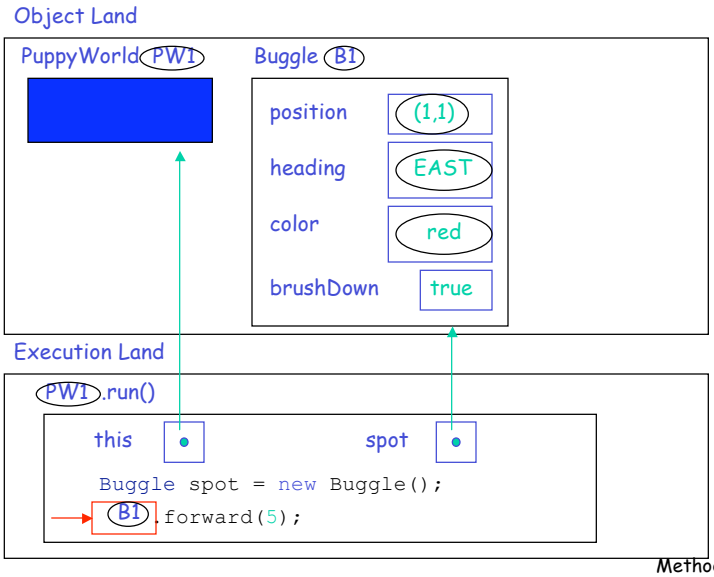
We move on



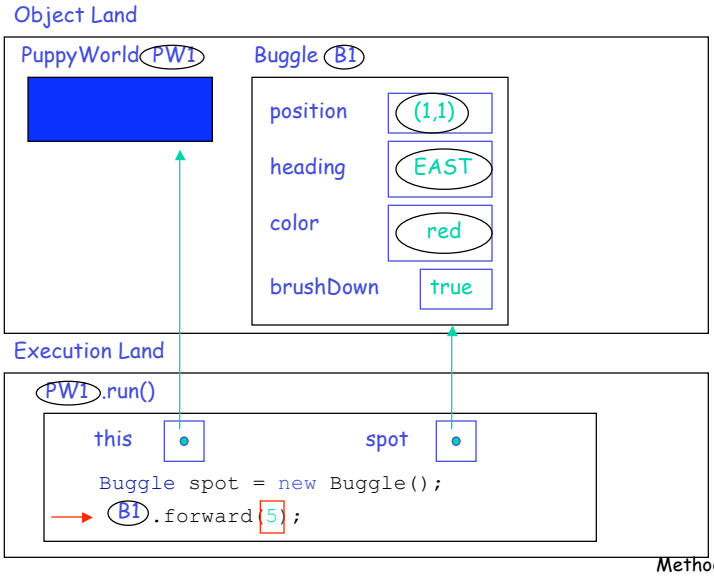
spot is evaluated



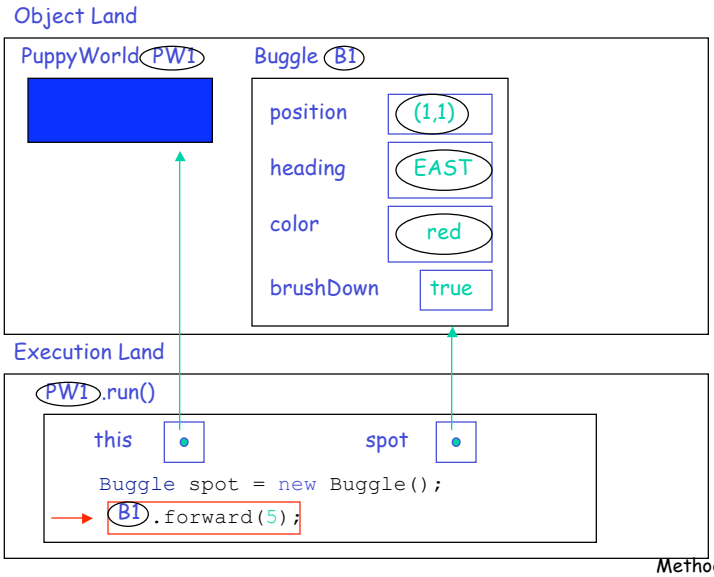
spot references the Buggle object B1



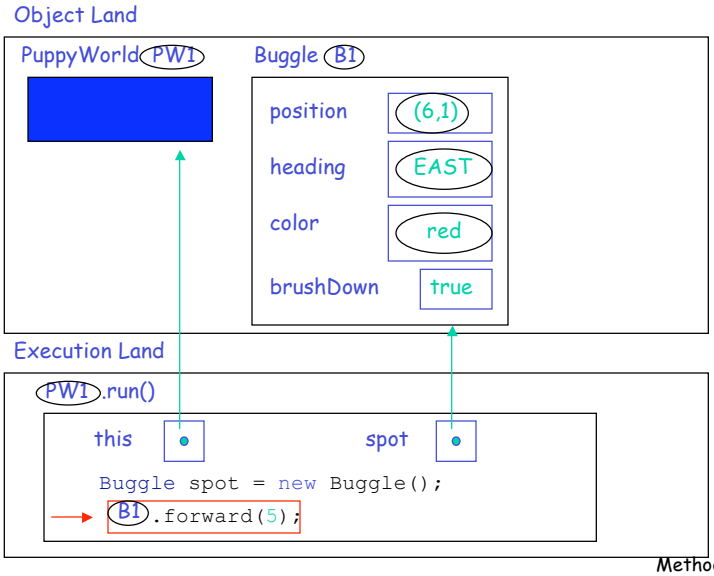
The argument is evaluated



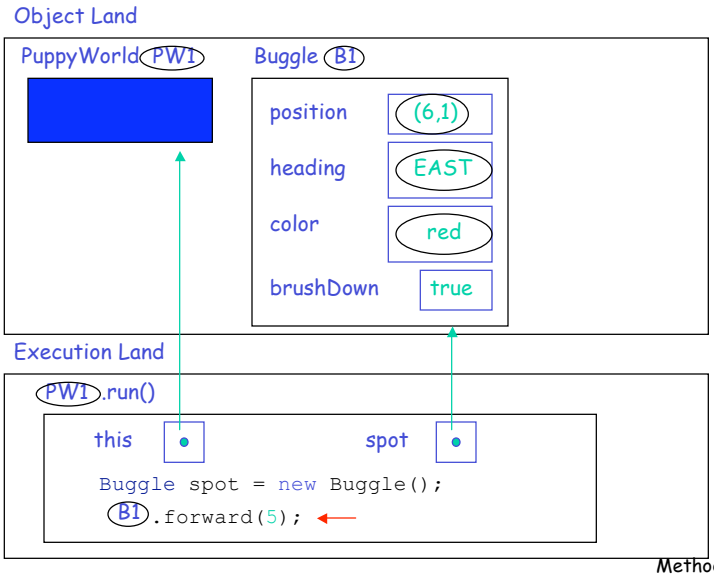
The `forward()` message is sent to B1



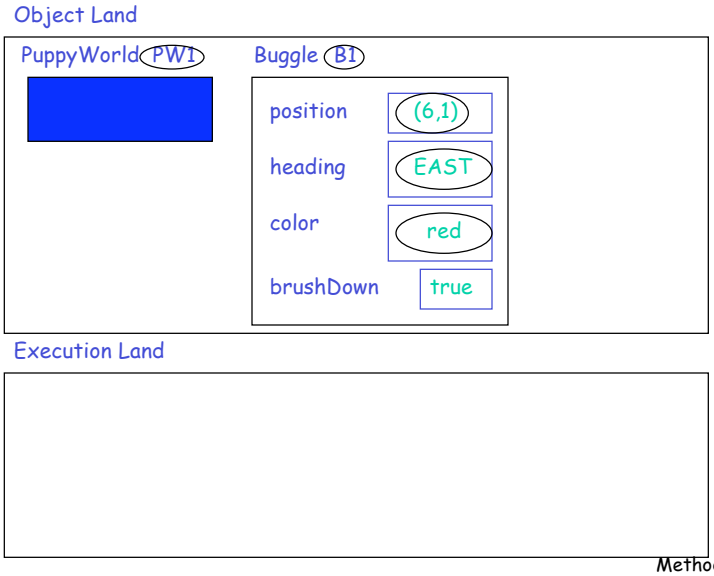
spot runs away



run() finishes execution

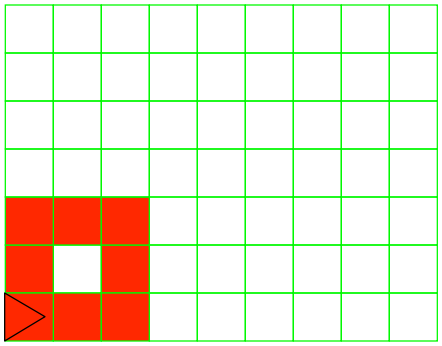


run() finishes execution



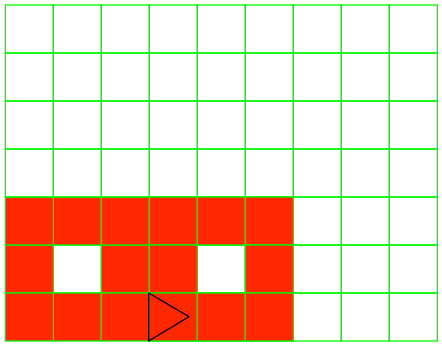
Spot builds a home

```
public class PuppyWorld extends BuggleWorld {
     public void run() {
          Buggle spot = new Buggle();
          // spot builds a doghouse
          spot.forward(2);
          spot.left();
          spot.forward(2);
          spot.left();
          spot.forward(2);
          spot.left();
          spot.forward(2);
          spot.left();
     }
}
```



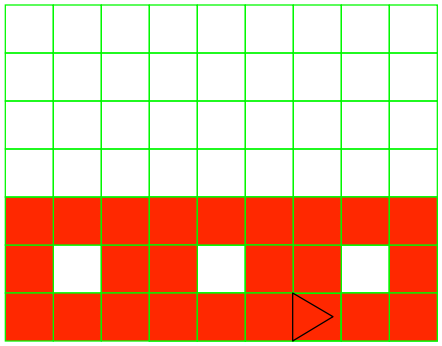
Spot builds a second home

```
public class PuppyWorld extends BuggleWorld {
    public void run() {
       Buggle spot = new Buggle();
       // spot builds a primary residence
       spot.forward(2);
       spot.left();
       spot.forward(2);
       spot.left();
       spot.forward(2);
       spot.left();
       spot.forward(2);
       spot.left();
       // spot builds a vacation home
       spot.forward(3);
       spot.forward(2);
       spot.left();
       spot.forward(2);
       spot.left();
       spot.forward(2);
       spot.left();
       spot.forward(2);
       spot.left();
    }
}
```



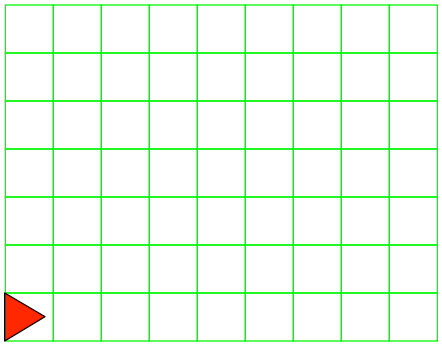
Spot gets greedy

```
public class PuppyWorld extends BuggleWorld {
    public void run() {
       Buggle spot = new Buggle();
       // spot builds a primary residence
        spot.forward(2);
        spot.left();
        spot.forward(2);
        spot.left();
        spot.forward(2);
        spot.left();
        spot.forward(2);
        spot.left();
        // spot builds a vacation home
        spot.forward(3);
        spot.forward(2);
        spot.left();
        spot.forward(2);
        spot.left();
        spot.forward(2);
        spot.left();
        spot.forward(2);
        spot.left();
        // spot builds a third home
        spot.forward(3);
        spot.forward(2);
        spot.left();
        spot.forward(2);
        spot.left();
        spot.forward(2);
        spot.left();
        spot.forward(2);
        spot.left();
    }
}
```



We would like to say . . . *

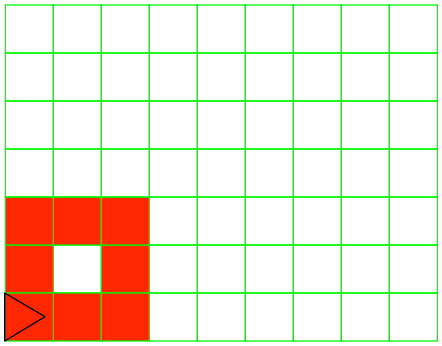
```
public class PuppyWorld extends BuggleWorld {
   public void run() {
      Buggle spot = new Buggle();
      spot.buildDogHouse();
      spot.forward(3);
      spot.buildDogHouse();
      spot.forward(3);
      spot.buildDogHouse();
   }
}
```



* But poor spot would not obey. Why?

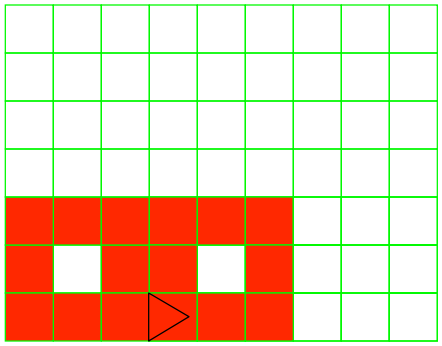
A new breed of Buggle

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

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



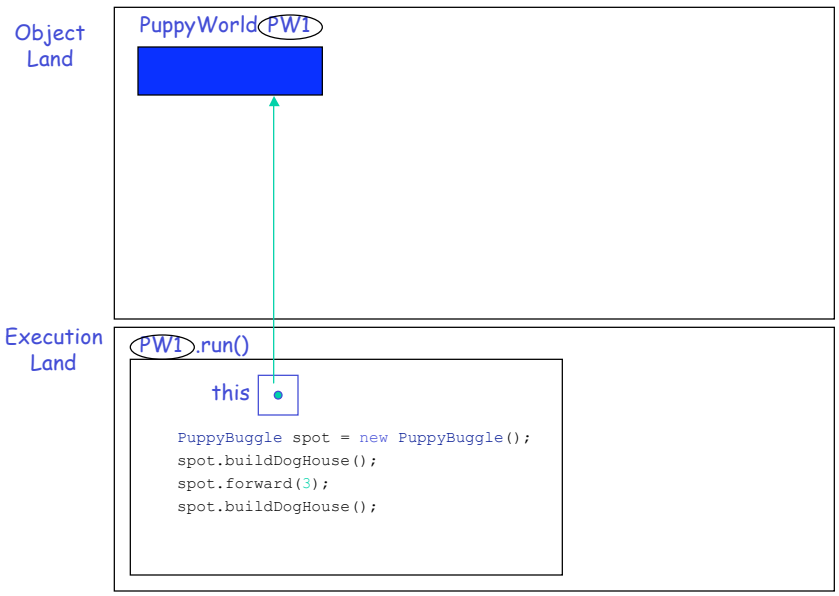
A new breed of Buggle

```
public class PuppyWorld extends BuggleWorld {
      public void run() {
            PuppyBuggle spot = new PuppyBuggle();
            spot.buildDogHouse(); // build 1st house
            spot.forward(3);
            spot.buildDogHouse(); // build 2nd house
      }
}
class PuppyBuggle extends Buggle {
      public void buildDogHouse() {
            this.forward(2);
            this.left();
            this.forward(2);
            this.left();
            this.forward(2);
            this.left();
            this.forward(2);
            this.left();
      }
}
```

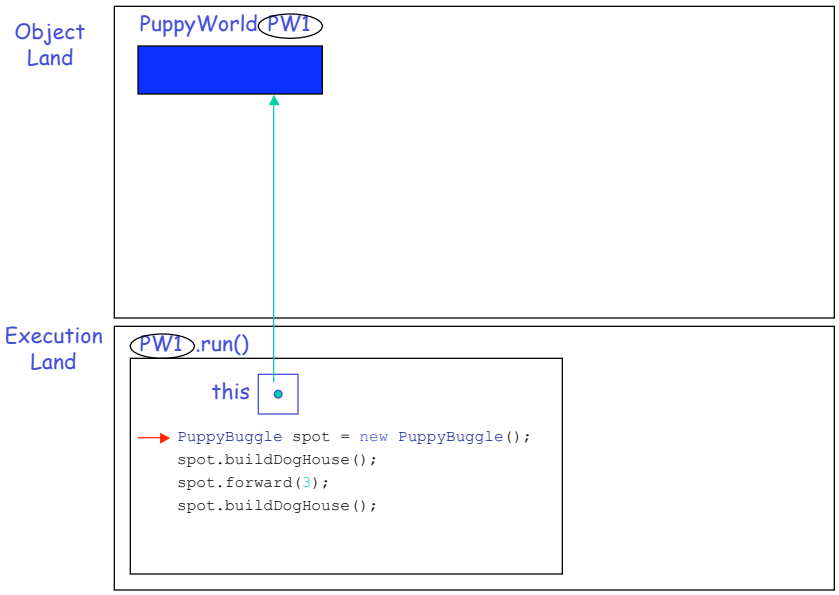


JEMs tracing spot building doghouses

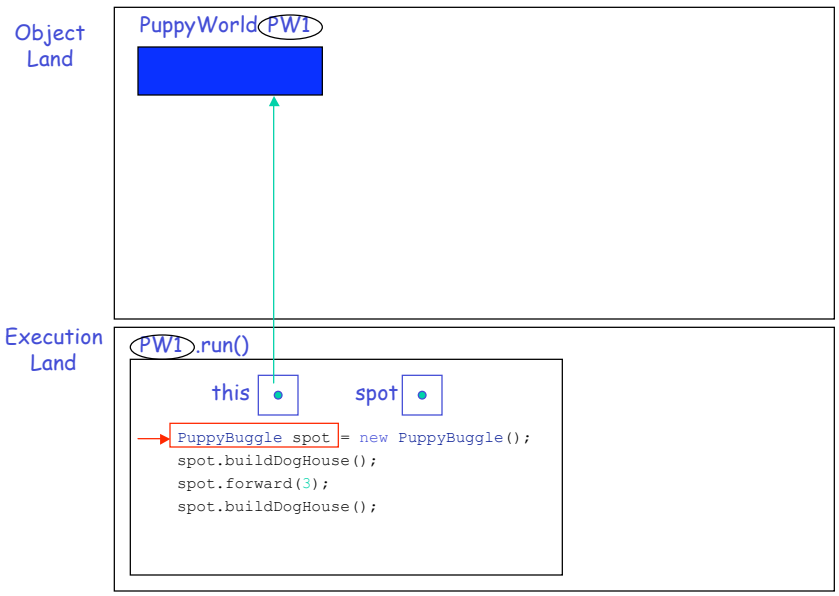
`run()` method is invoked



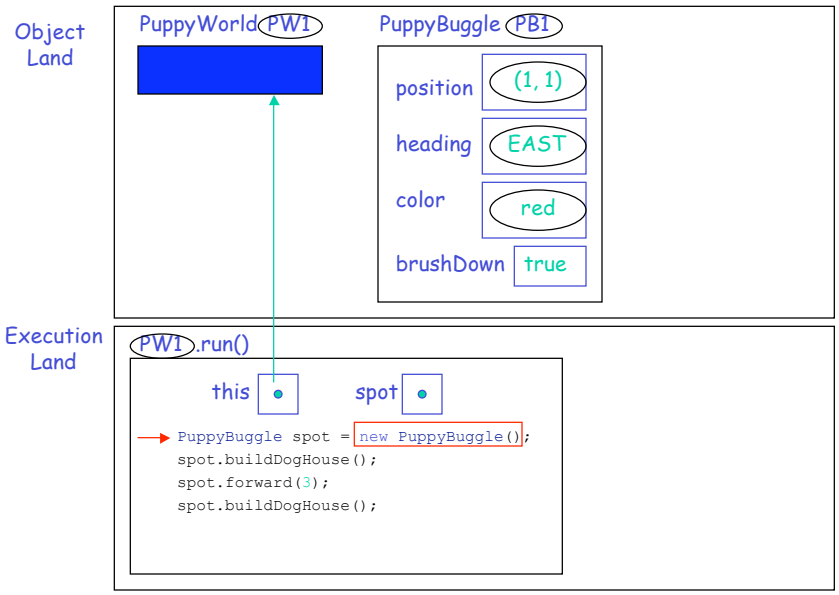
`run()` begins to run



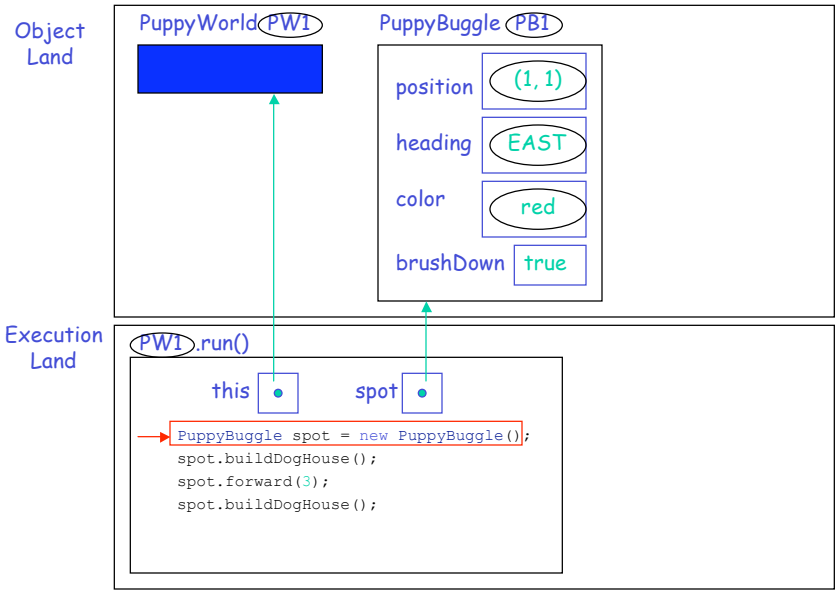
A local variable is created



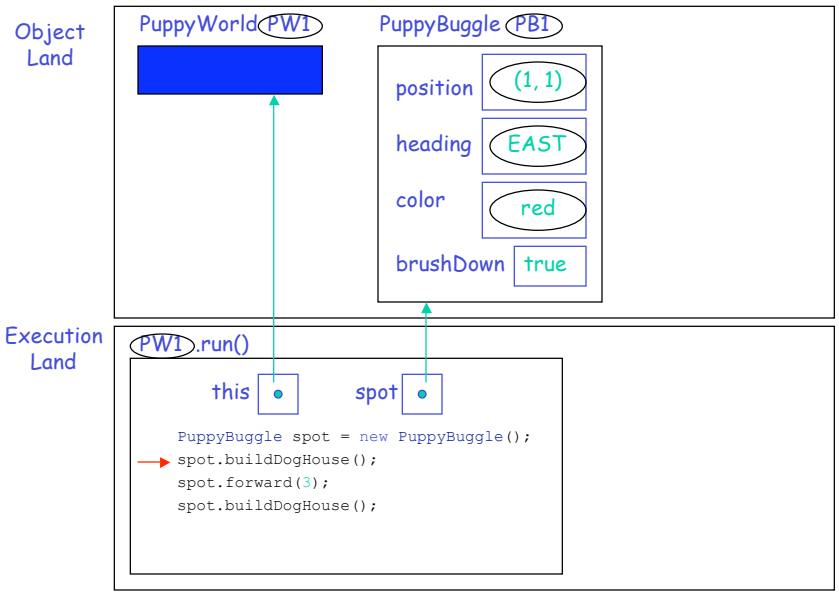
Construct new PuppyBuggle object



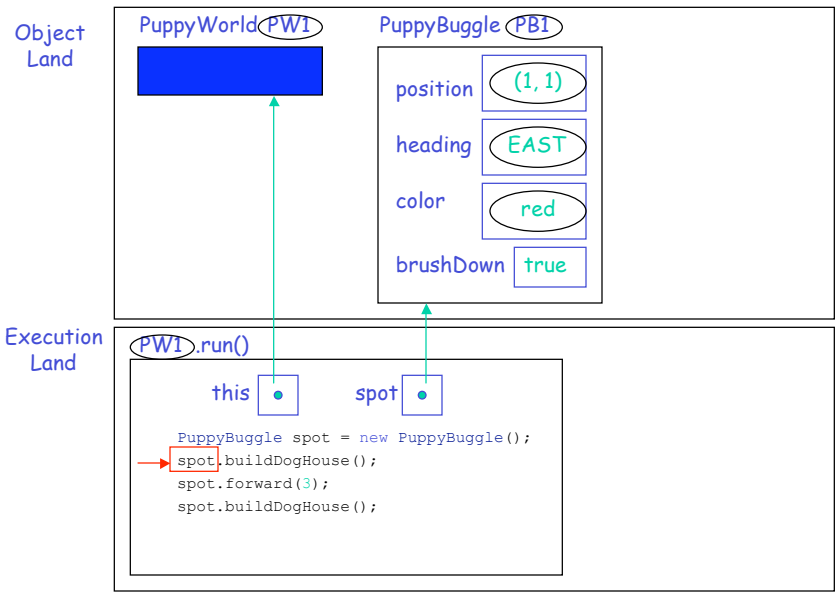
Assignment statement executes



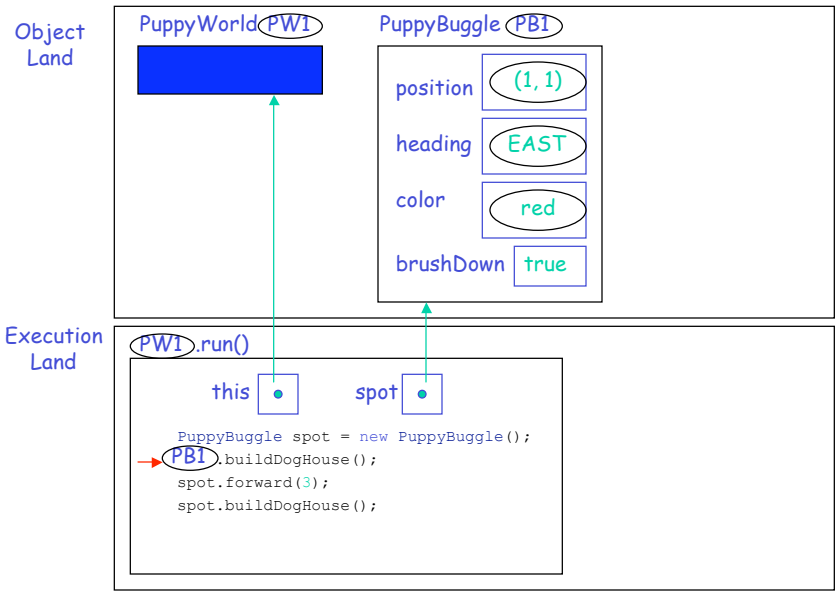
Move to the next statement



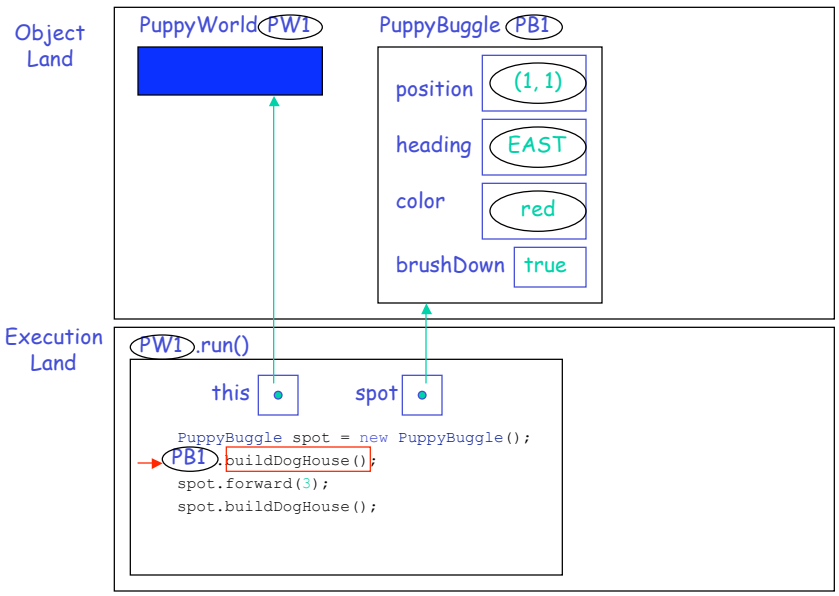
spot is evaluated



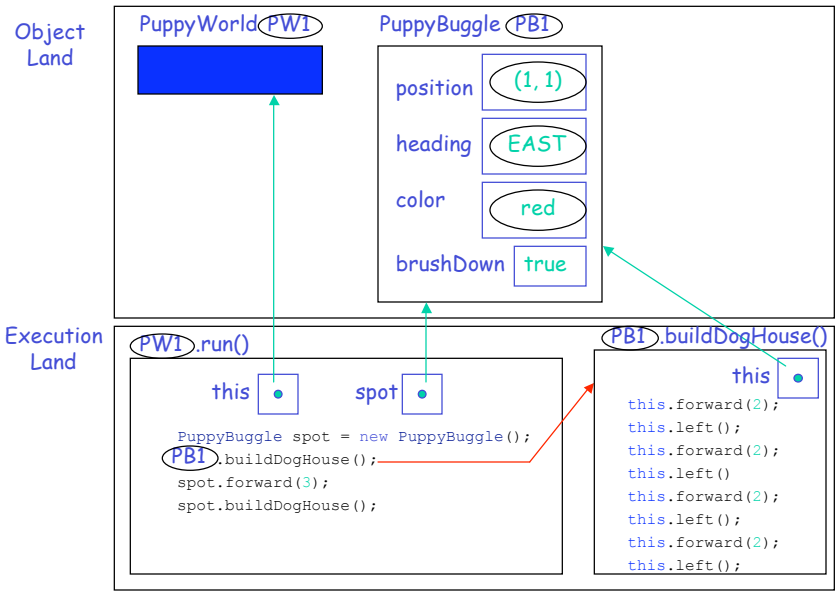
spot is evaluated



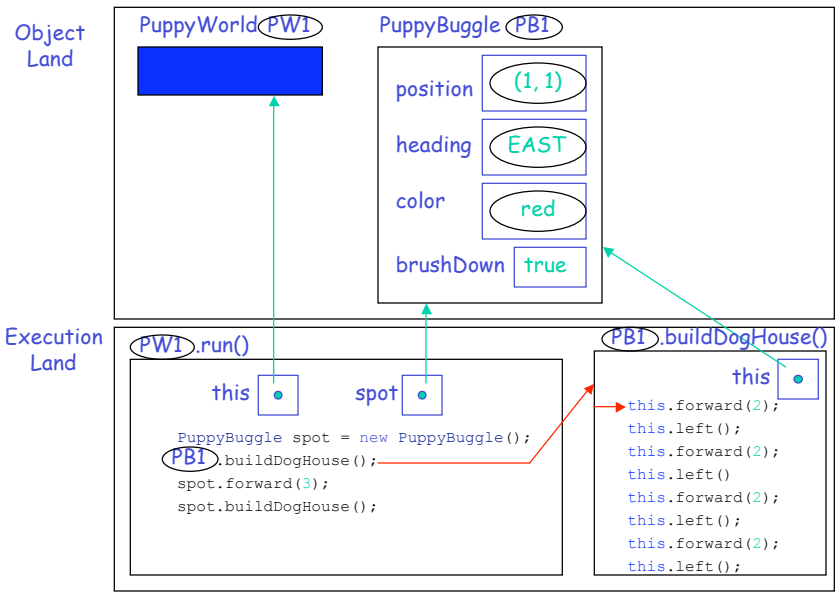
The `buildDogHouse()` method is sent to PB1



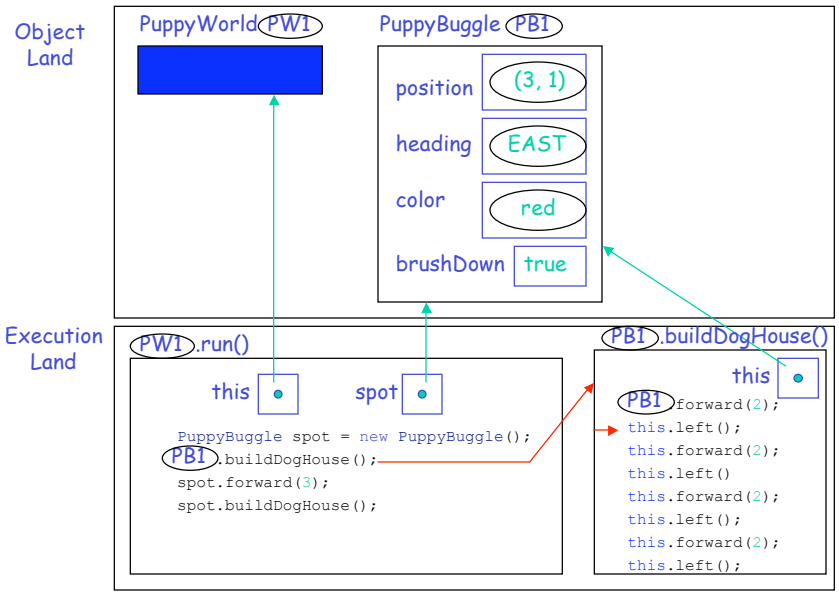
New frame is created for `buildDogHouse()`



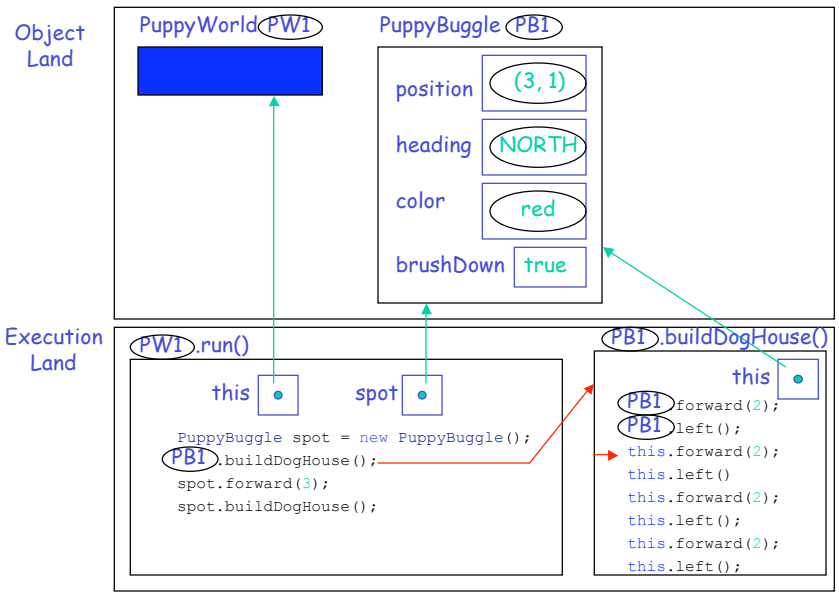
`buildDogHouse()` begins to run



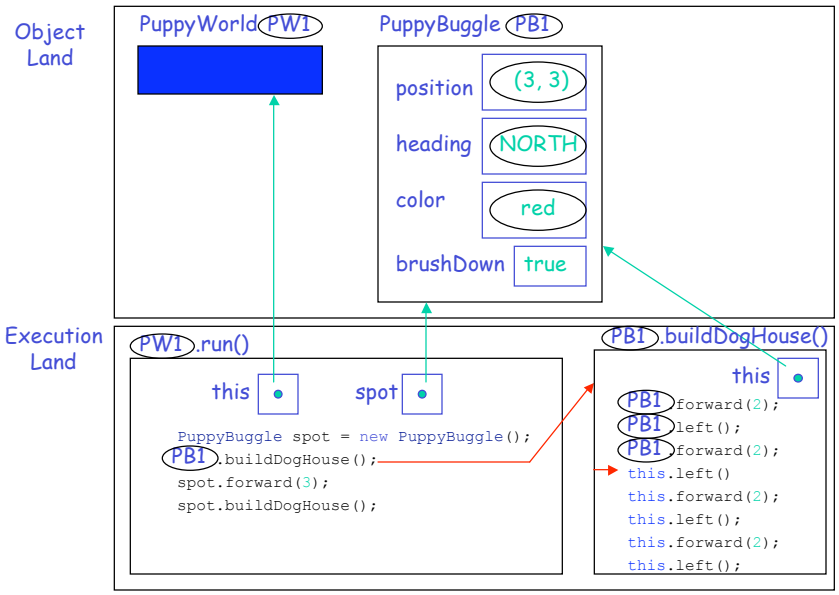
this PuppyBuggle moves forward



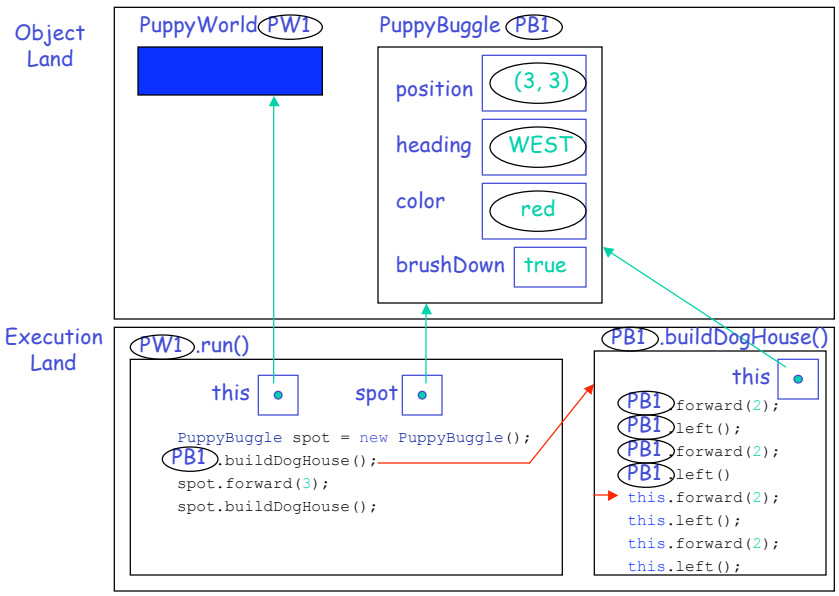
this PuppyBuggle turns left



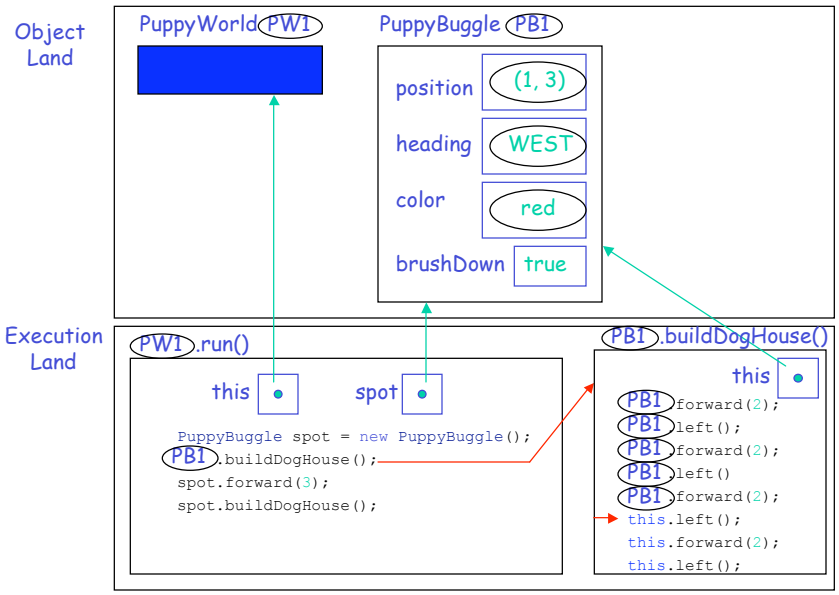
this PuppyBuggle moves forward



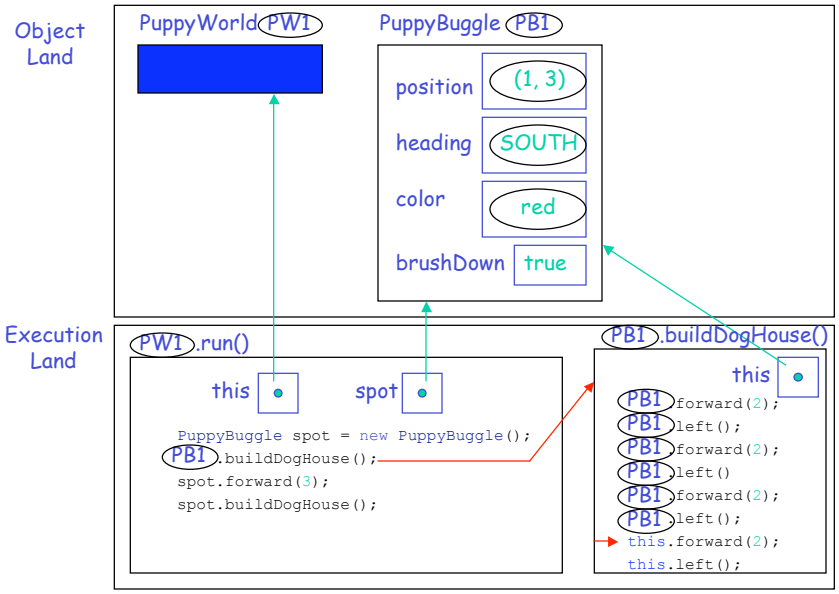
this PuppyBuggle turns left



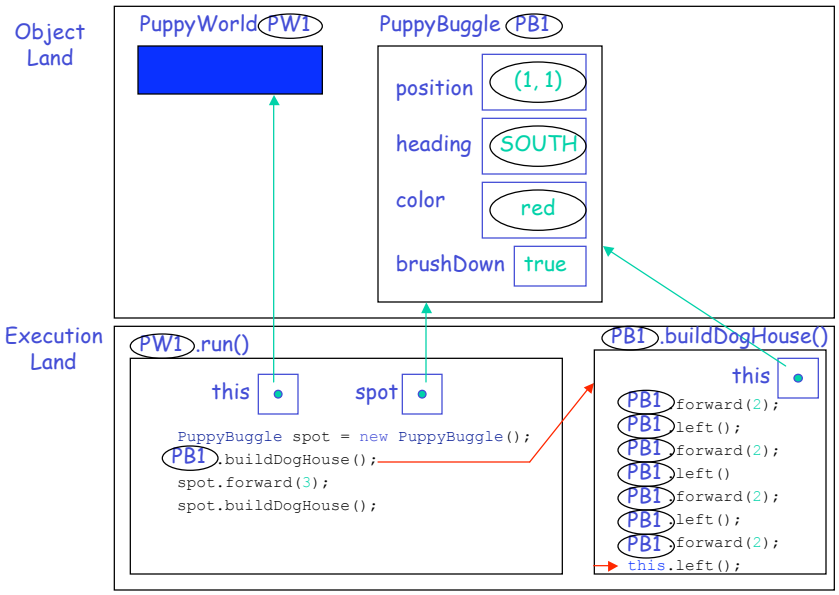
this PuppyBuggle moves forward



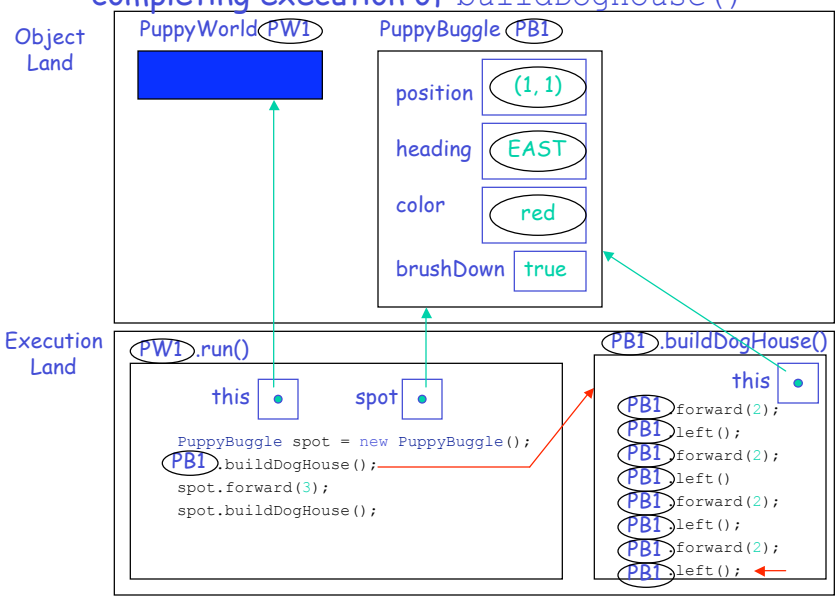
this PuppyBuggle turns left



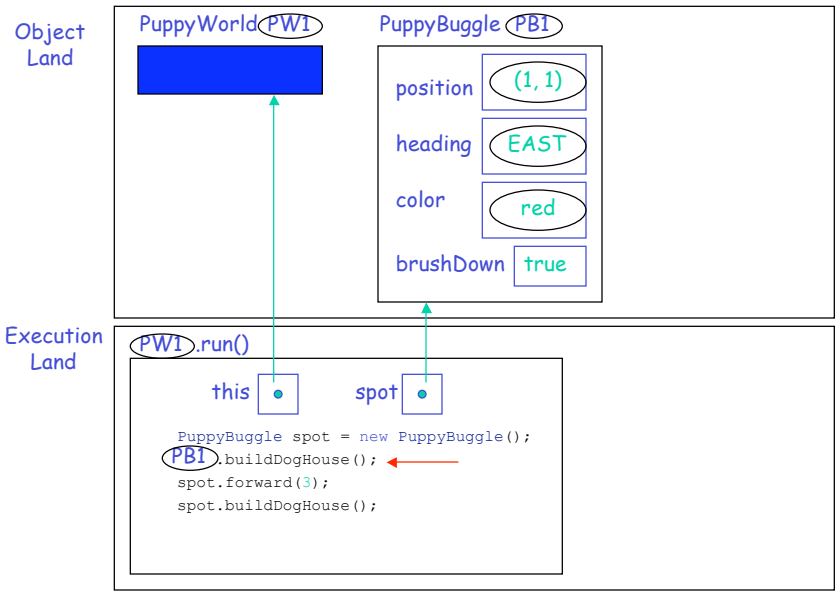
`this` PuppyBuggle moves forward



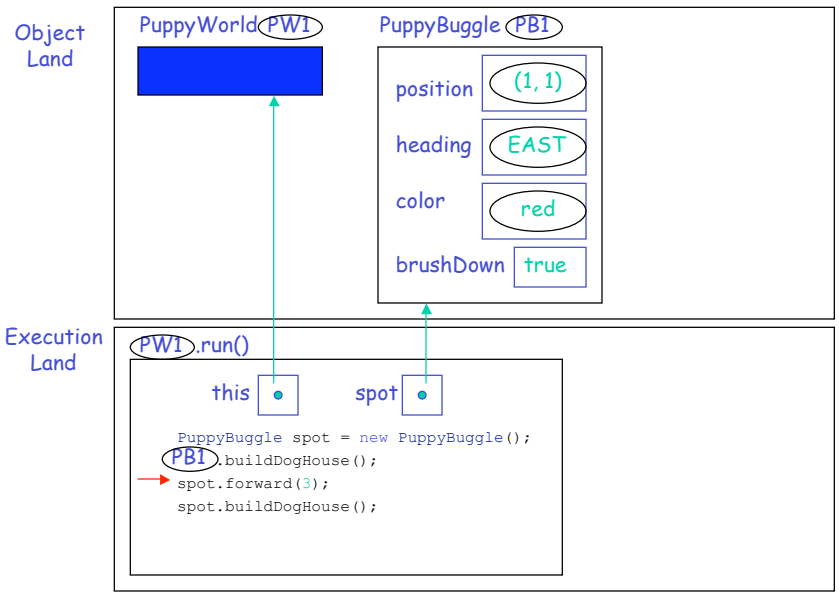
`this` PuppyBuggle turns left, completing execution of `buildDogHouse()`



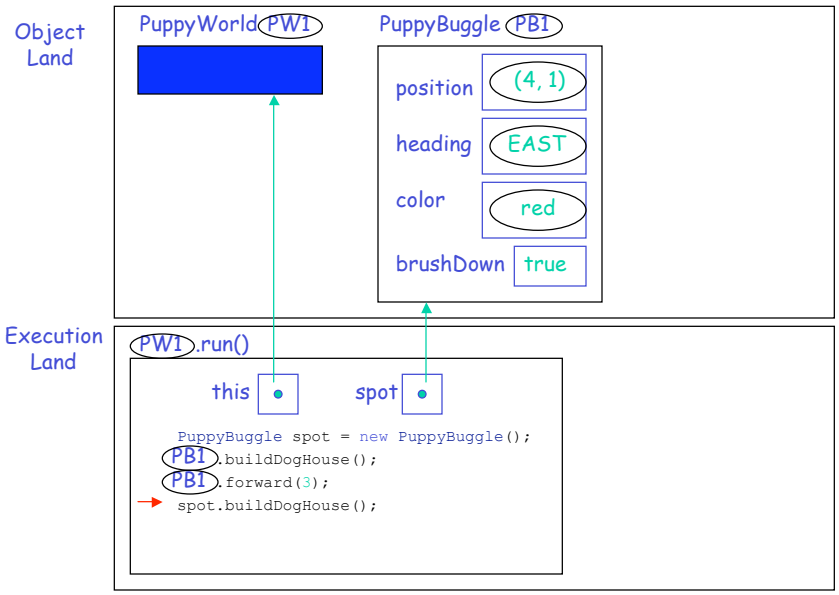
Execution frame closes



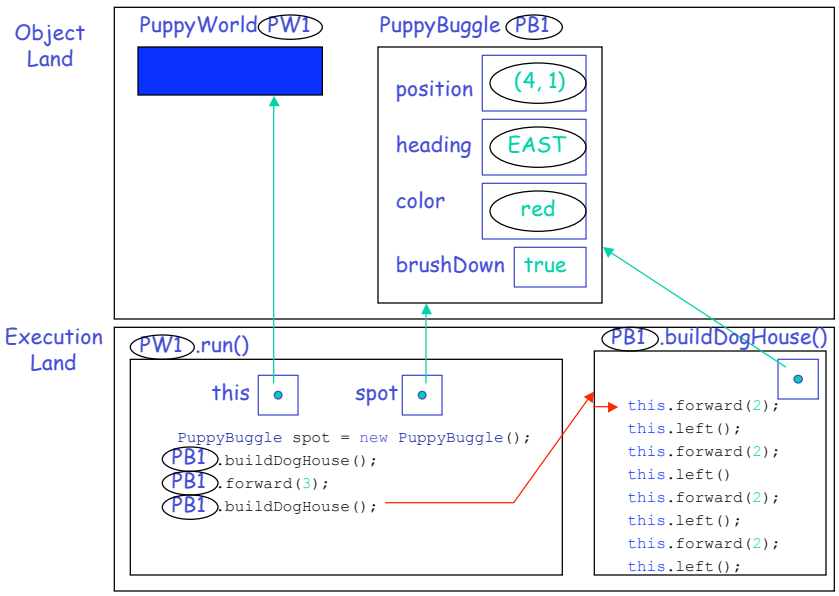
Move to the next instruction



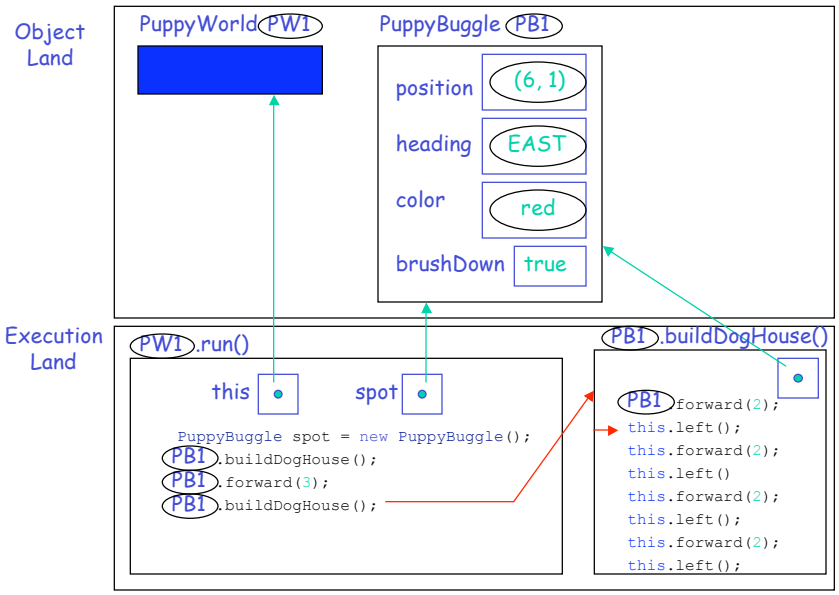
spot moves forward



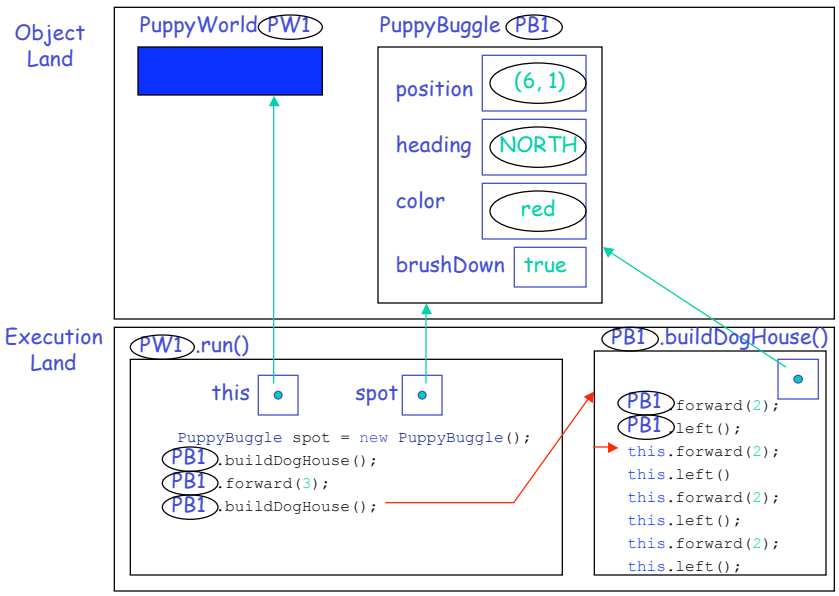
spot builds another doghouse



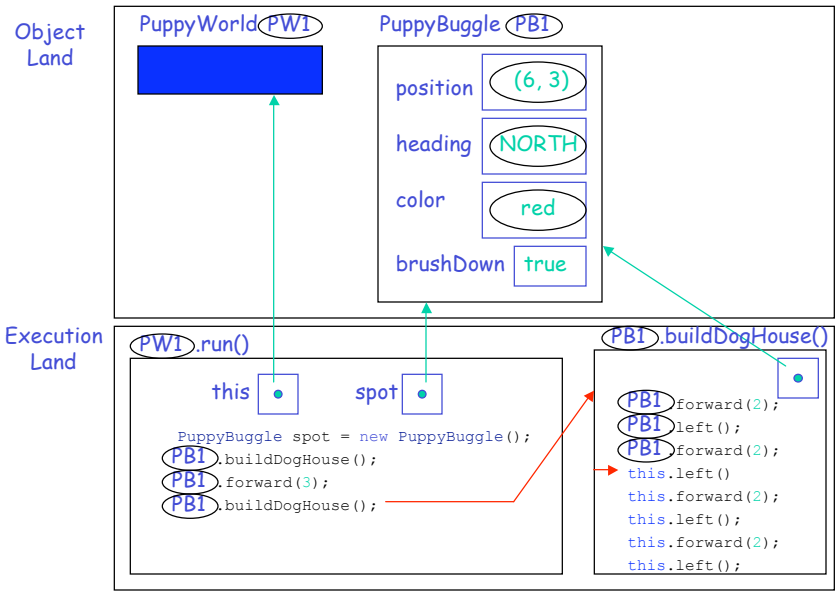
this PuppyBuggle moves forward



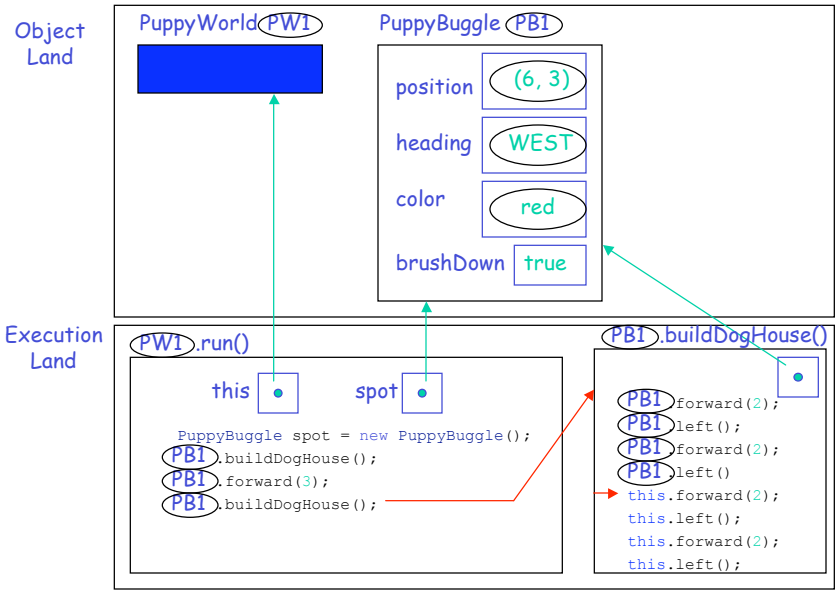
this PuppyBuggle turns left



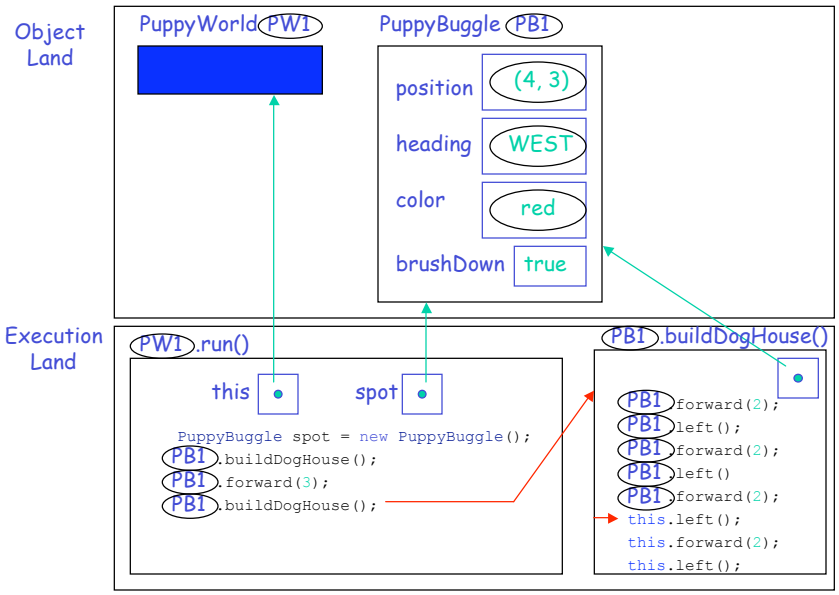
this PuppyBuggle moves forward



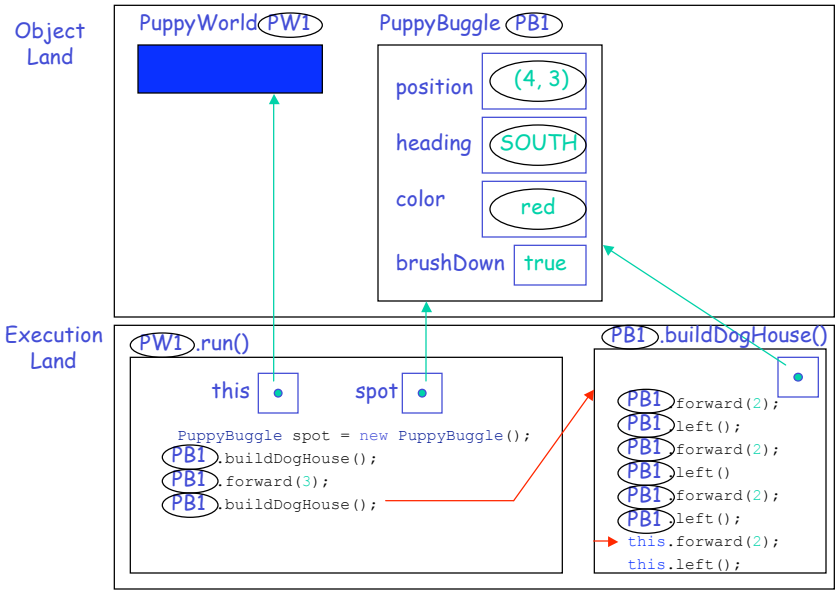
this PuppyBuggle turns left



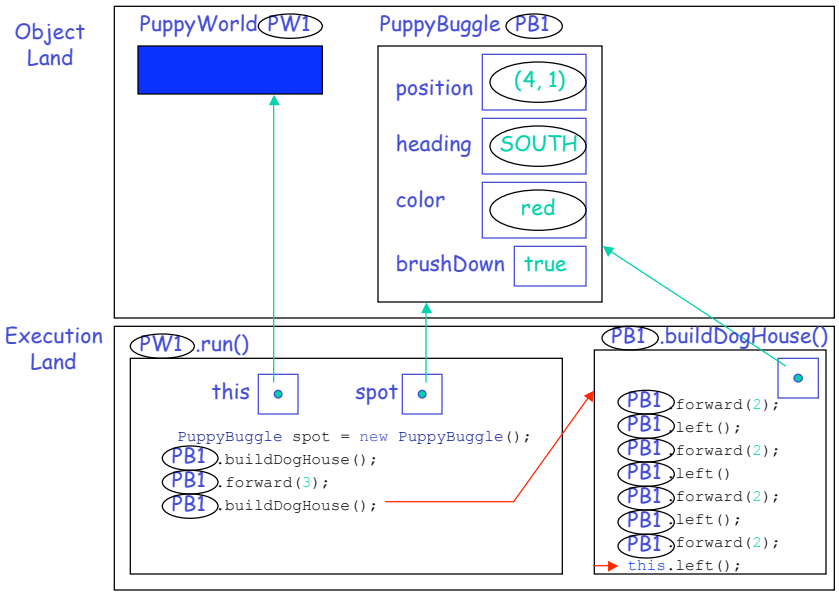
this PuppyBuggle moves forward



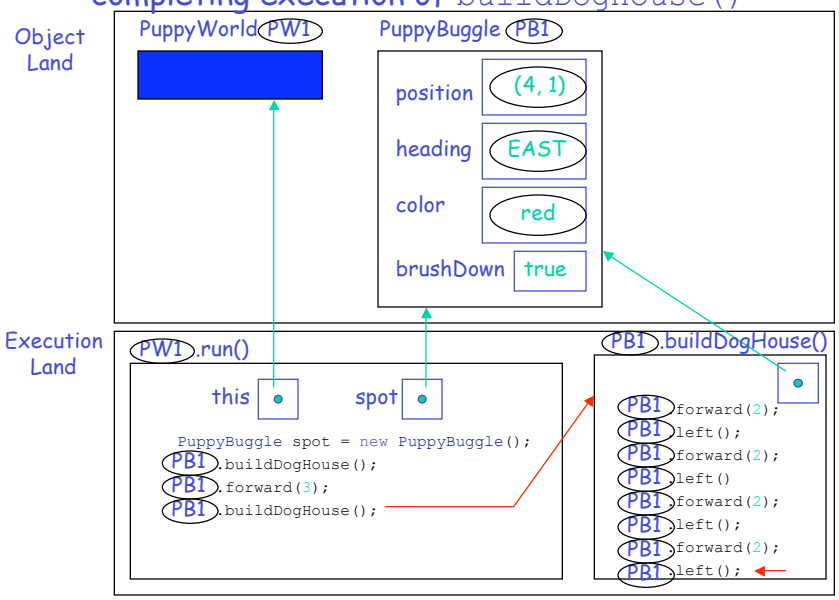
this PuppyBuggle turns left



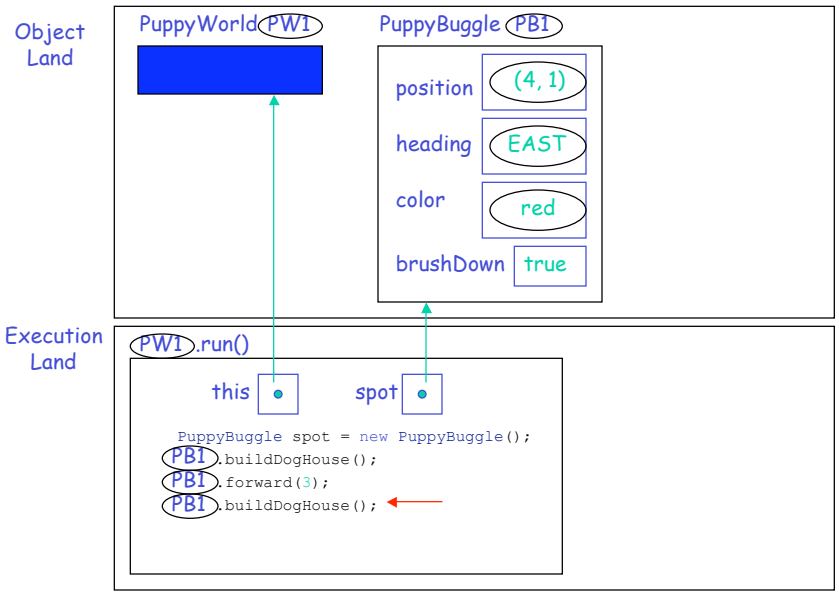
`this` PuppyBuggle moves forward



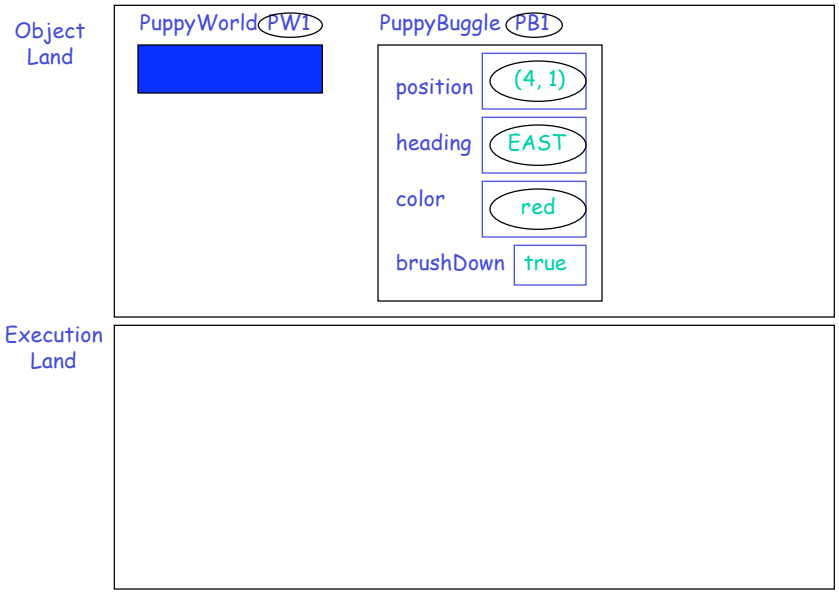
`this` PuppyBuggle turns left, completing execution of `buildDogHouse()`



`buildDogHouse()` execution frame closes



`run()` execution frame closes

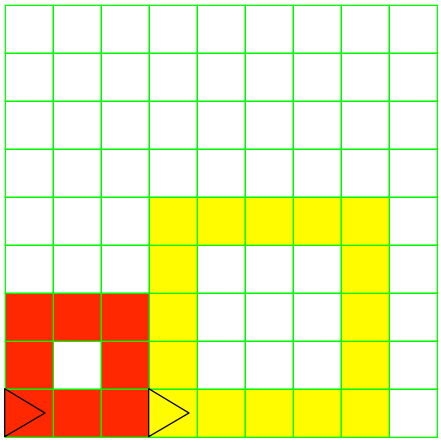


Eventually the garbage collector makes her rounds

Object Land

Execution Land

Lassie wants a dogHouse too*



*But collies are much bigger than terriers.

We could . . .

```
public class PuppyWorld extends BuggleWorld {
     public void run()
     {
          PuppyBuggle spot = new PuppyBuggle();
          PuppyBuggle lassie = new PuppyBuggle();
          lassie.setLocation(new Location(4,1));
          lassie.setColor(Color.yellow);
          spot.buildDogHouse();
          lassie.buildBigDogHouse();
     }
}
class PuppyBuggle extends Buggle {
     public void buildDogHouse() {… }
     public void buildBigDogHouse() {… }
}
```

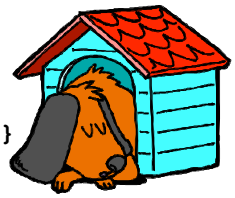
It's déjà vu all over again

```
class PuppyBuggle extends Buggle {

  public void buildDogHouse()
  {
     this.forward(2);
     this.left();
     this.forward(2);
     this.left();
     this.forward(2);
     this.left();
     this.forward(2);
     this.left();
  }

}
```

```
public void buildBigDogHouse()
{
     this.forward(4);
     this.left();
     this.forward(4);
     this.left();
     this.forward(4);
     this.left();
     this.forward(4);
     this.left();
}
```

A solution using parameters

```
public class PuppyWorld extends BuggleWorld {
     public void run()
     {
          PuppyBuggle spot = new PuppyBuggle();
          PuppyBuggle lassie = new PuppyBuggle();
          lassie.setLocation(new Location(4,1));
          lassie.setColor(Color.yellow);
          spot.buildDogHouse(3);
          lassie.buildDogHouse(5);
     }
}
class PuppyBuggle extends Buggle {
     public void buildDogHouse(int n) {… }
}
```



Methods with parameters

```
public class PuppyWorld extends BuggleWorld
{ …
}
class PuppyBuggle extends Buggle {
     public void buildDogHouse(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();
     }
}
```