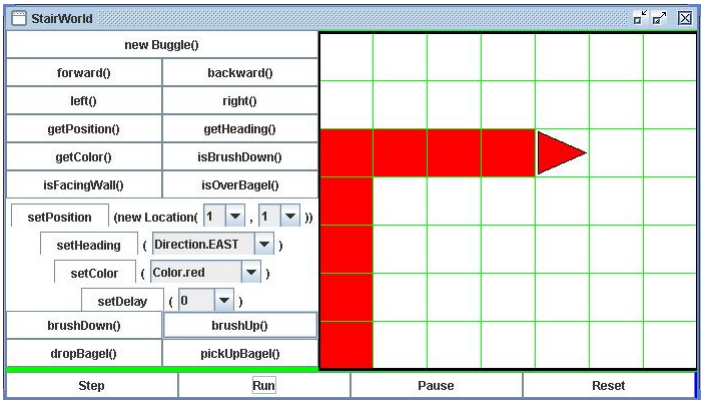
Methods with Parameters and Return Values

Friday, Sep. 14, 2007

CS111 Computer Programming

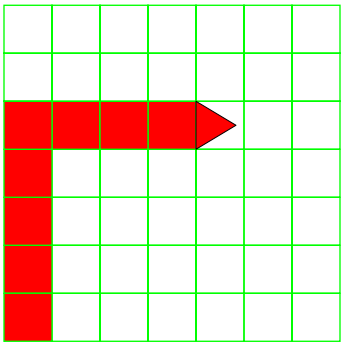
Department of Computer Science
Wellesley College

One small step for Buggles... one giant leap for Bugglekind



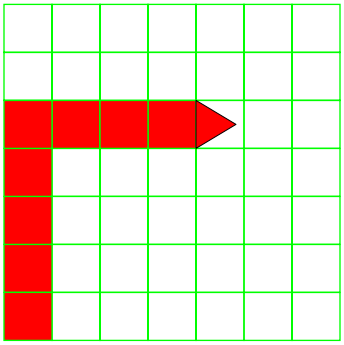
One stair of a fixed size

```
public class StairWorld extends BuggleWorld {
     ...

     public void run() {
          StairBuggle stepper = new StairBuggle();
          stepper.buildStair();
     }
}


class StairBuggle extends Buggle {
     public void buildStair() {
          this.left();
          this.forward(4);
          this.right();
          this.forward(4);
     }
}
```



One stair, and we choose the size

```
public class StairWorld extends BuggleWorld {
     ...

     public void run() {
          StairBuggle stepper = new StairBuggle();
          stepper.buildStair(5);
     }
}


class StairBuggle extends Buggle {
     public void buildStair(int stairSize) {
          this.left();
          this.forward(stairSize-1);
          this.right();
          this.forward(stairSize-1);
     }
}
```



A stairway to heaven

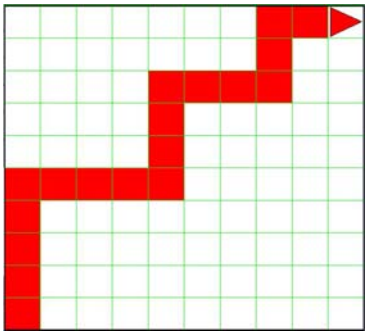
```
public class StairWorld extends BuggleWorld {
     ...

     public void run() {
          StairBuggle stepper = new StairBuggle();
          stepper.buildStair(5);
          stepper.buildStair(4);
          stepper.buildStair(3);
     }
}

class StairBuggle extends Buggle {
     public void buildStair(int stairSize) {
          this.left();
          this.forward(stairSize-1);
          this.right();
          this.forward(stairSize-1);
     }
}
```

instance method invocations

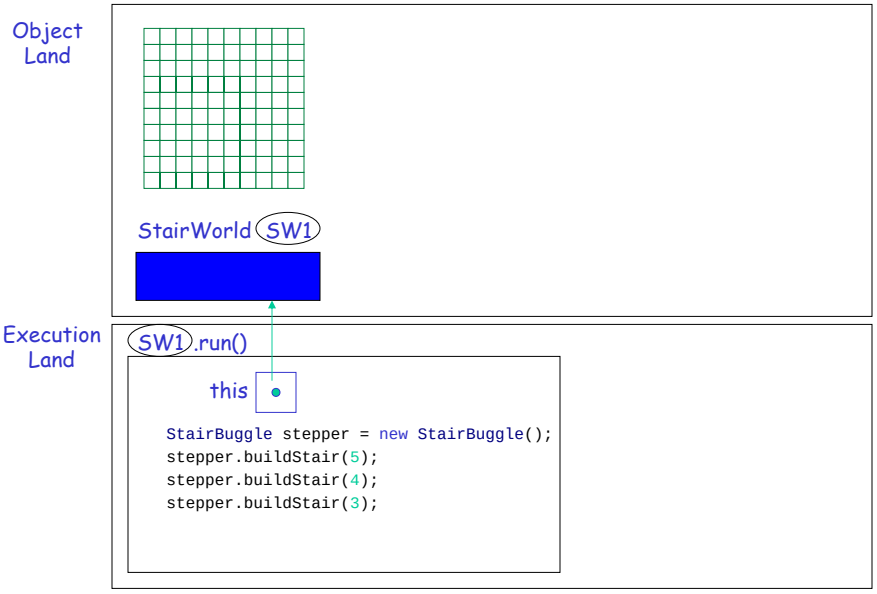
actual parameters (arguments)

instance method declaration

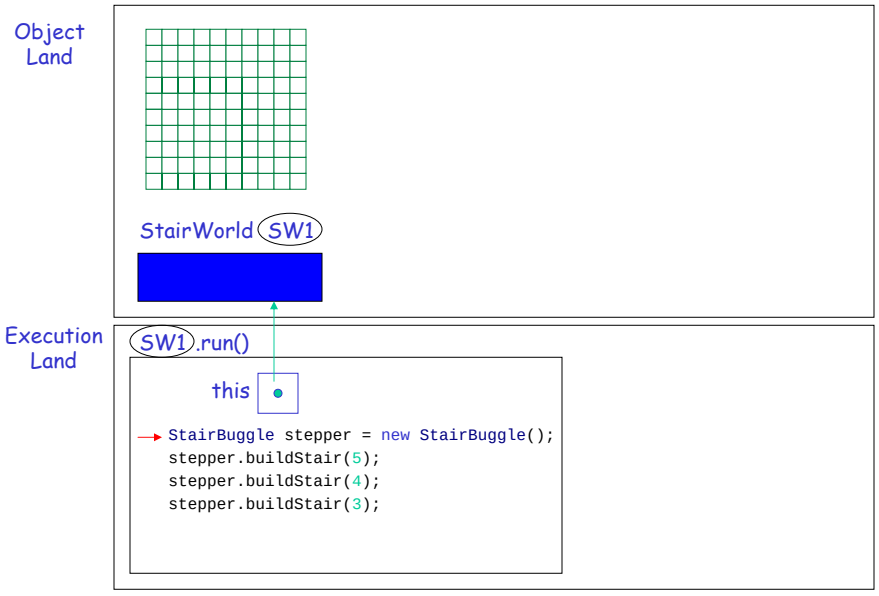
formal parameter



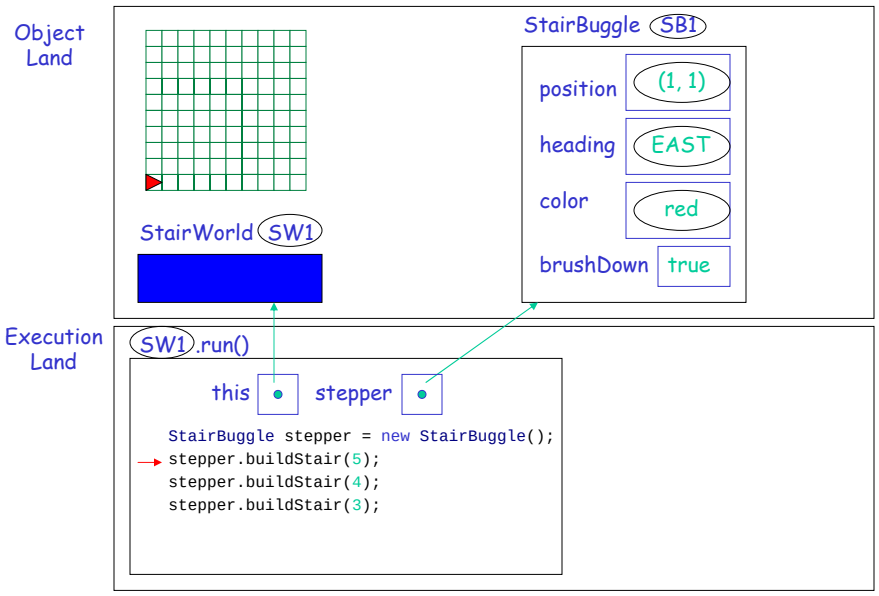
Suppose `run()` is invoked for an instance of `StairWorld`



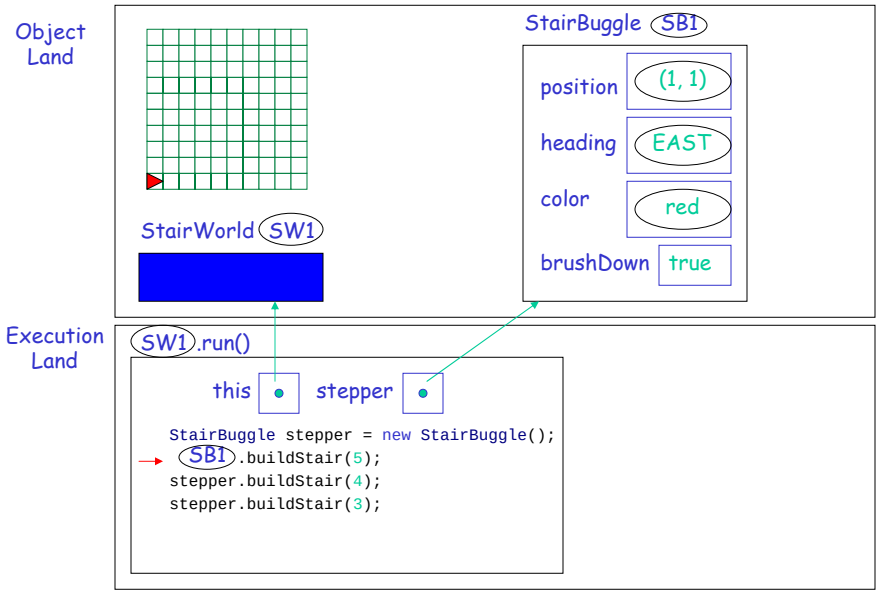
Begin with the first statement in run()



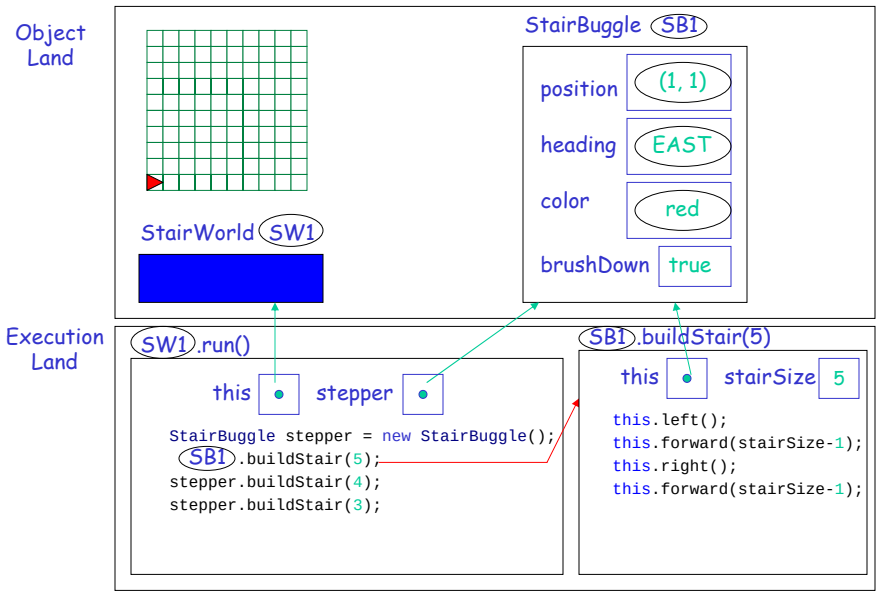
Create a StairBuggle and go to the second statement



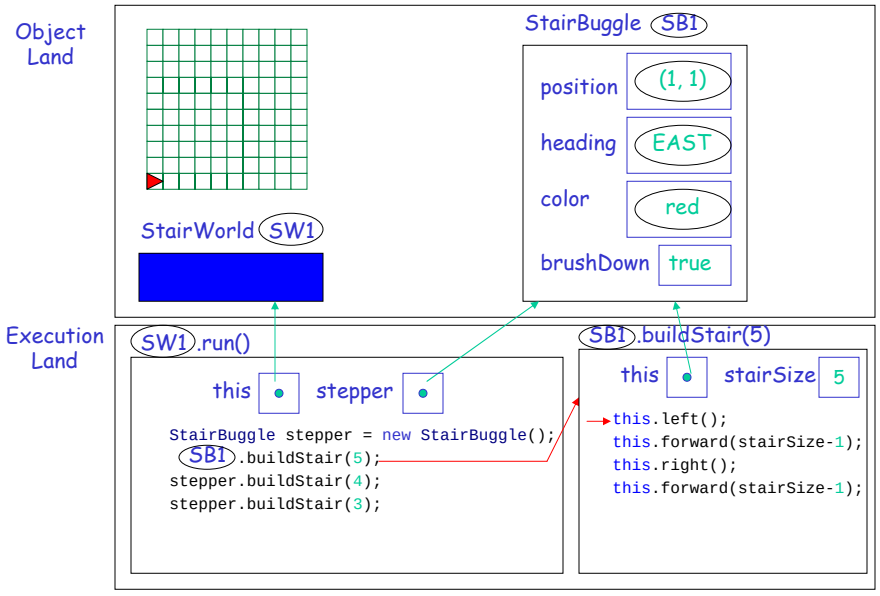
Request that the first step be built



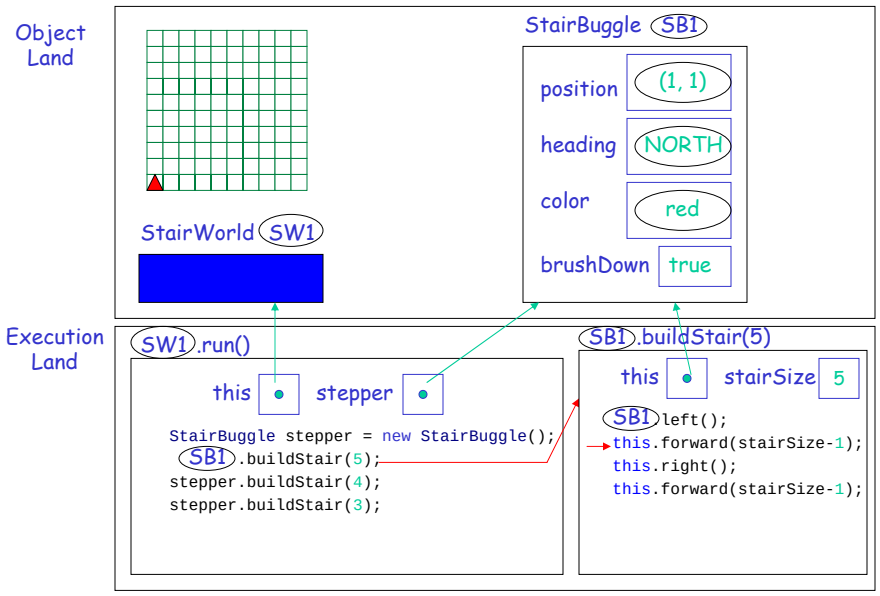
Create an execution frame for `buildStair(5)`



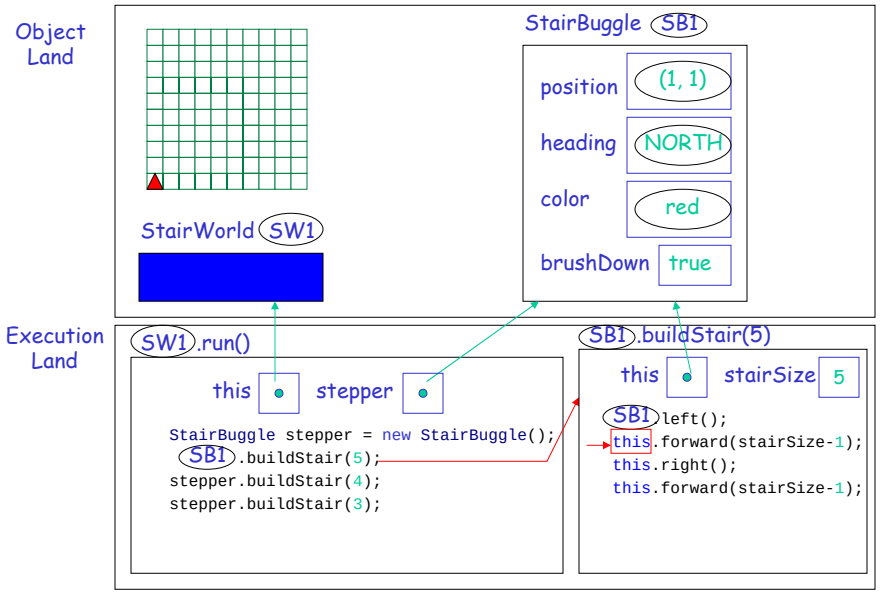
Execute the first statement in `buildStair(5)`



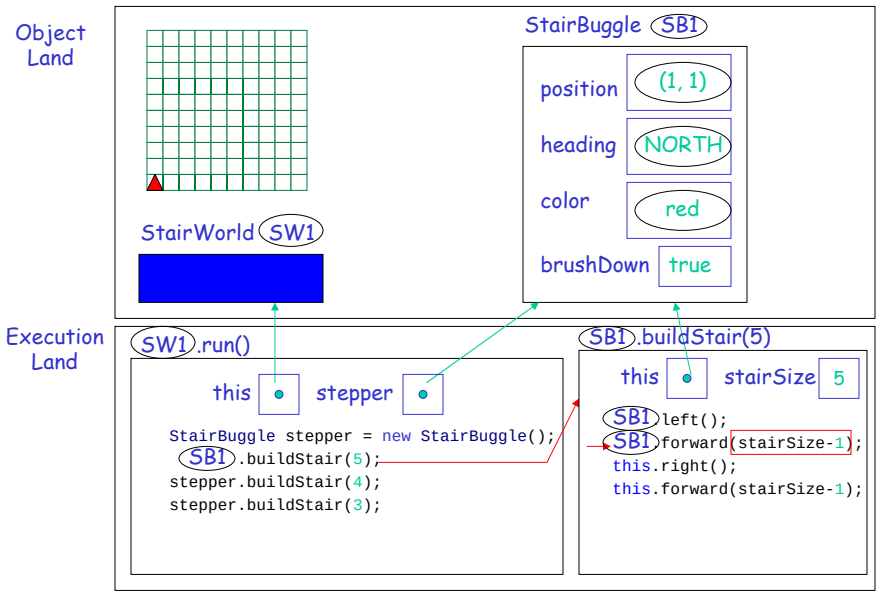
Turn left and go to the second statement in `buildStair(5)`



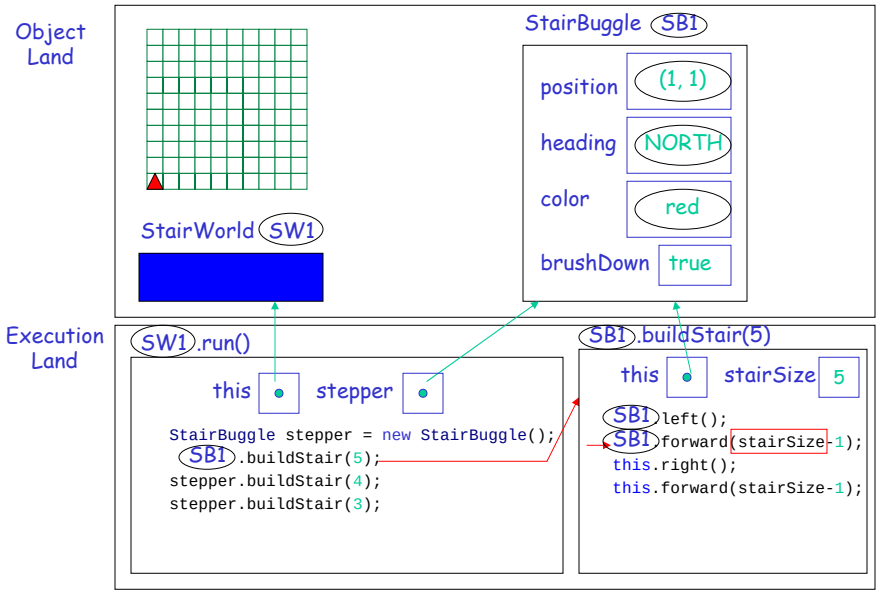
Evaluate the receiver, `this`, in the second statement



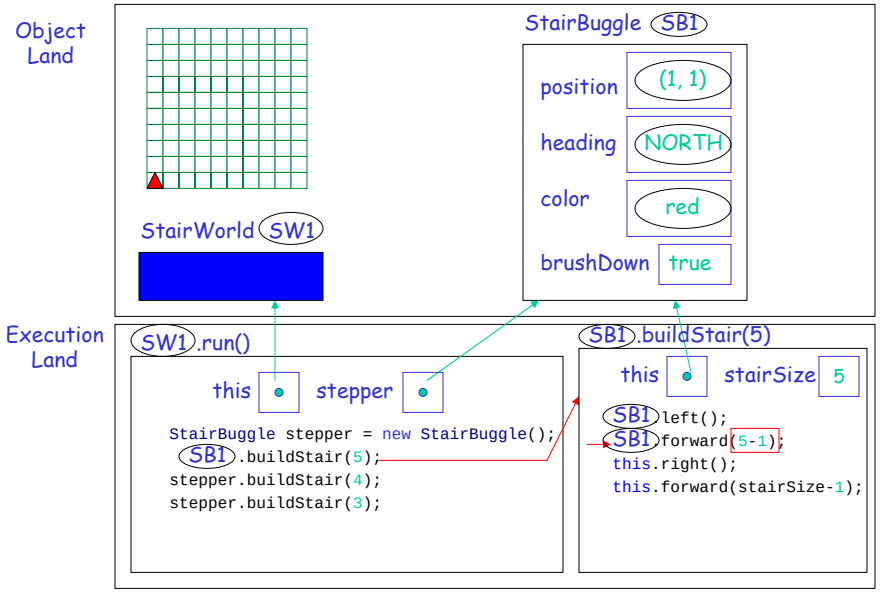
Evaluate the argument expression `stairSize - 1`



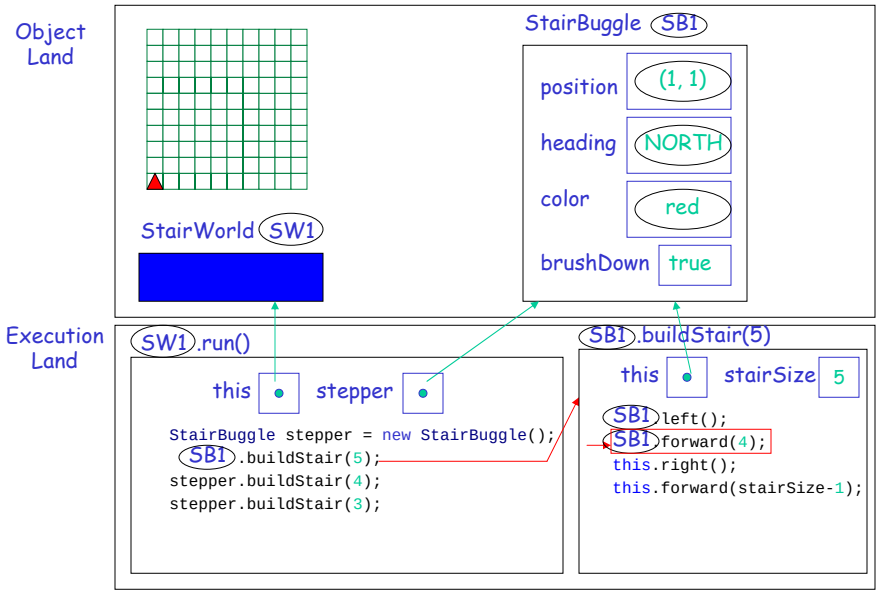
Evaluate the argument subexpression `stairSize`



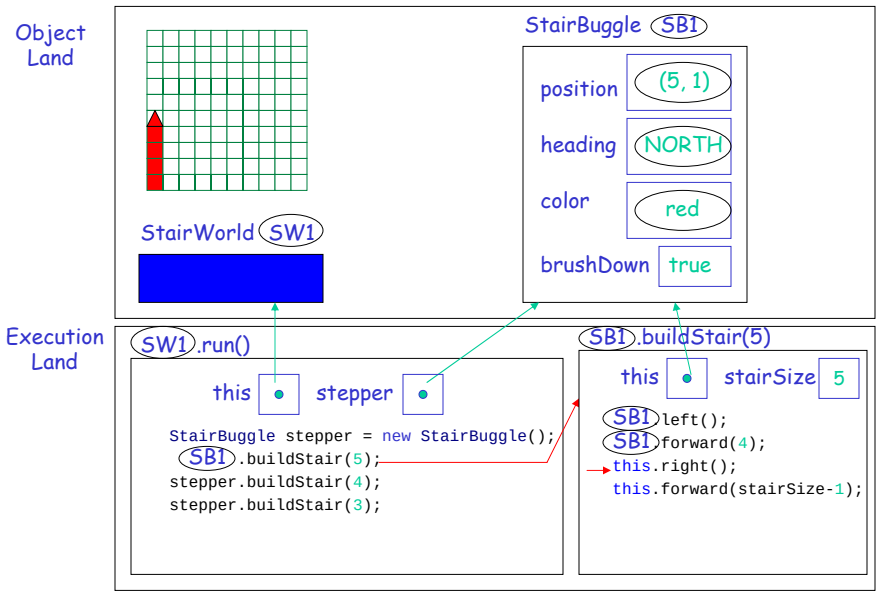
Evaluate the argument expression `5 - 1`



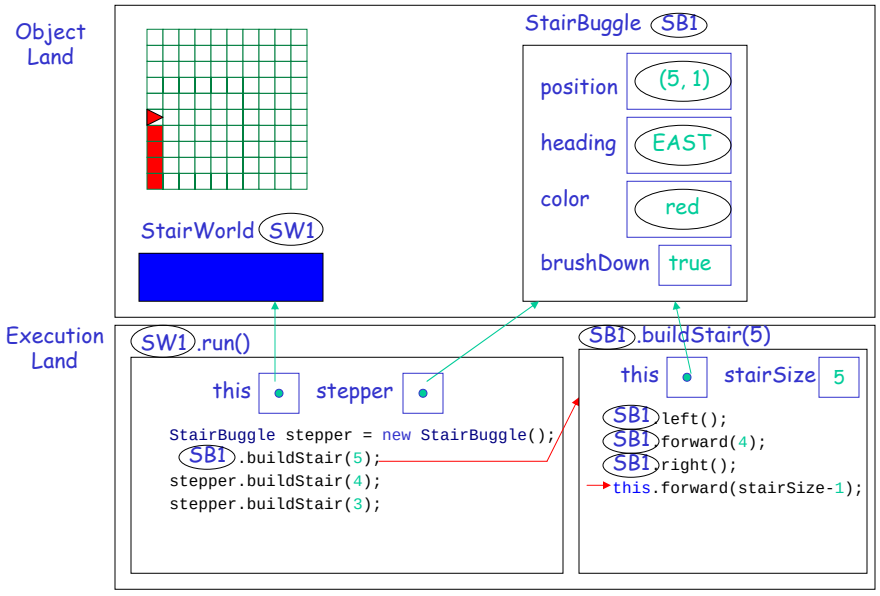
Finally ready to go forward!



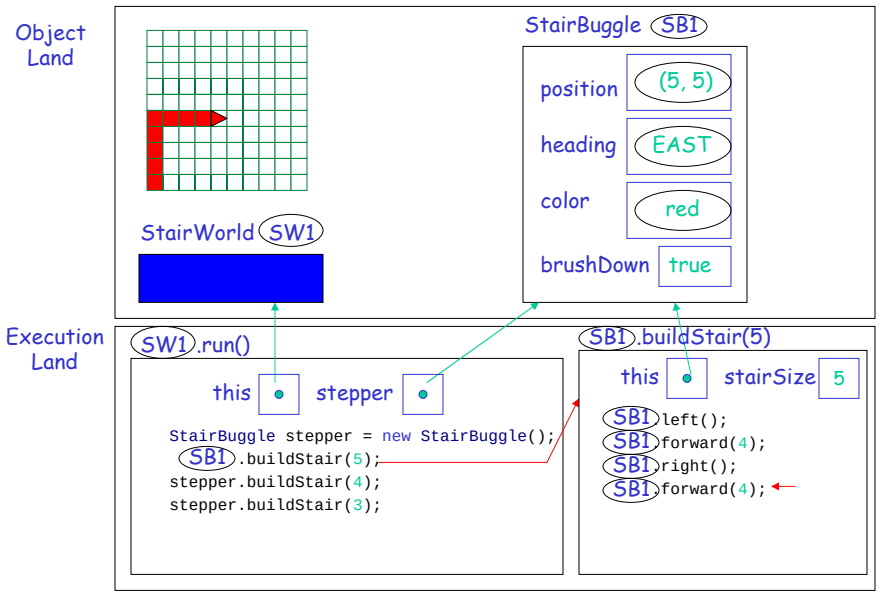
Go forward and start the third statement



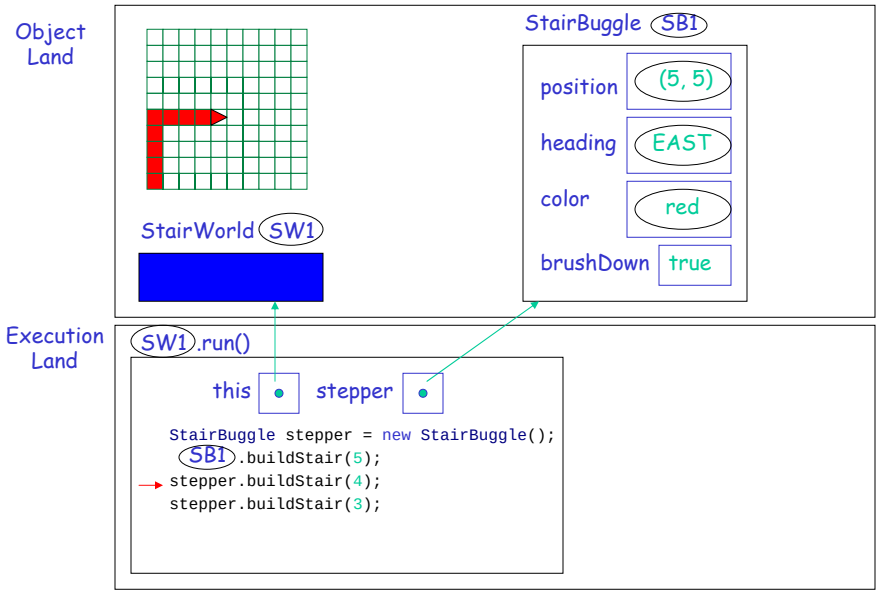
Turn right and start the fourth statement



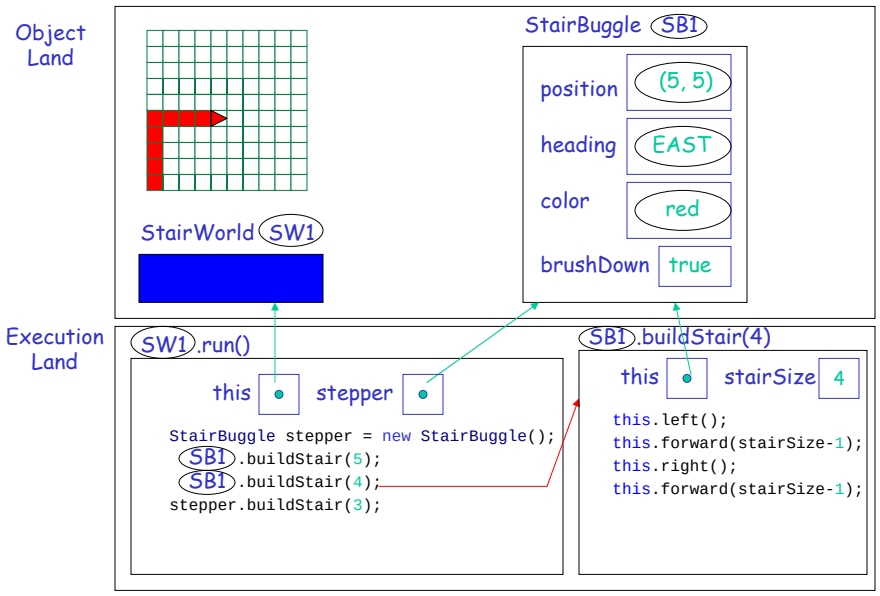
Done with the first step!



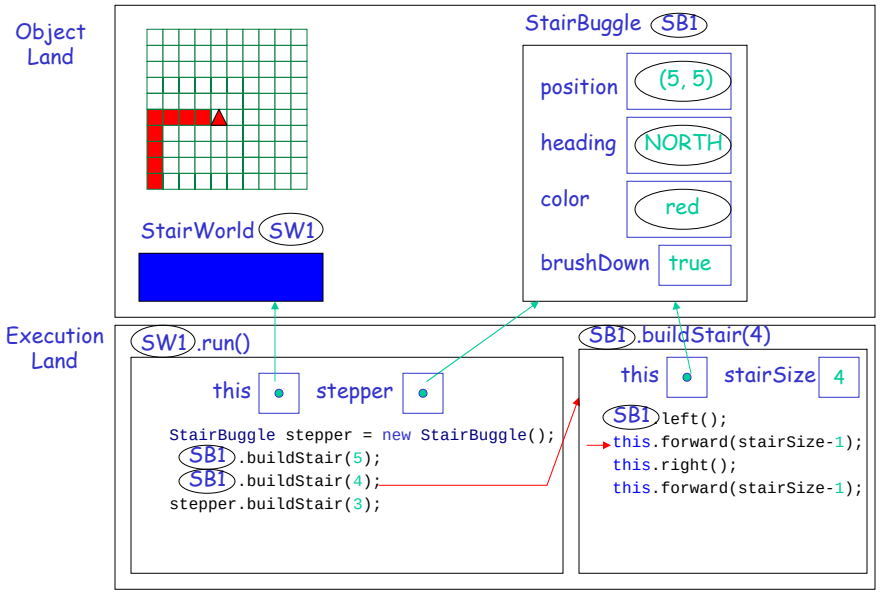
Request that the second step be built



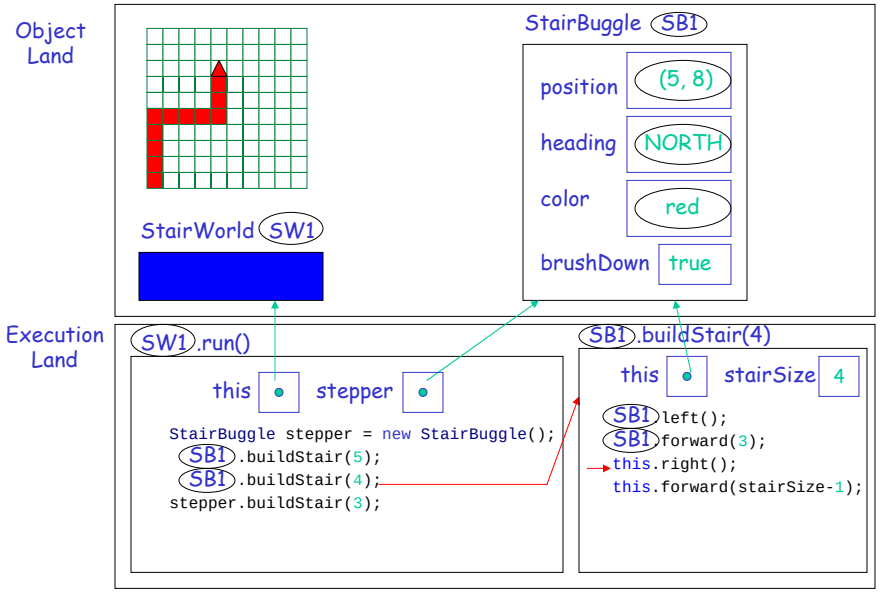
Create an execution frame for `buildStair(4)`



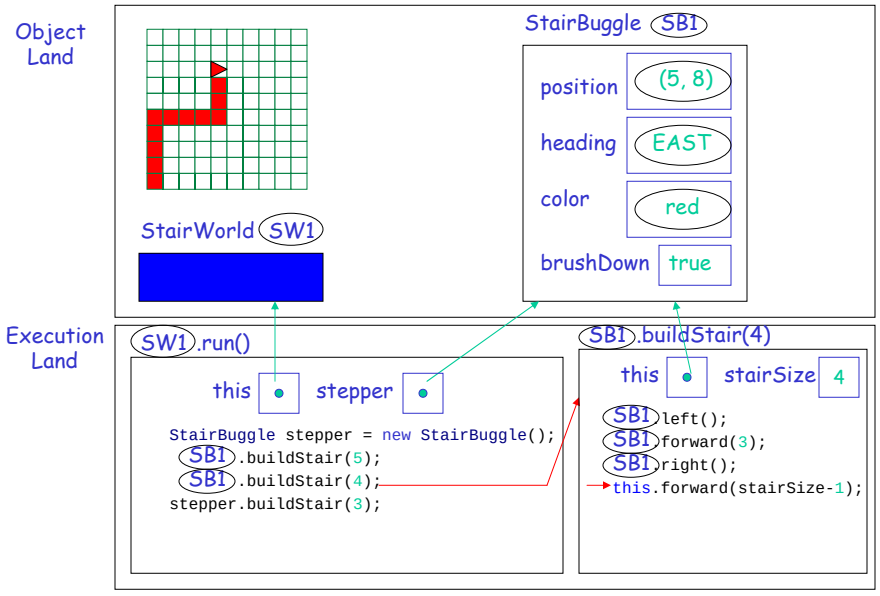
Go left at the base of the stair



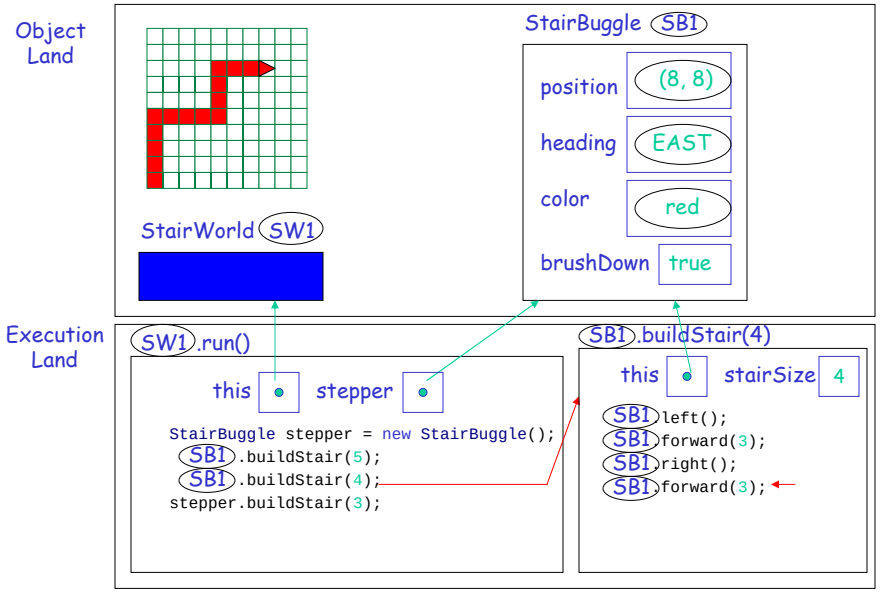
Draw the vertical of the stair



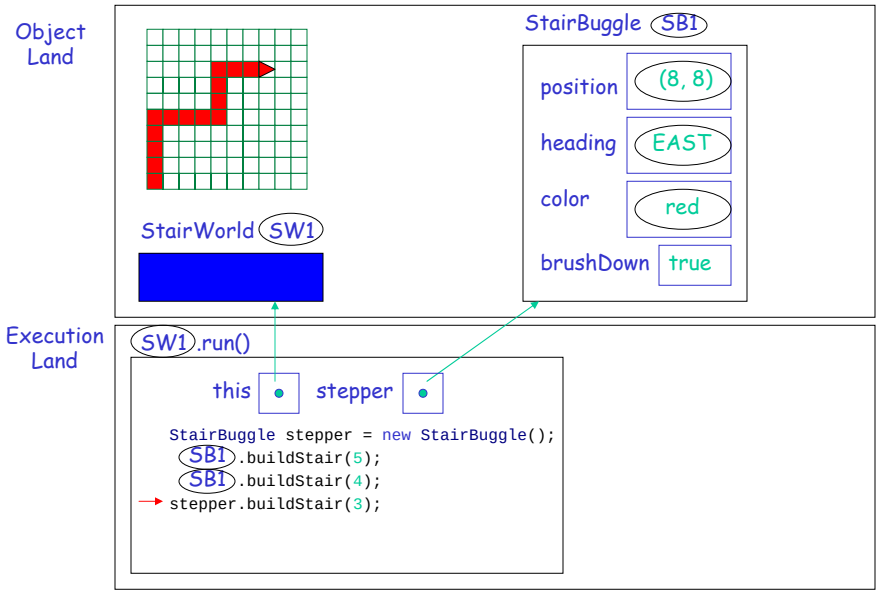
Turn right



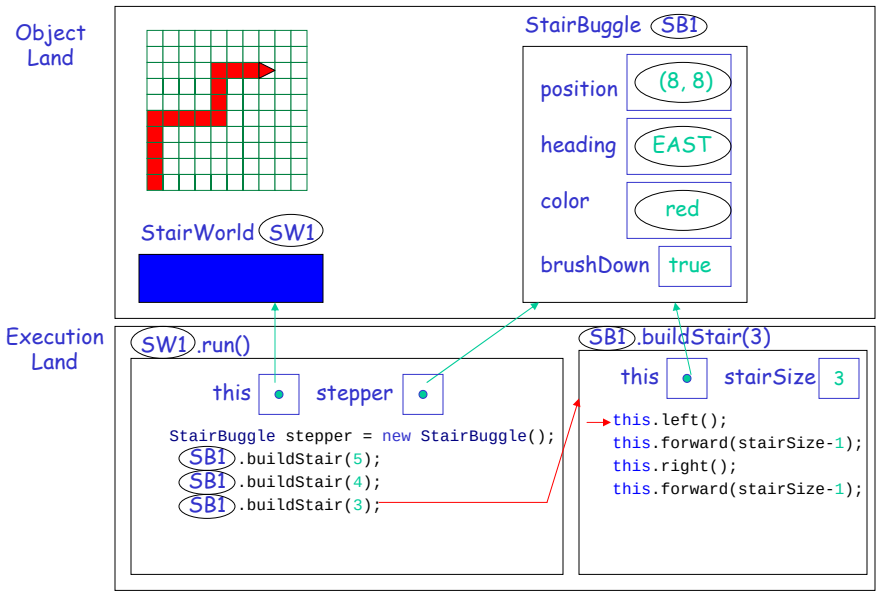
Draw the horizontal to finish the second stair



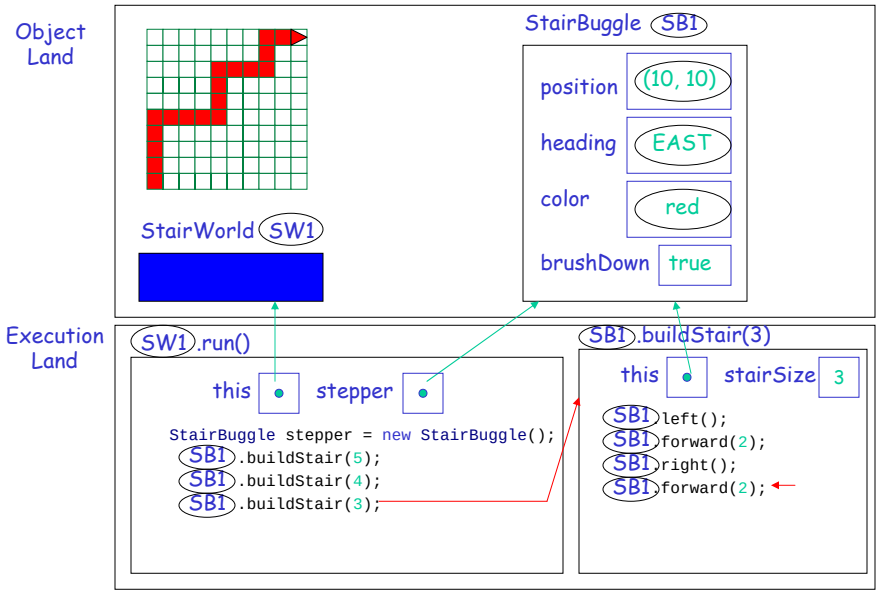
Request that the third step be built



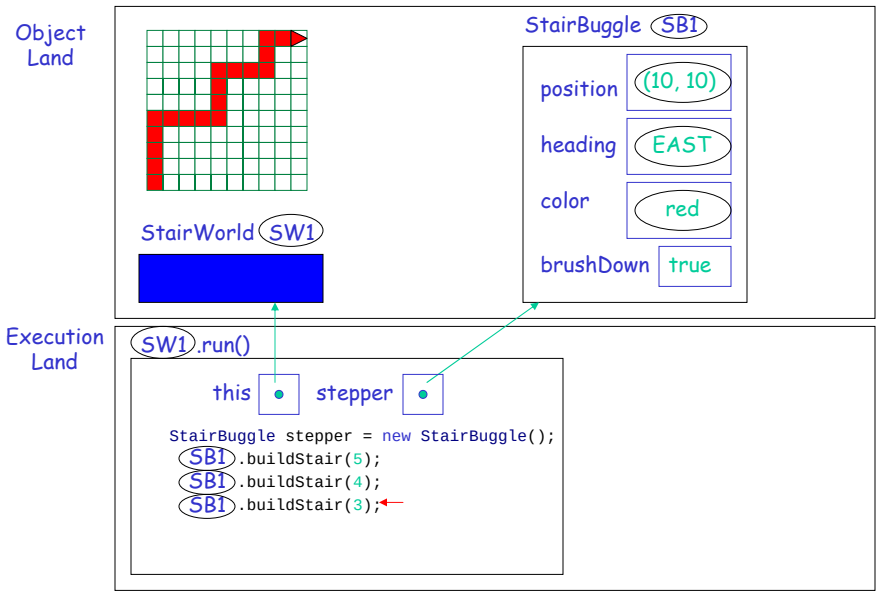
Create an execution frame for `buildStair(3)`



Build the third step in one fell swoop



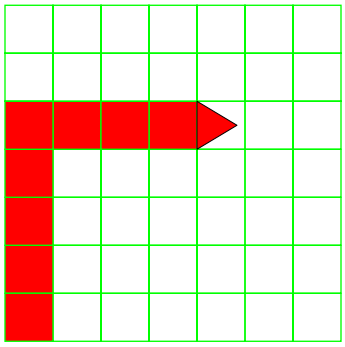
Done with run()!



A void method

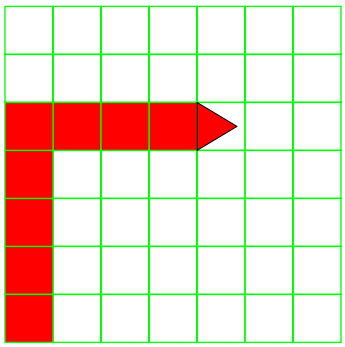
```
/* tells the StairBuggle to draw a stair */

public void buildStep(int stairSize)
{
      this.left();
      this.forward(stairSize-1);
      this.right();
      this.forward(stairSize-1);
}
```

reserved word void says method does not return a value



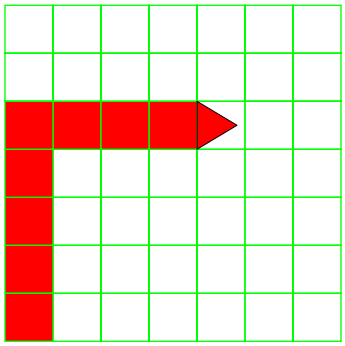
A fruitful method

```
/* tells the StairBuggle to draw a stair and
   to return the StairBuggle's position after
   completing the stair.
*/

public          buildStep(int stairSize)
{
      this.left();
      this.forward(stairSize-1);
      this.right();
      this.forward(stairSize-1);

}
```

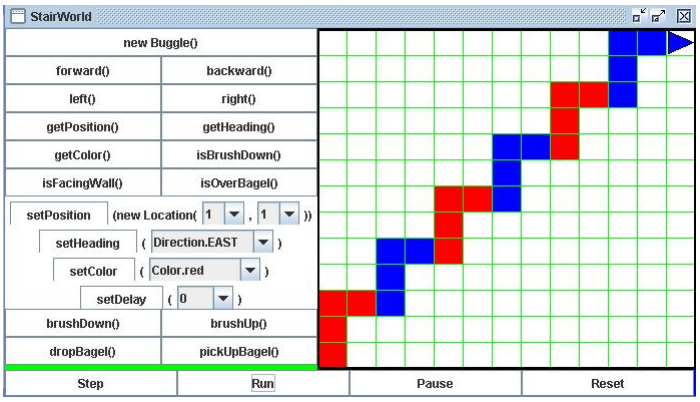


A fruitful method

```
/* tells the StairBuggle to draw a stair and
   to return the StairBuggle's position after
   completing the stair.
*/

public Location buildStep(int stairSize)
{
     this.left();
     this.forward(stairSize-1);
     this.right();
     this.forward(stairSize-1);
     return this.getPosition();
}
```

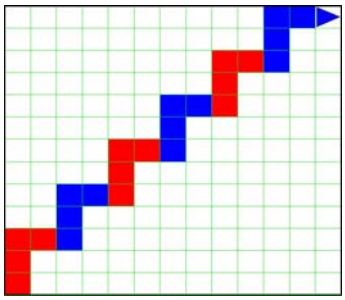


Jack and Jill go up the hill



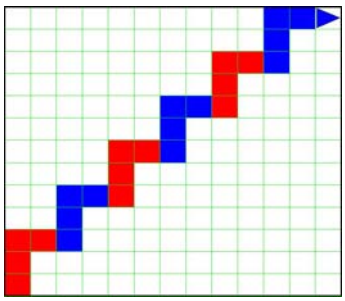
More to the point

```
public class StairWorld extends BuggleWorld {
     public void run() {
          StairBuggle jack = new StairBuggle();
          StairBuggle jill = new StairBuggle();
          jill.setColor(Color.blue);

          // our new code goes here

     }
}

class StairBuggle extends Buggle {
     public Location buildStair(int stairSize) {
          this.left();
          this.forward(stairSize-1);
          this.right();
          this.forward(stairSize-1);
          return this.getPosition();
     }
}
```

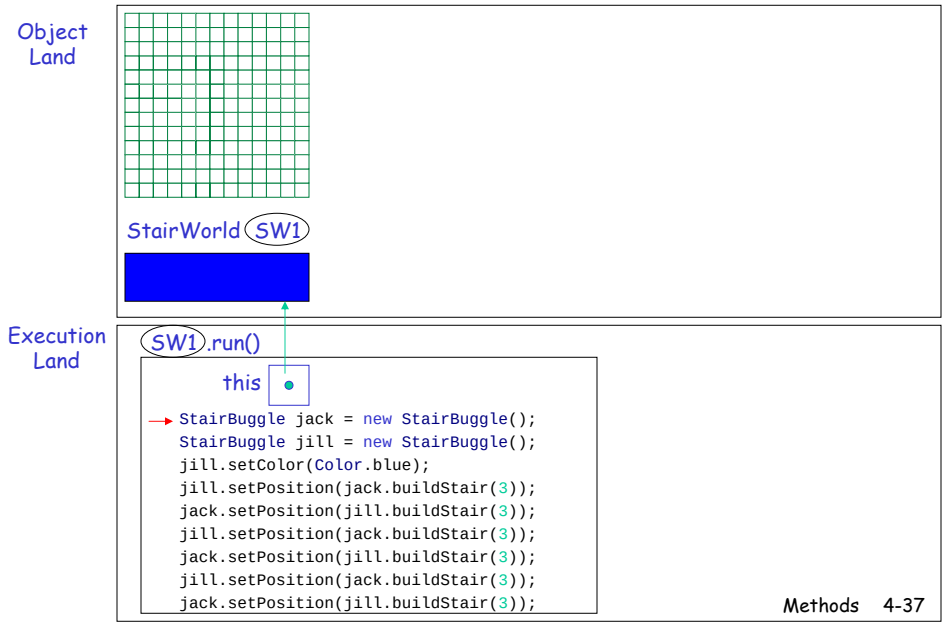


Jack and Jill and Jack and Jill and Jack and Jill

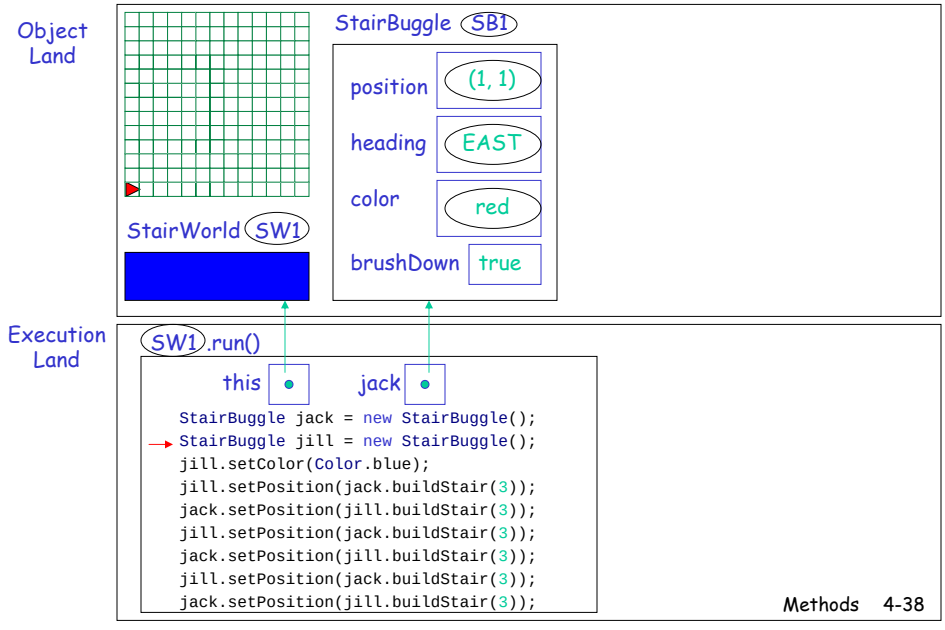
```
public class StairWorld extends BuggleWorld {
     public void run() {
          StairBuggle jack = new StairBuggle();
          StairBuggle jill = new StairBuggle();
          jill.setColor(Color.blue);
          jill.setPosition(jack.buildStair(3));
          jack.setPosition(jill.buildStair(3));
          jill.setPosition(jack.buildStair(3));
          jack.setPosition(jill.buildStair(3));
          jill.setPosition(jack.buildStair(3));
          jack.setPosition(jill.buildStair(3));
     }
}

class StairBuggle extends Buggle {
     public Location buildStair(int stairSize) {
          this.left();
          this.forward(stairSize-1);
          this.right();
          this.forward(stairSize-1);
          return this.getPosition();
     }
}
```



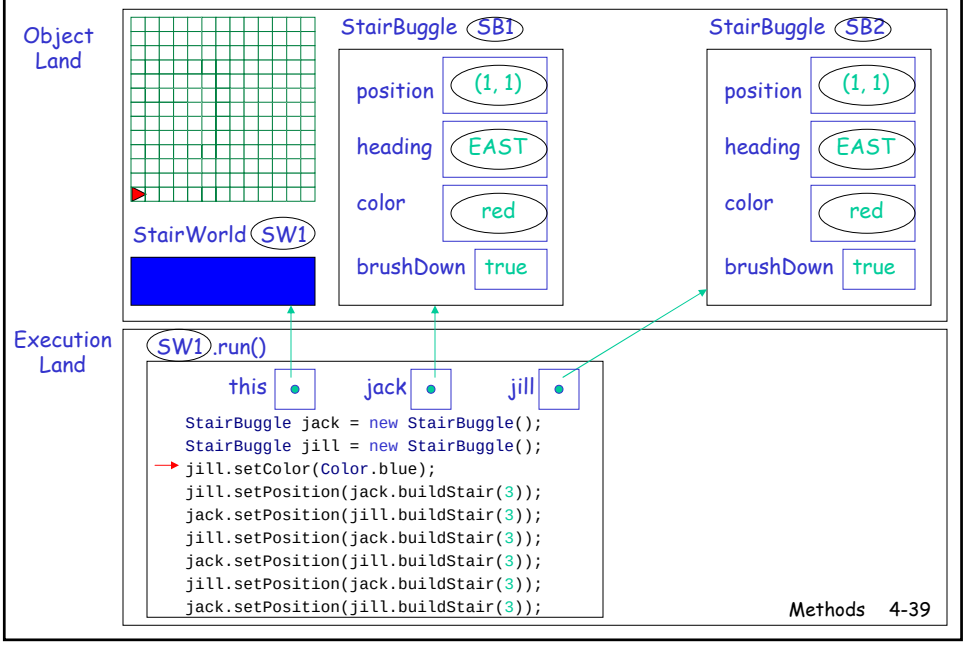
Suppose `run()` is invoked for an instance of `StairWorld`



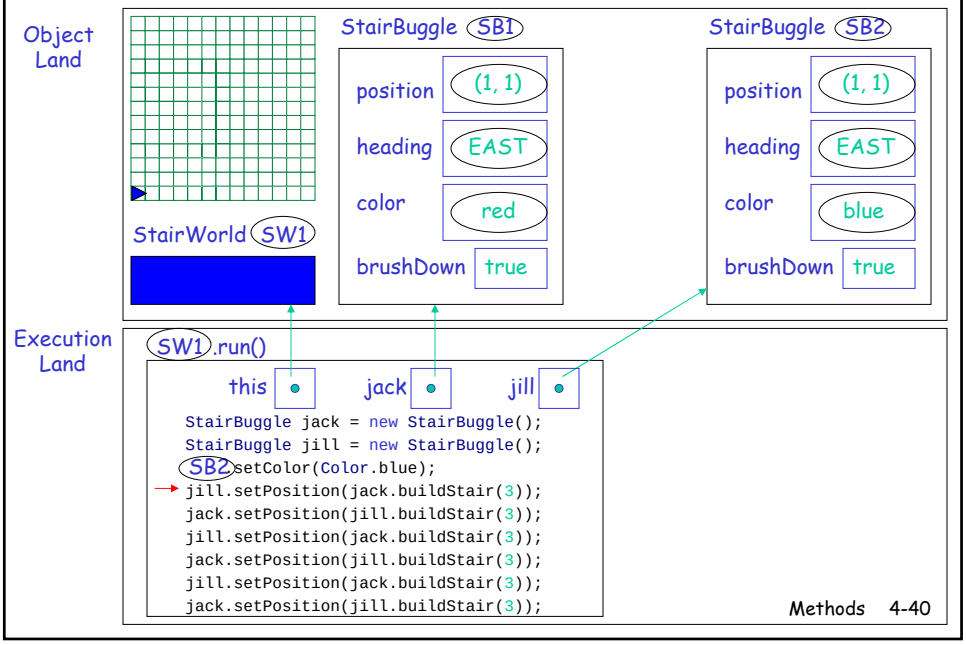
Jack is born



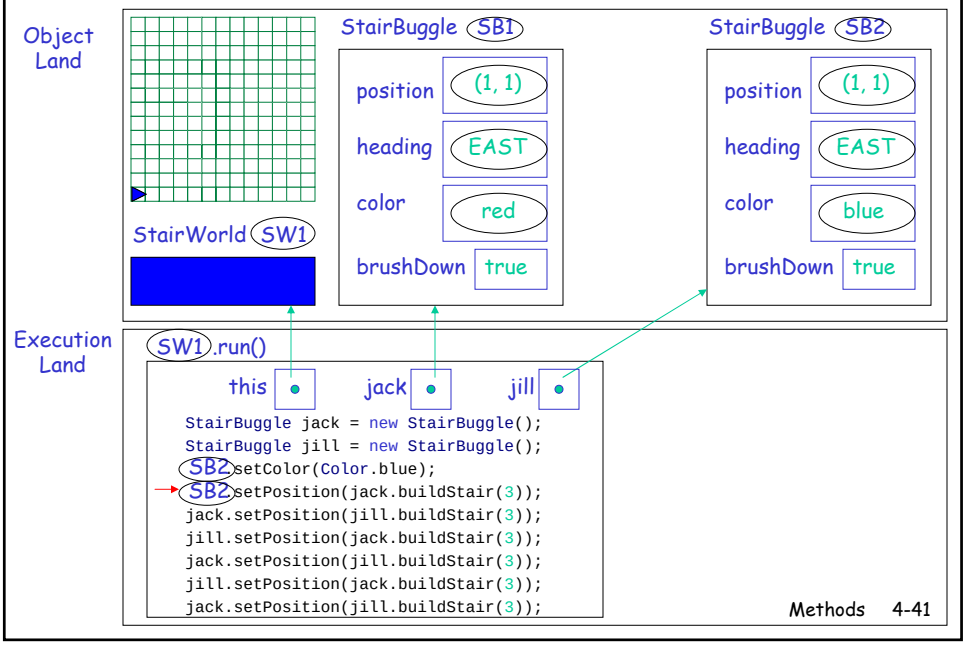
Jill is Born (rooming with Jack)



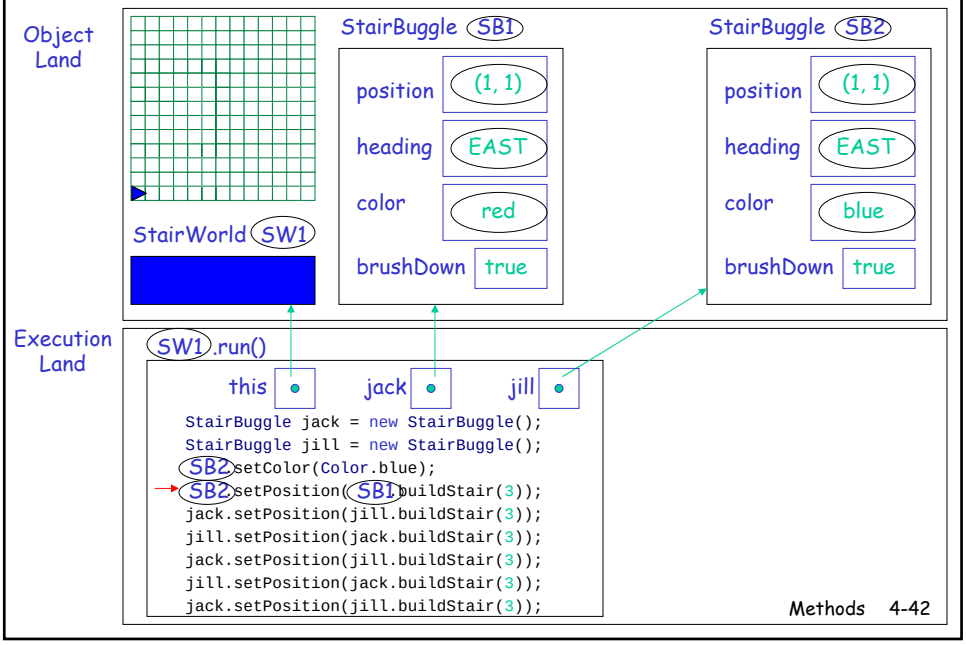
Jill feels blue



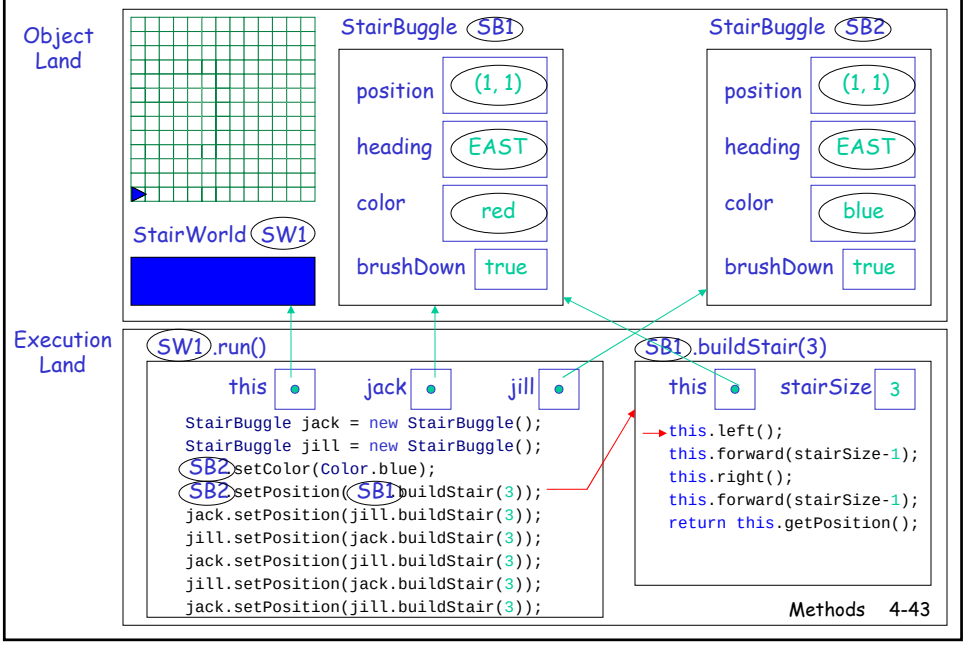
Jill is the receiver of setPosition()



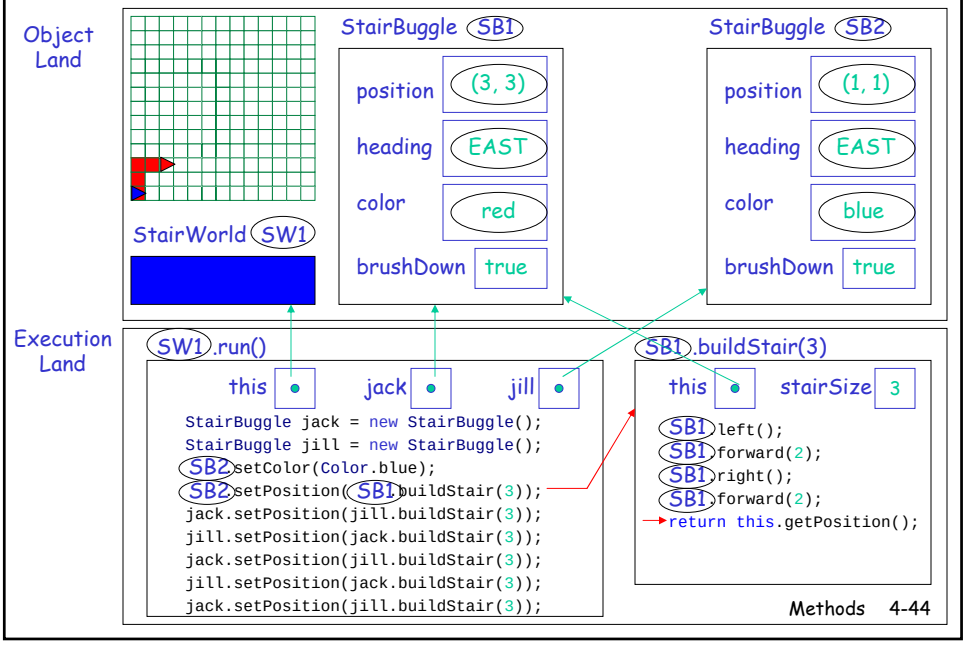
setPosition() needs a Location argument

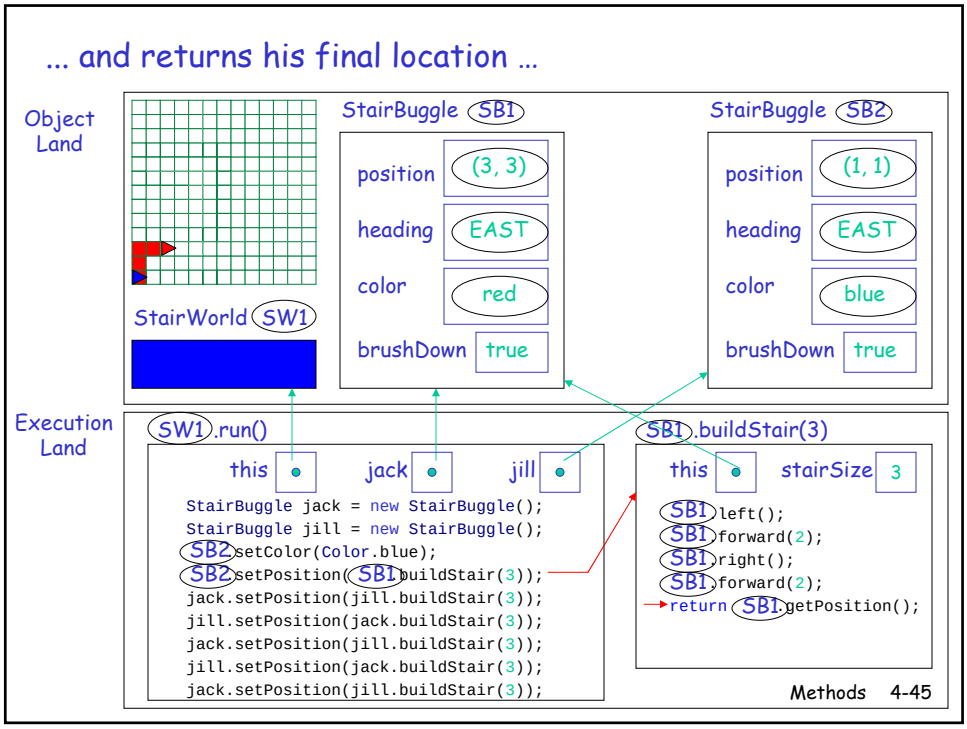


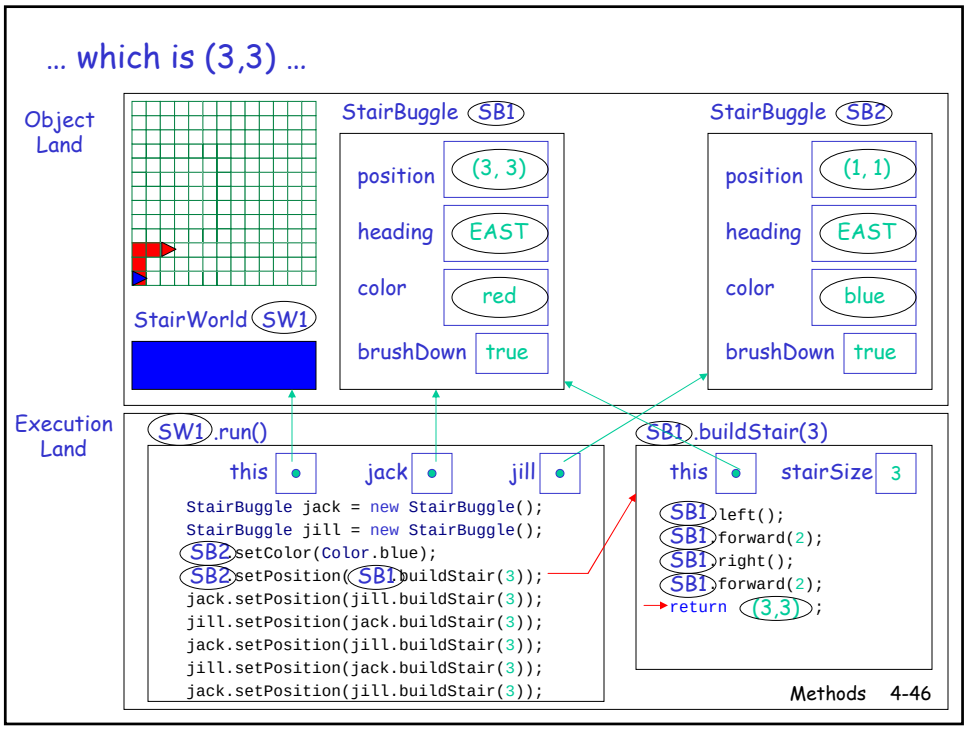
Create an execution frame for Jack's `buildStair(3)`



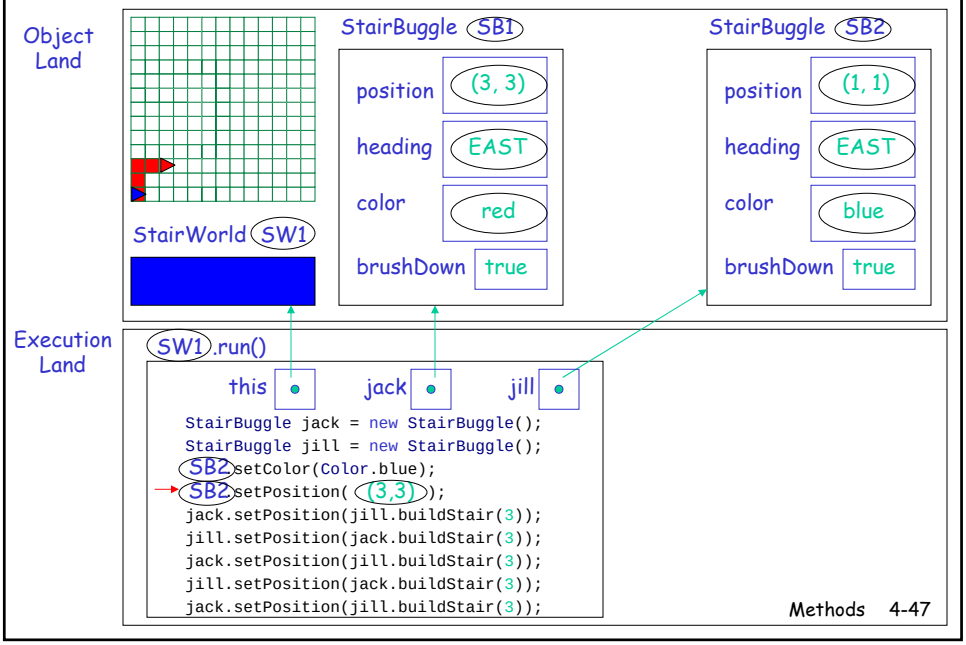
Jack draws the first stair …



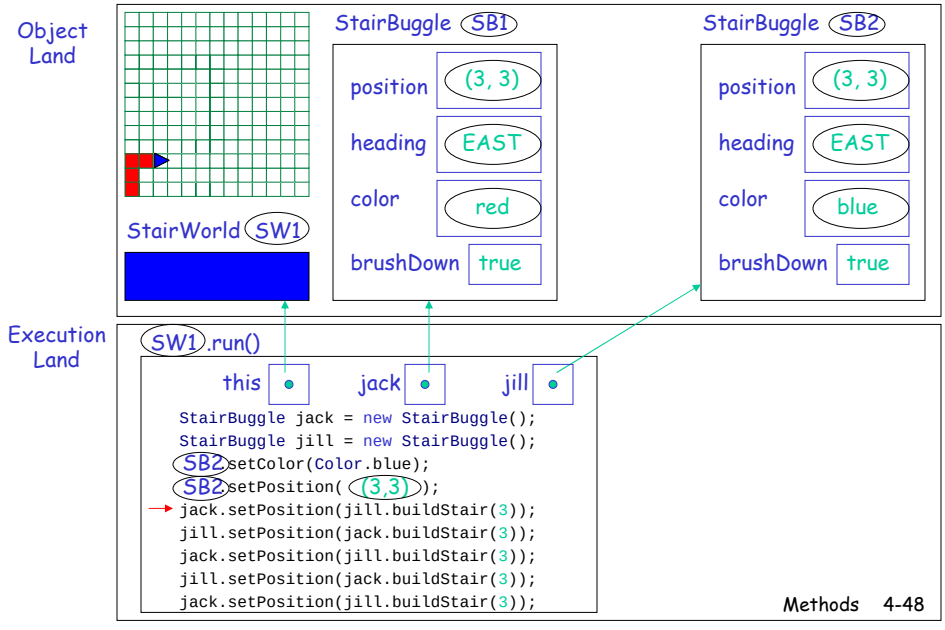

... and returns his final location ...
Object Land
StairBuggle SB1
position (3, 3)
heading EAST
color red
brushDown true
StairBuggle SB2
position (1, 1)
heading EAST
color blue
brushDown true
StairWorld SW1
Execution Land
SW1.run()
this
jack
jill
StairBuggle jack = new StairBuggle();
StairBuggle jill = new StairBuggle();
SB2.setColor(Color.blue);
SB2.setPosition(SB1.buildStair(3));
jack.setPosition(jill.buildStair(3));
jill.setPosition(jack.buildStair(3));
jack.setPosition(jill.buildStair(3));
jill.setPosition(jack.buildStair(3));
jack.setPosition(jill.buildStair(3));
SB1.buildStair(3)
this
stairSize 3
SB1.left();
SB1.forward(2);
SB1.right();
SB1.forward(2);
return SB1.getPosition();
Methods 4-45


... which is (3,3) ...
Object Land
StairBuggle SB1
position (3, 3)
heading EAST
color red
brushDown true
StairBuggle SB2
position (1, 1)
heading EAST
color blue
brushDown true
StairWorld SW1
Execution Land
SW1.run()
this
jack
jill
StairBuggle jack = new StairBuggle();
StairBuggle jill = new StairBuggle();
SB2.setColor(Color.blue);
SB2.setPosition(SB1.buildStair(3));
jack.setPosition(jill.buildStair(3));
jill.setPosition(jack.buildStair(3));
jack.setPosition(jill.buildStair(3));
jill.setPosition(jack.buildStair(3));
jack.setPosition(jill.buildStair(3));
SB1.buildStair(3)
this
stairSize 3
SB1.left();
SB1.forward(2);
SB1.right();
SB1.forward(2);
return (3,3);
Methods 4-46

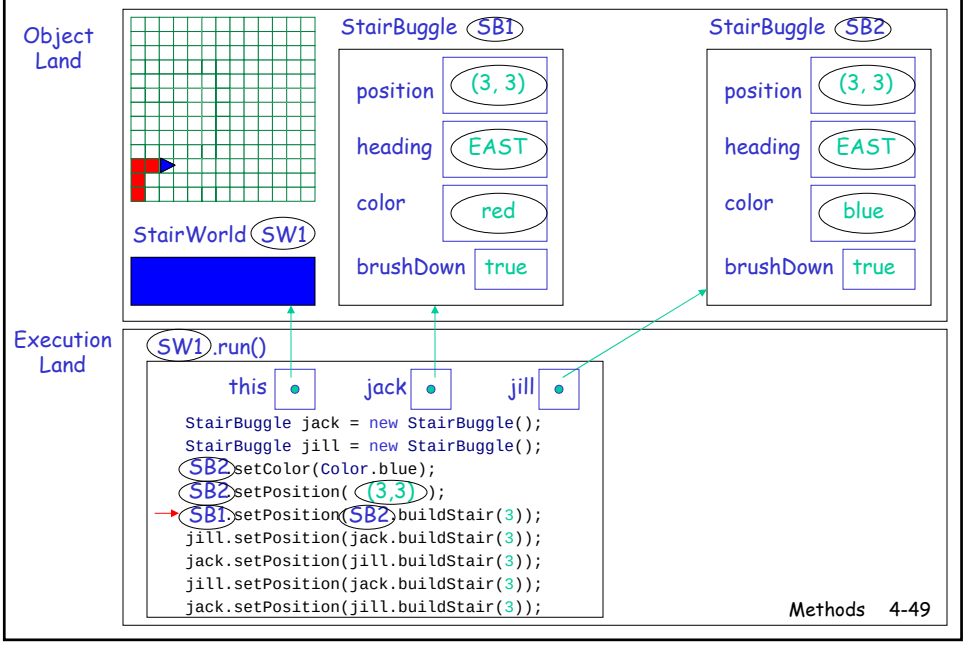
… to Jill's setPosition() invocation



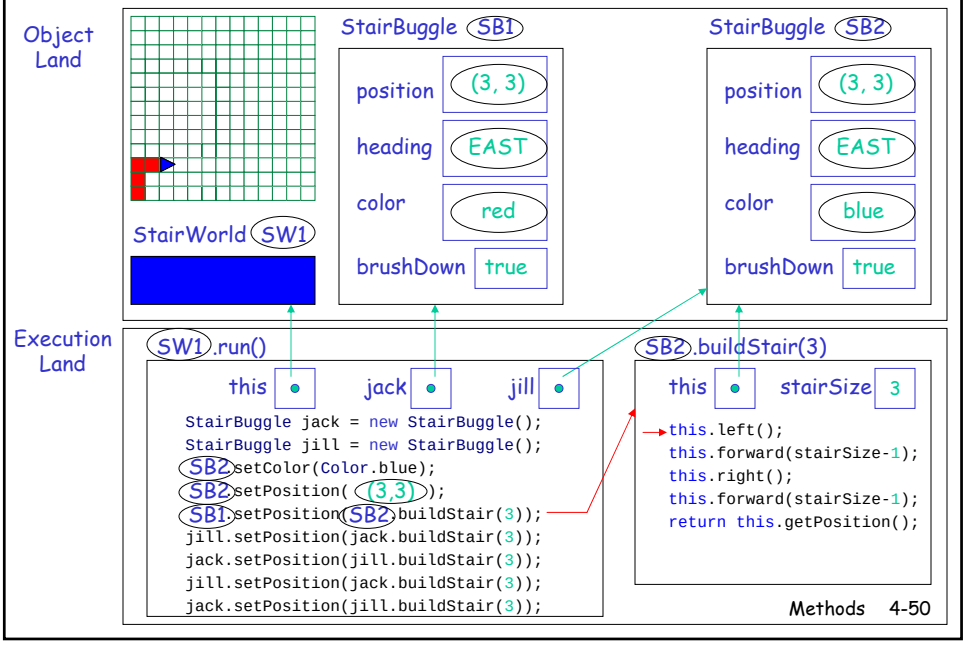
Jill joins Jack at his location



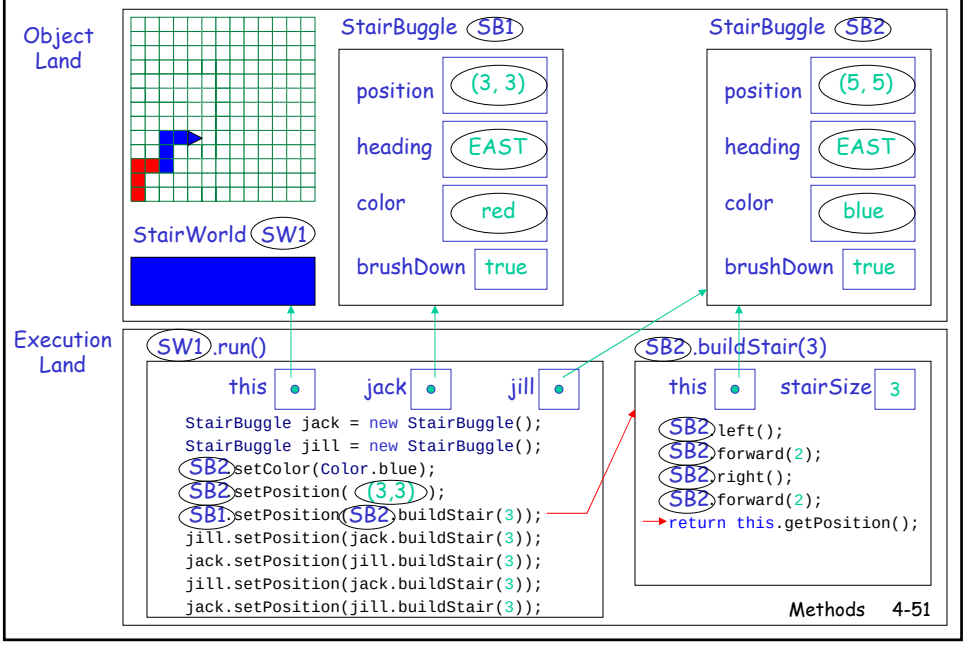
Now the roles are reversed



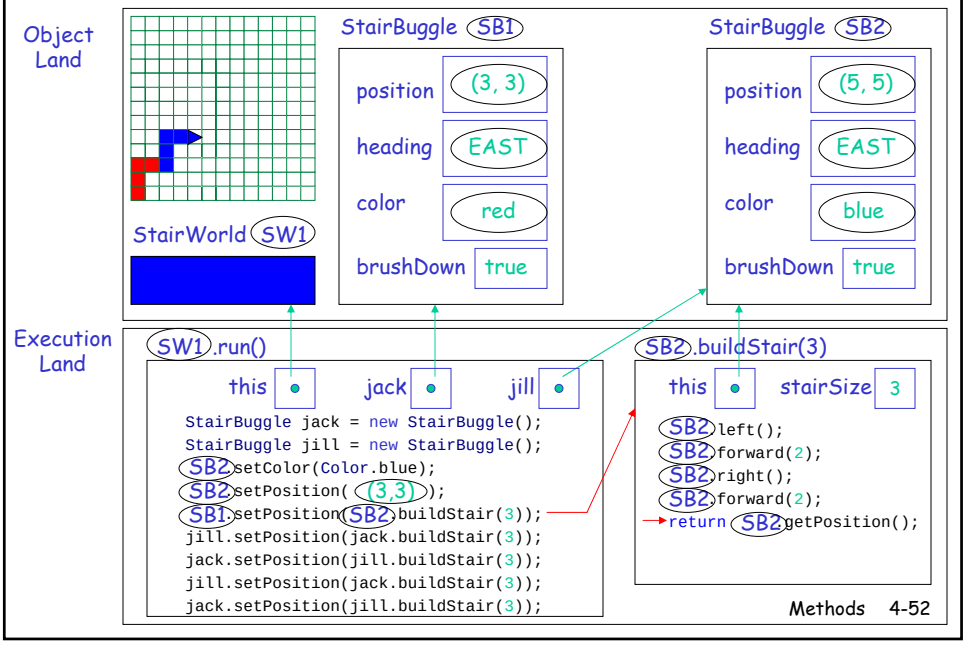
Create an execution frame for Jill's `buildStair(3)`

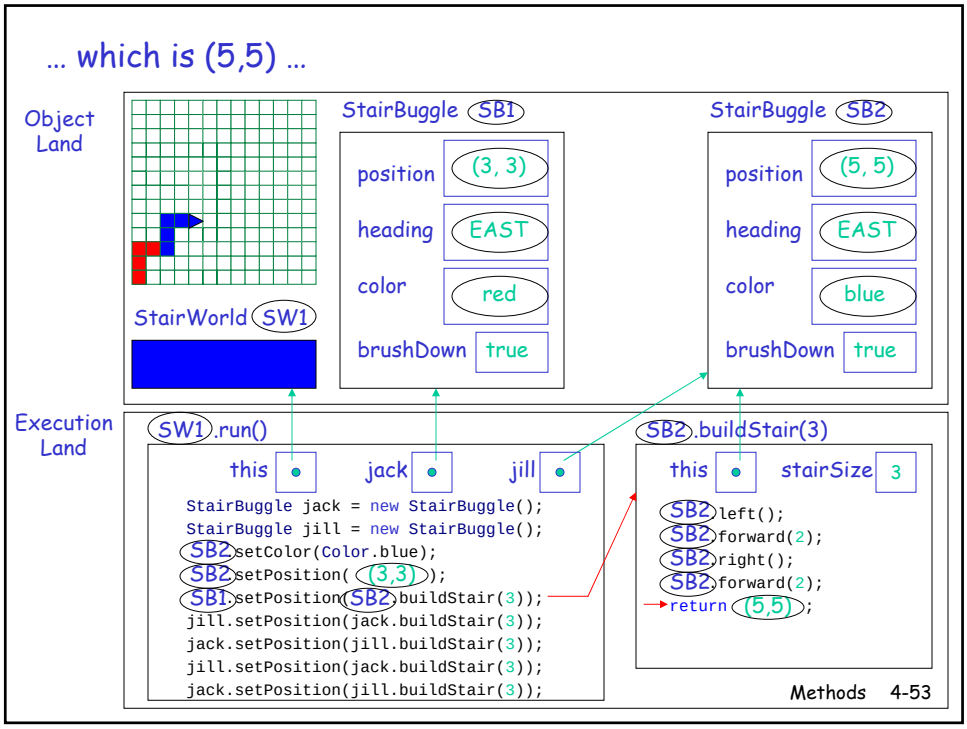


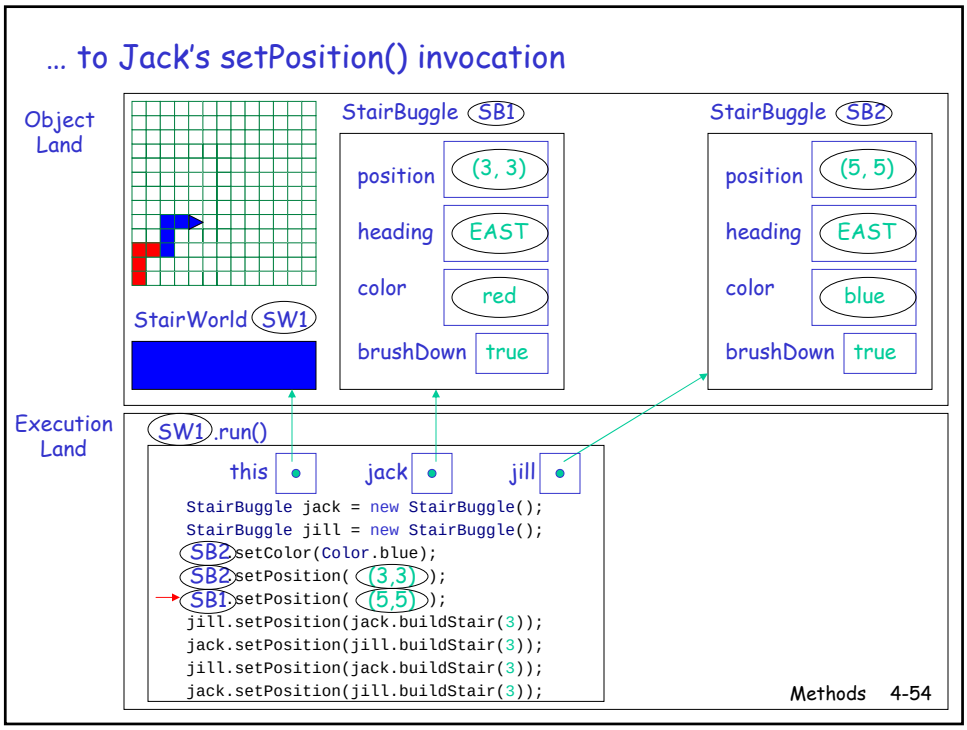
Jill draws the second stair …



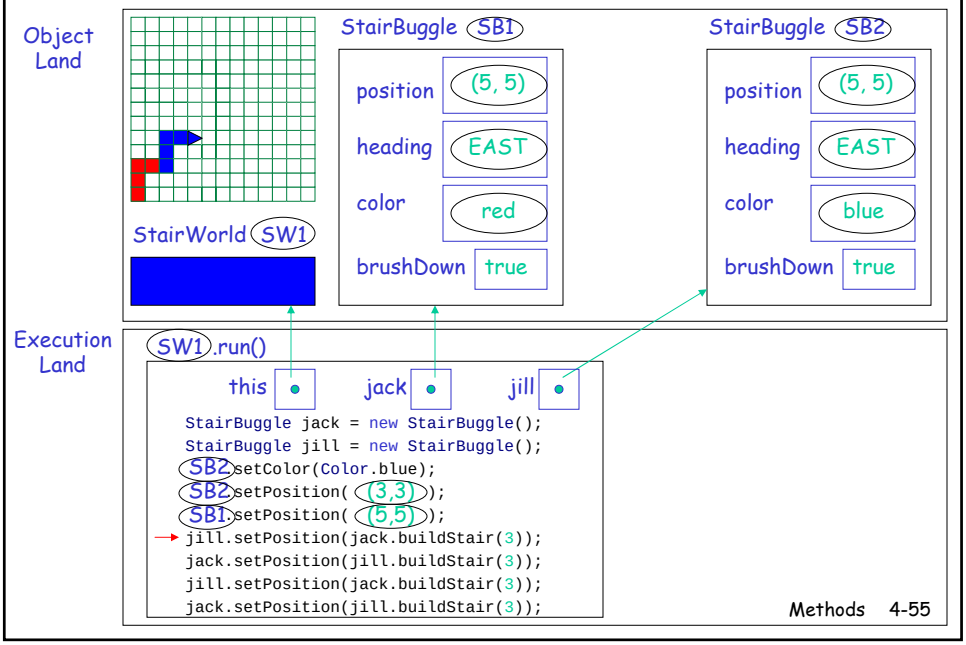
… and returns her location …



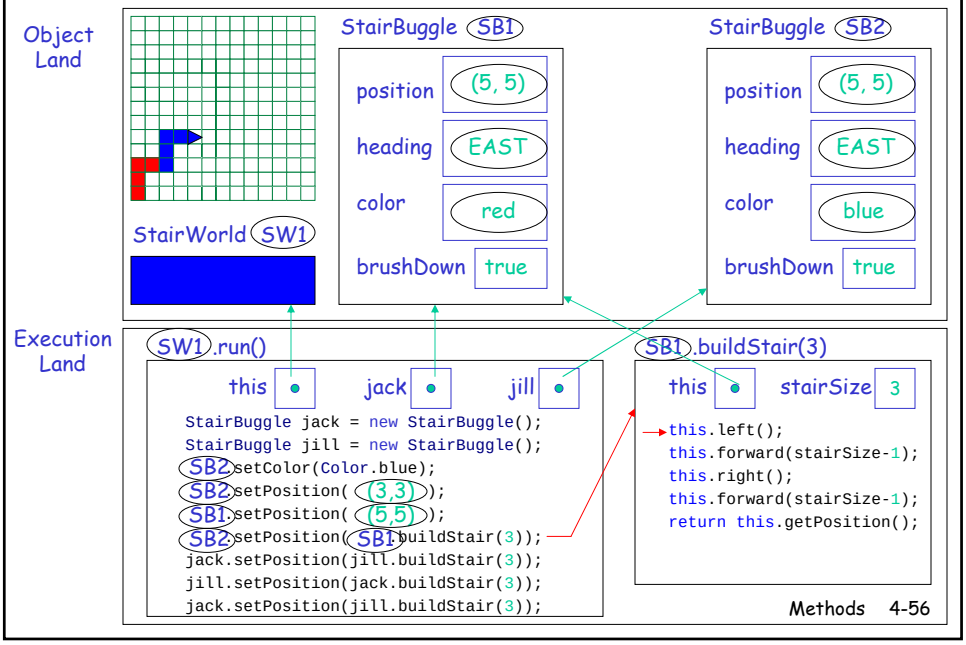

... which is (5,5) ...
Object Land
StairBuggle SB1
position (3, 3)
heading EAST
color red
brushDown true
StairBuggle SB2
position (5, 5)
heading EAST
color blue
brushDown true
StairWorld SW1
Execution Land
SW1.run()
this jack jill
StairBuggle jack = new StairBuggle();
StairBuggle jill = new StairBuggle();
SB2.setColor(Color.blue);
SB2.setPosition((3,3));
SB1.setPosition(SB2.buildStair(3));
jill.setPosition(jack.buildStair(3));
jack.setPosition(jill.buildStair(3));
jill.setPosition(jack.buildStair(3));
jack.setPosition(jill.buildStair(3));
SB2.buildStair(3)
this stairSize 3
SB2.left();
SB2.forward(2);
SB2.right();
SB2.forward(2);
return (5,5);
Methods 4-53


... to Jack's setPosition() invocation
Object Land
StairBuggle SB1
position (3, 3)
heading EAST
color red
brushDown true
StairBuggle SB2
position (5, 5)
heading EAST
color blue
brushDown true
StairWorld SW1
Execution Land
SW1.run()
this jack jill
StairBuggle jack = new StairBuggle();
StairBuggle jill = new StairBuggle();
SB2.setColor(Color.blue);
SB2.setPosition((3,3));
SB1.setPosition((5,5));
jill.setPosition(jack.buildStair(3));
jack.setPosition(jill.buildStair(3));
jill.setPosition(jack.buildStair(3));
jack.setPosition(jill.buildStair(3));
Methods 4-54

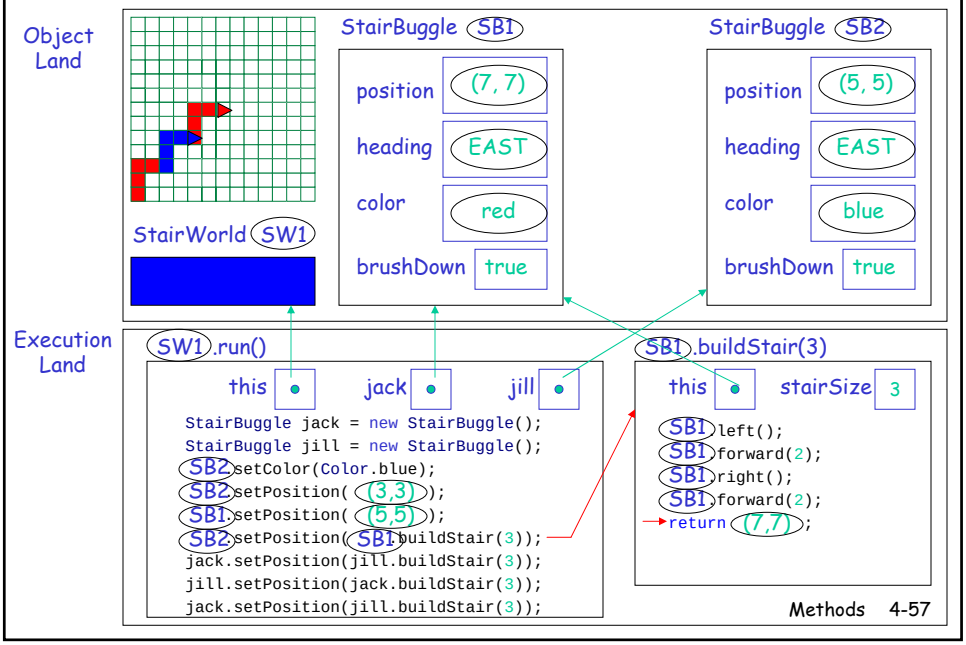
Jack joins Jill at her location



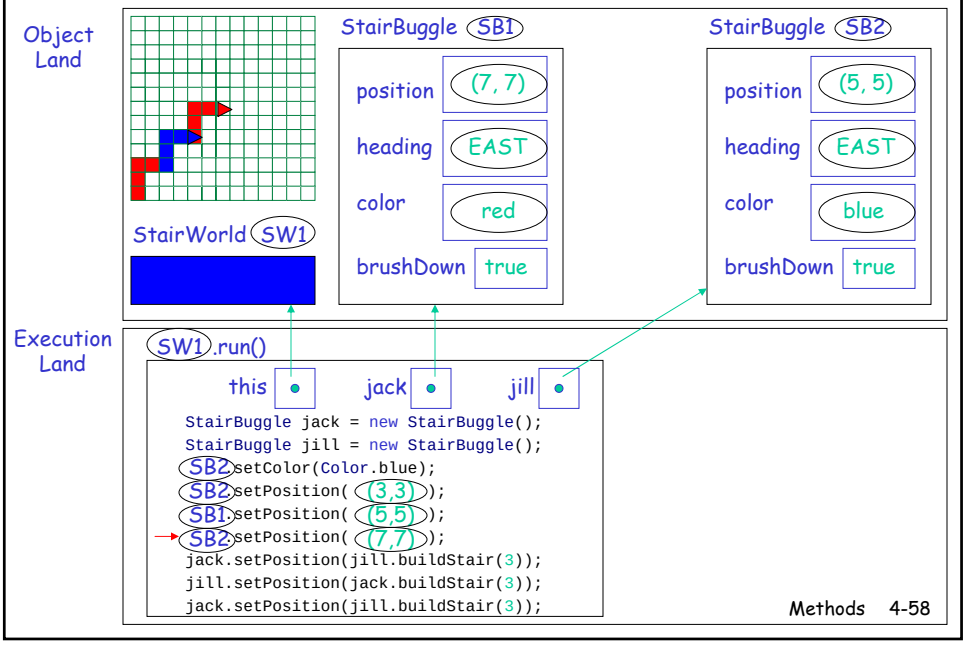
Jack begins to draw the third stair



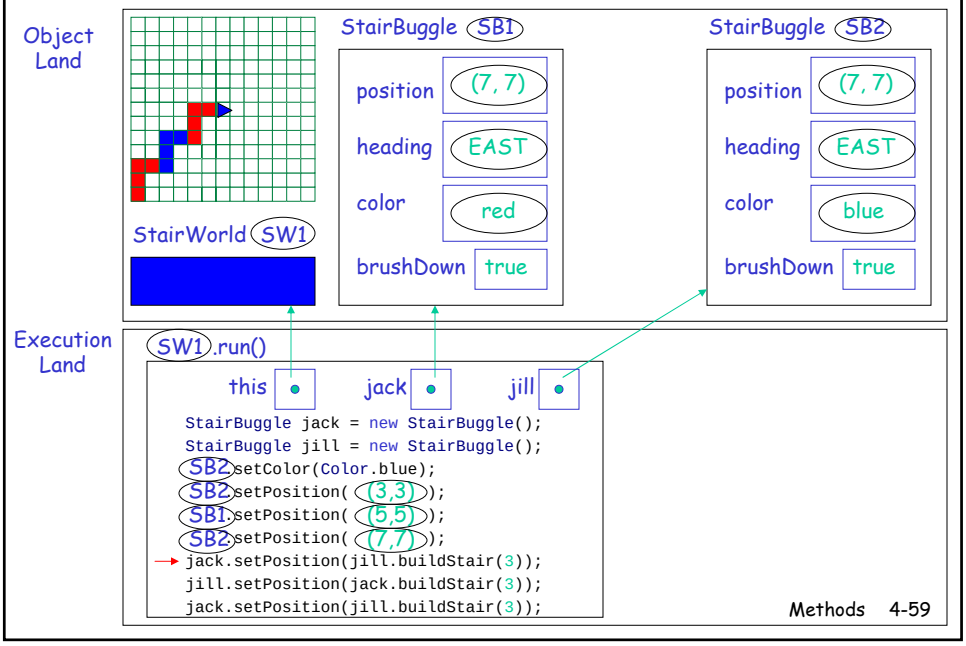
Jack returns the location at the end of the third step …



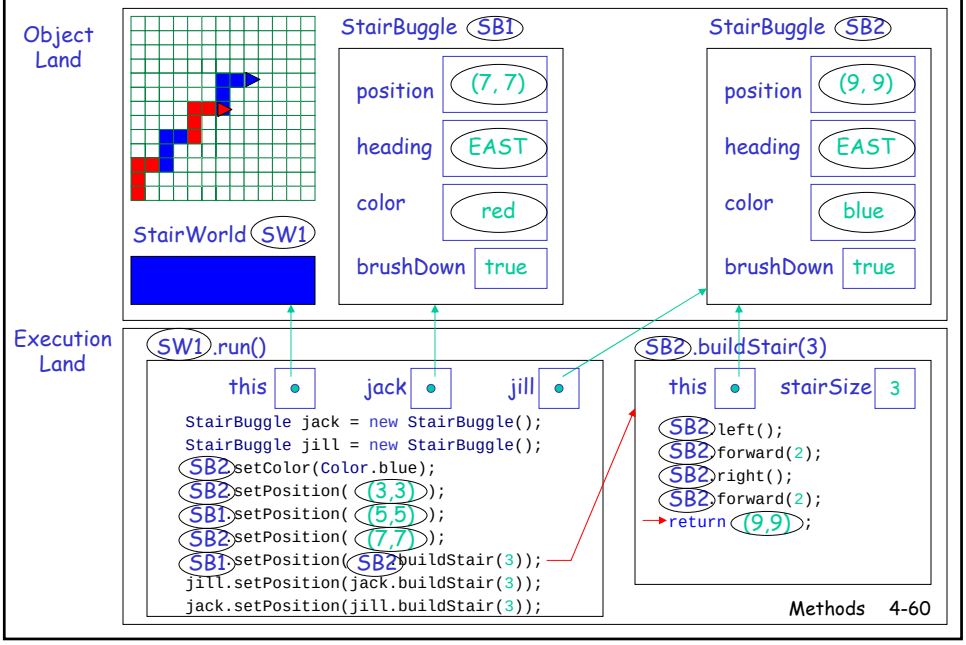
… to Jill's setPosition() invocation



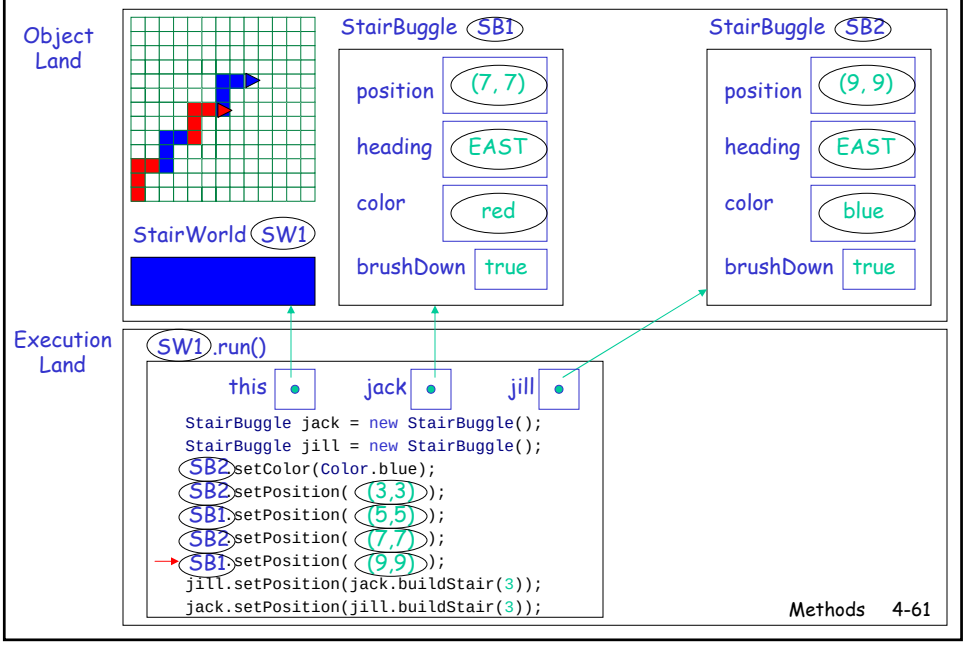
Jill joins Jack at (7,7)



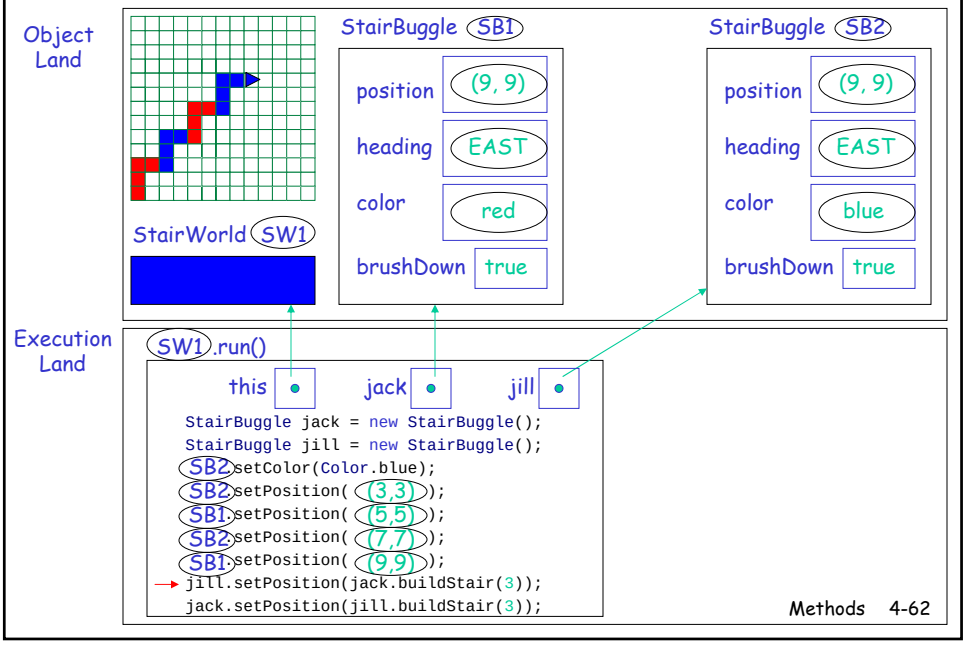
Jill draws the fourth step, and returns her final position …



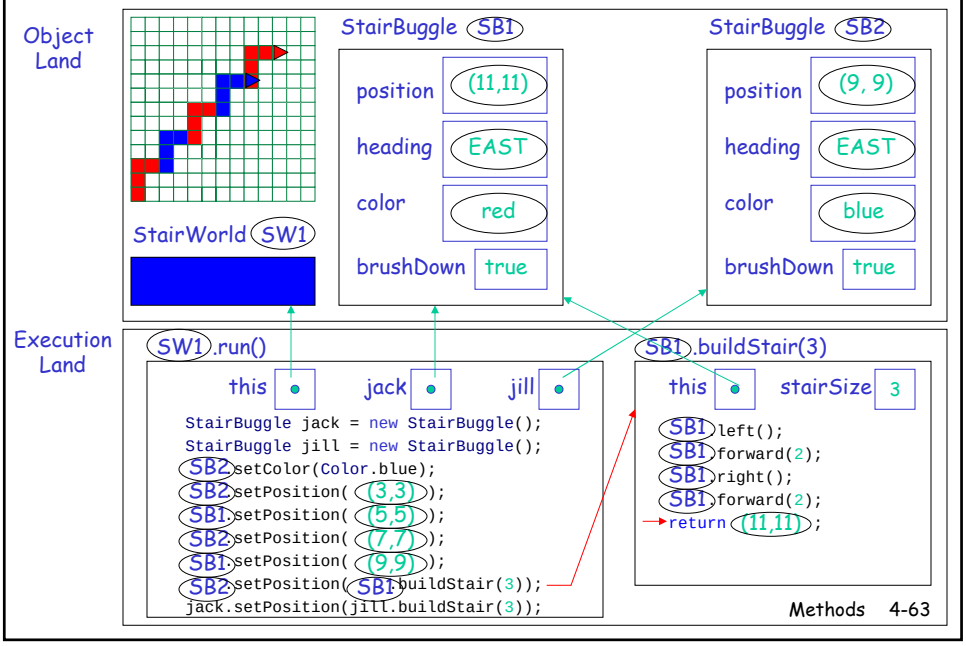
… to Jack's setPosition() invocation



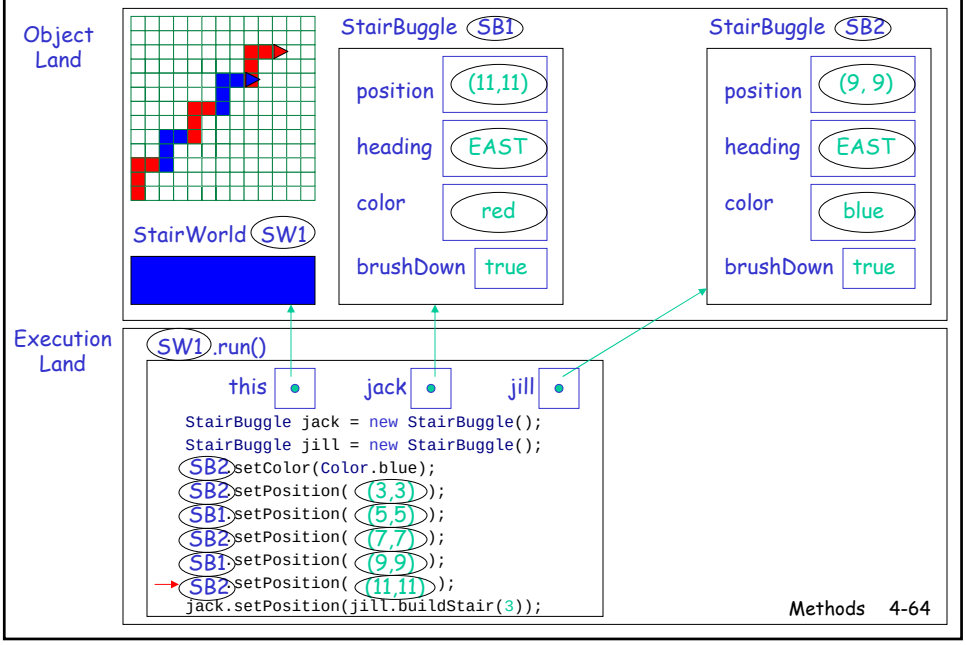
Jack joins Jill at (9,9)



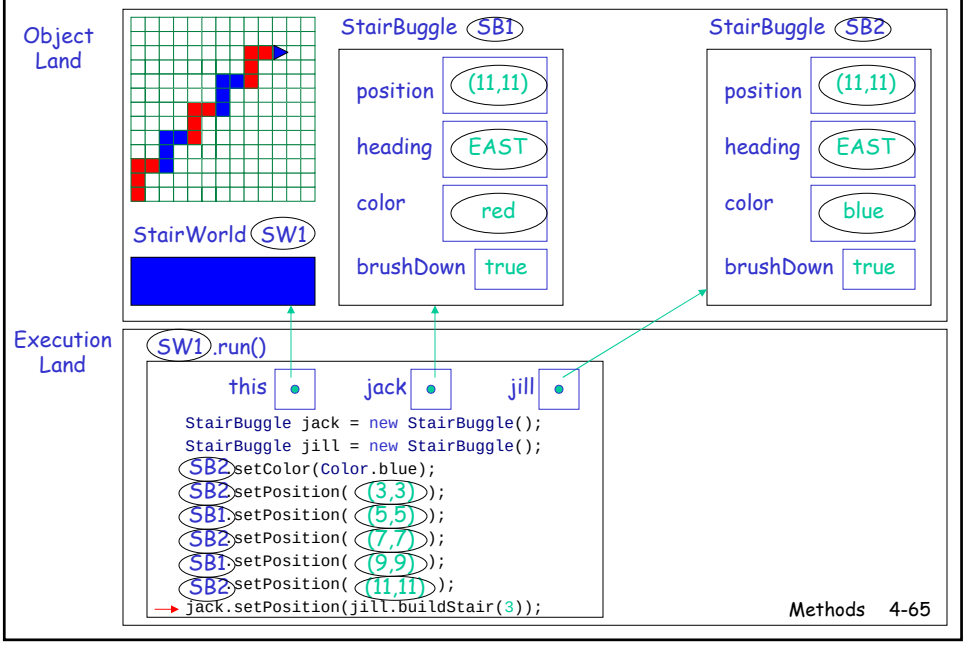
Jack draws the fifth step, and returns his final position



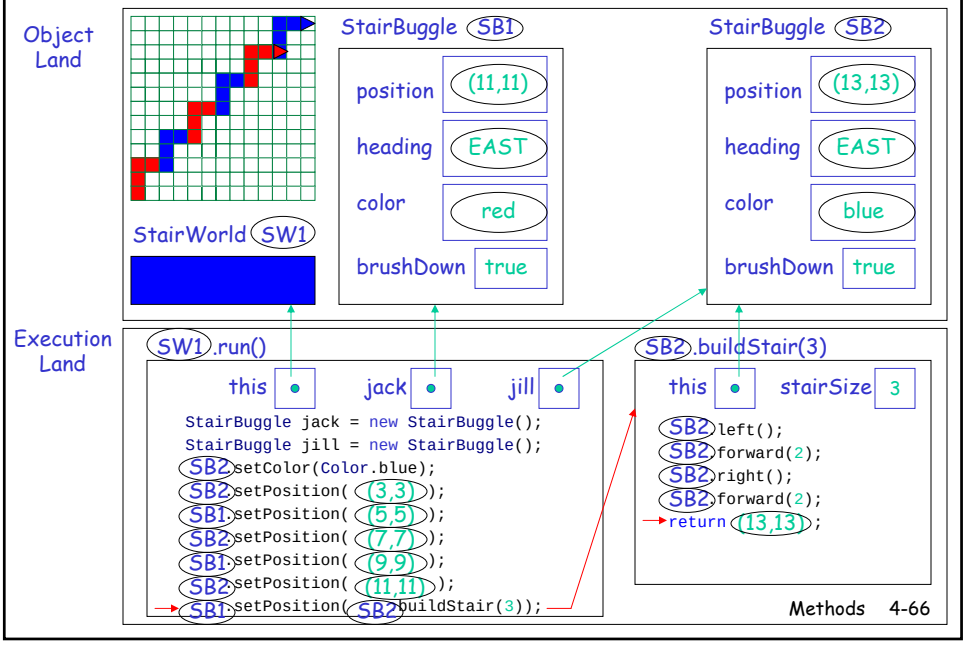
… to Jill's setPosition() invocation



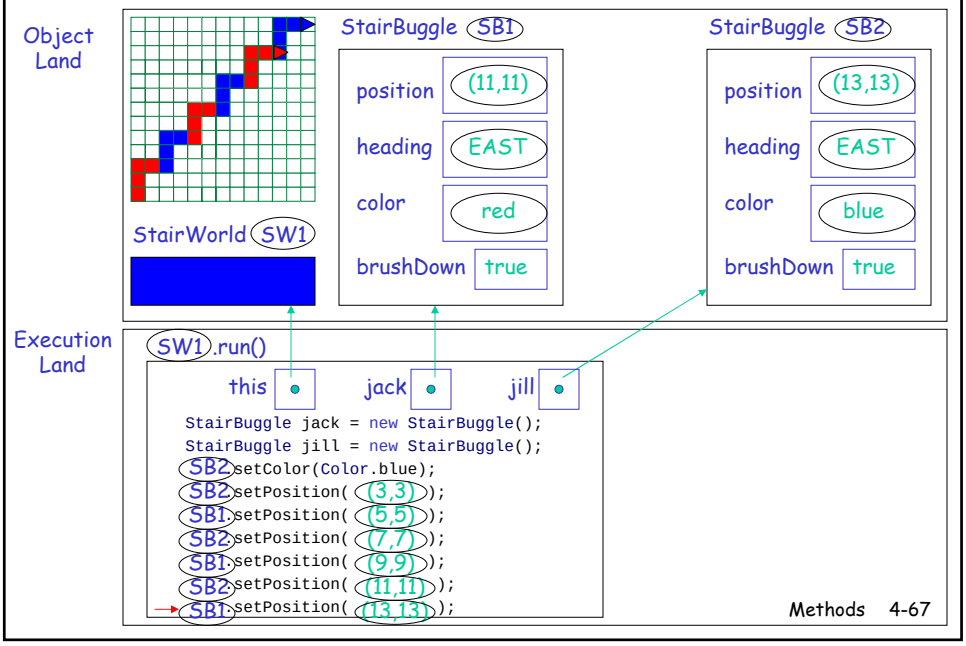
Jill joins Jack at (11,11)



Jill draws the sixth step, and returns her final position …



… to Jack's setPosition() invocation



Jack joins jill at (13,13)