



Control Flow

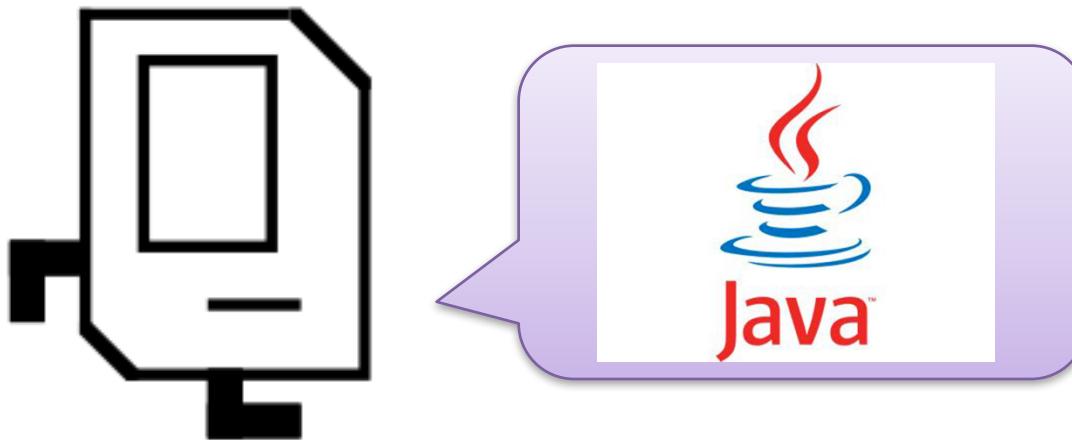
Chris Piech
CS106A, Stanford University

PREVIOUSLY ON

GAME OF THRONES

CS106A

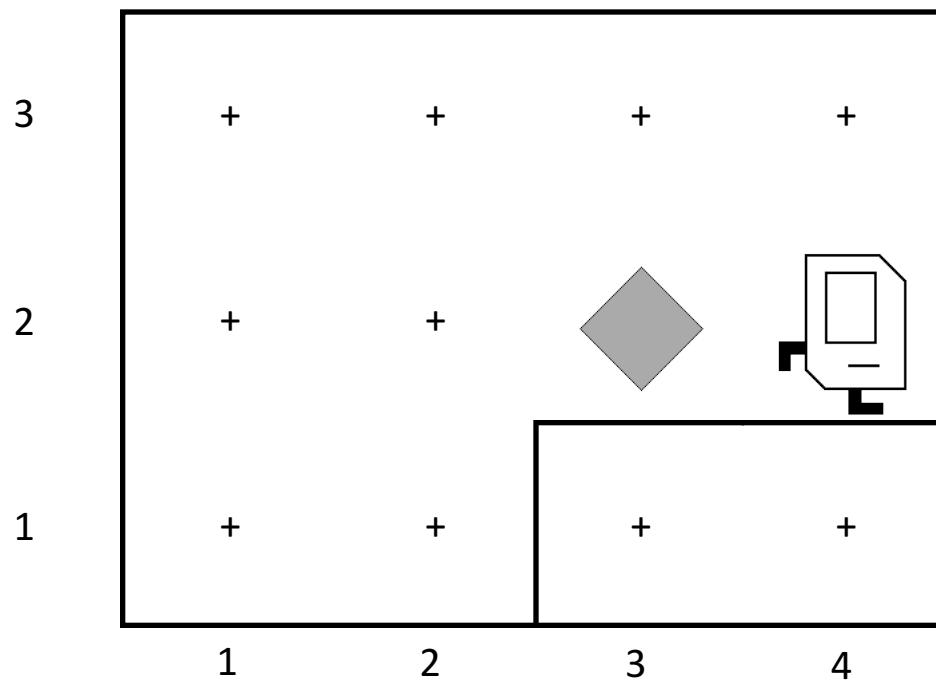
Karel the Robot



- * While Karel is in Java, when you program your Karel assignment we ask that you stick to the concepts in the course reader



First Challenge



Anatomy of a Program

```
import stanford.karel.*;  
  
public class OurKarelProgram extends Karel {  
    public void run() {  
        move();  
        pickBeeper();  
        move();  
        turnLeft();  
        move();  
        turnRight();  
        move();  
        putBeeper();  
        move();  
    }  
  
    private void turnRight() {  
        turnLeft();  
        turnLeft();  
        turnLeft();  
    }  
}
```

This is the program's
source code



Anatomy of a Program

```
import stanford.karel.*;  
  
public class OurKarelProgram extends Karel {  
    public void run() {  
        move();  
        pickBeeper();  
        move();  
        turnLeft();  
        move();  
        turnRight();  
        move();  
        putBeeper();  
        move();  
    }  
  
    private void turnRight() {  
        turnLeft();  
        turnLeft();  
        turnLeft();  
    }  
}
```

This piece of the program's **source code** is called a **method**.



Anatomy of a Program

```
import stanford.karel.*;  
  
public class OurKarelProgram extends Karel {  
    public void run() {  
        move();  
        pickBeeper();  
        move();  
        turnLeft();  
        move();  
        turnRight();  
        move();  
        putBeeper();  
        move();  
    }  
  
    private void turnRight() {  
        turnLeft();  
        turnLeft();  
        turnLeft();  
    }  
}
```

This line of code gives the
name of the method
(here, run)



Anatomy of a Program

```
import stanford.karel.*;  
  
public class OurKarelProgram extends Karel {  
    public void run() {  
        move();  
        pickBeeper();  
        move();  
        turnLeft();  
        move();  
        turnRight();  
        move();  
        putBeeper();  
        move();  
    }  
  
    private void turnRight() {  
        turnLeft();  
        turnLeft();  
        turnLeft();  
    }  
}
```

This line of code gives the
name of the method
(here, turnRight)



Anatomy of a Program

```
import stanford.karel.*;  
  
public class OurKarelProgram extends Karel {  
    public void run() {  
        move();  
        pickBeeper();  
        move();  
        turnLeft();  
        move();  
        turnRight();  
        move();  
        putBeeper();  
        move();  
    }  
  
    private void turnRight() {  
        turnLeft();  
        turnLeft();  
        turnLeft();  
    }  
}
```

This is called an ***import statement***. It tells Java what Karel is.



Anatomy of a Program

```
import stanford.karel.*;  
  
public class OurKarelProgram extends Karel {  
    public void run() {  
        move();  
        pickBeeper();  
        move();  
        turnLeft();  
        move();  
        turnRight();  
        move();  
        putBeeper();  
        move();  
    }  
  
    private void turnRight() {  
        turnLeft();  
        turnLeft();  
        turnLeft();  
    }  
}
```

This is called a
code block



Anatomy of a Program

```
import stanford.karel.*;  
  
public class OurKarelProgram extends Karel {  
    public void run() {  
        move();  
        pickBeeper();  
        move();  
        turnLeft();  
        move();  
        turnRight();  
        move();  
        putBeeper();  
        move();  
    }  
  
    private void turnRight() {  
        turnLeft();  
        turnLeft();  
        turnLeft();  
    }  
}
```



Anatomy of a Program

```
import stanford.karel.*;

public class OurKarelProgram extends Karel {
    public void run() {
        move();
        pickBeeper();
        move();
        turnLeft();
        move();
        turnRight();
        move();
        putBeeper();
        move();
    }

    private void turnRight() {
        turnLeft();
        turnLeft();
        turnLeft();
    }
}
```



Anatomy of a Program

```
import stanford.karel.*;

public class OurKarelProgram extends Karel {
    public void run() {
        move();
        pickBeeper();
        move();
        turnLeft();
        move();
        turnRight();
        move();
        putBeeper();
        move();
    }

    private void turnRight() {
        turnLeft();
        turnLeft();
        turnLeft();
    }
}
```



Anatomy of a Program

```
import stanford.karel.*;  
  
public class OurKarelProgram extends Karel {  
    public void run() {  
        move();  
        pickBeeper();  
        move();  
        turnLeft();  
        move();  
        turnRight();  
        move();  
        putBeeper();  
        move();  
    }  
  
    private void turnRight() {  
        turnLeft();  
        turnLeft();  
        turnLeft();  
    }  
}
```



Anatomy of a Program

```
import stanford.karel.*;  
  
public class OurKarelProgram extends Karel {  
    public void run() {  
        move();  
        pickBeeper();  
        move();  
        turnLeft();  
        move();  
        turnRight();  
        move();  
        putBeeper();  
        move();  
    }  
  
    private void turnRight() {  
        turnLeft();  
        turnLeft();  
        turnLeft();  
    }  
}
```

The run method is “public” so that Eclipse can call it.



Anatomy of a Program

```
import stanford.karel.*;  
  
public class OurKarelProgram extends Karel {  
    public void run() {  
        move();  
        pickBeeper();  
        move();  
        turnLeft();  
        move();  
        turnRight();  
        move();  
        putBeeper();  
        move();  
    }  
  
    private void turnRight() {  
        turnLeft();  
        turnLeft();  
        turnLeft();  
    }  
}
```

The turnRight method is “private” to indicate it is only visible to our current program.





Piech, CS106A, Stanford University



Method Definition

```
private void name() {  
    statements in the method body  
}
```

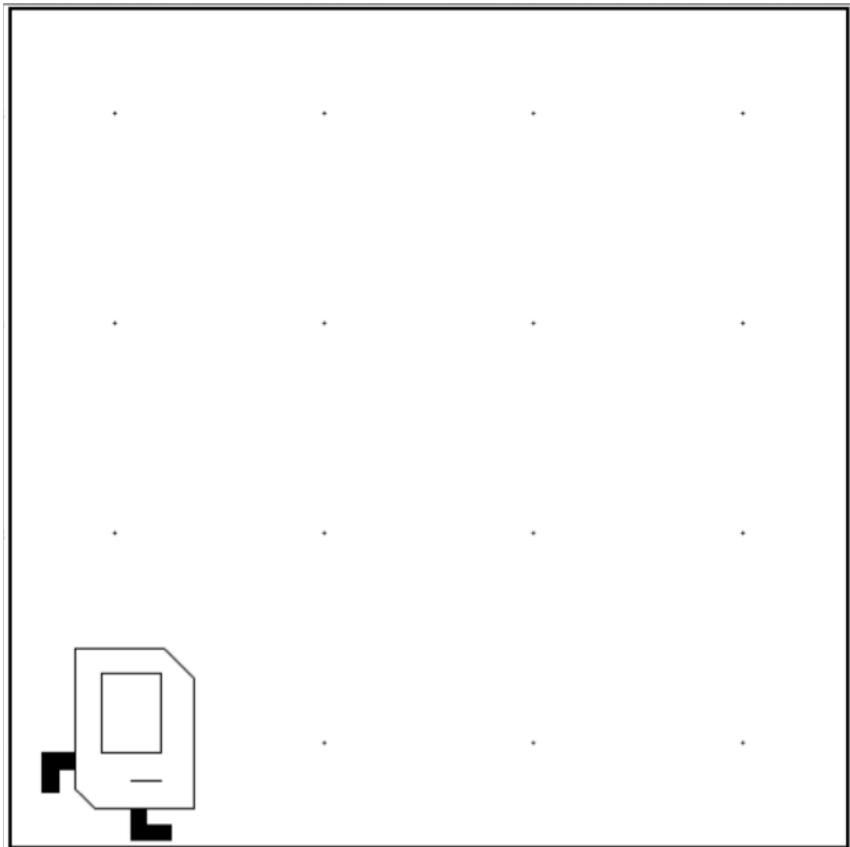
This adds a new
command to Karel's
vocabulary



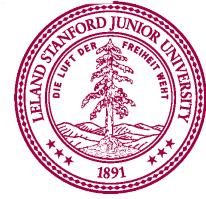
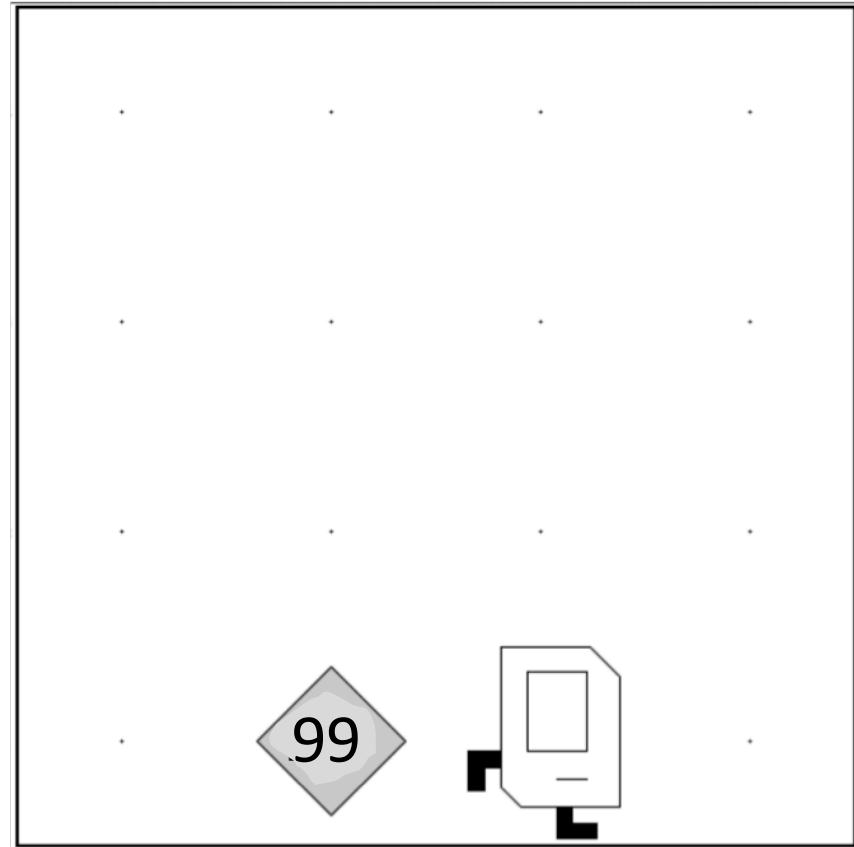
For loops,
While loops,
If/Else statements

Place 99 beepers?

Before



After



Place 99 beepers

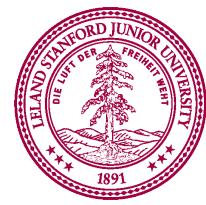
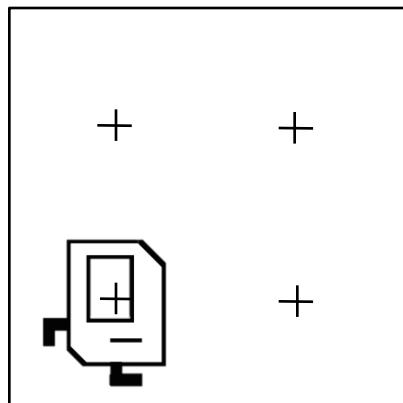
```
public class Place99Beepers extends SuperKarel {  
    public void run() {  
        move();  
        for(int i = 0; i < 99; i++) {  
            putBeeper();  
        }  
        move();  
    }  
}
```

This “for loop” repeats the code in its
“body” 99 times



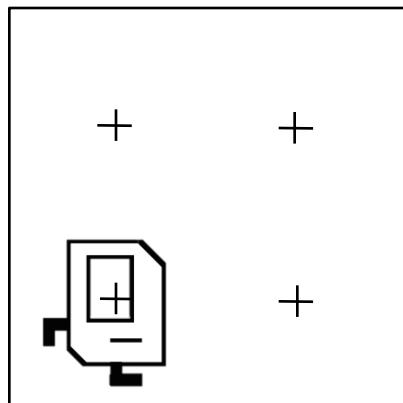
Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

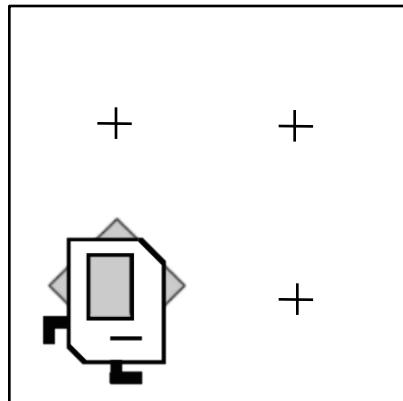


First time
through the
loop

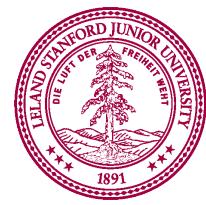


Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

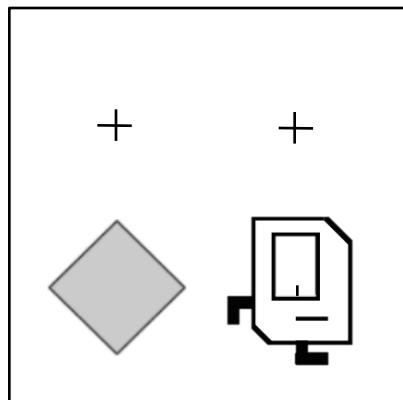


First time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeepers();  
            move();  
            turnLeft();  
        }  
    }  
}
```

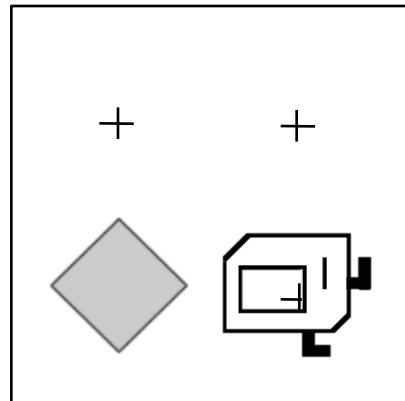


First time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

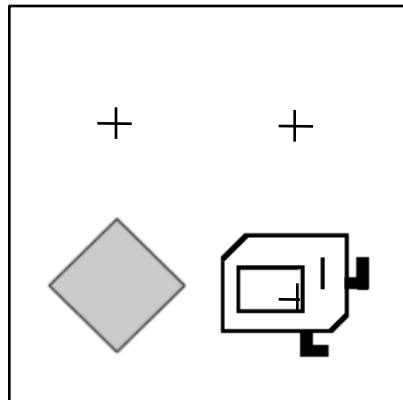


First time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

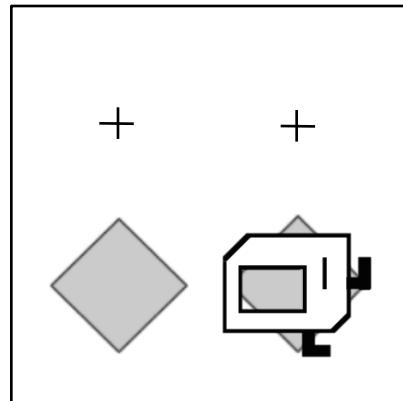


Second time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

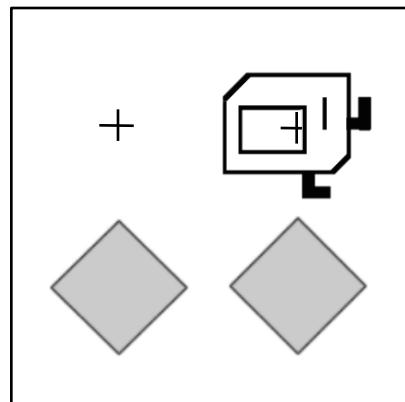


Second time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```



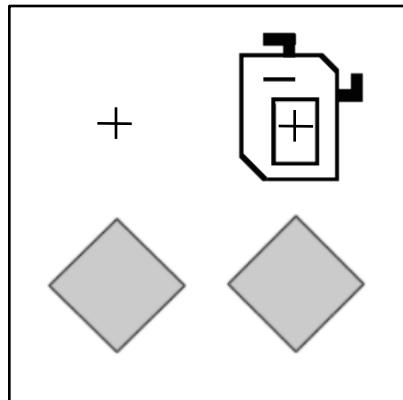
Second time
through the
loop



Place Beeper Square

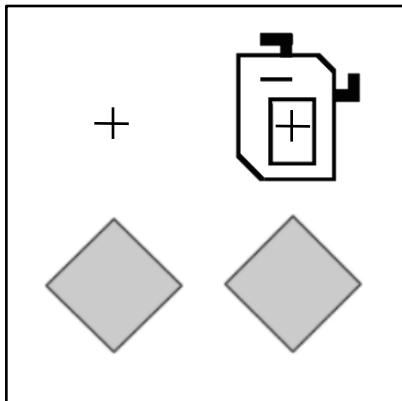
```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

Second time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

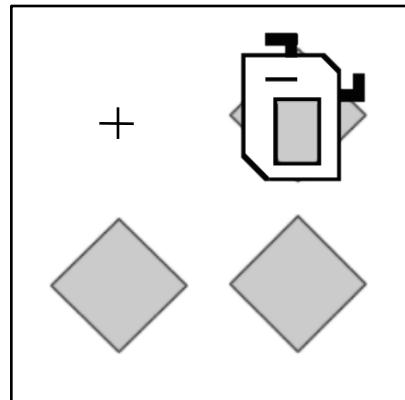


Third time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

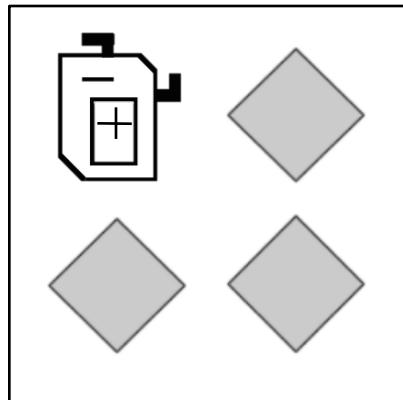


Third time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

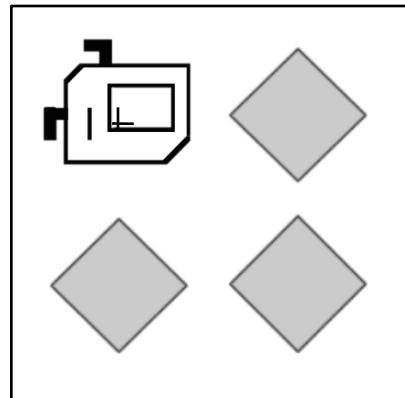


Third time
through the
loop

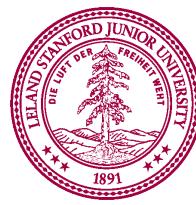


Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

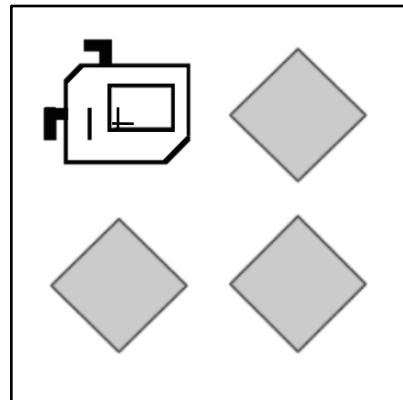


Third time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

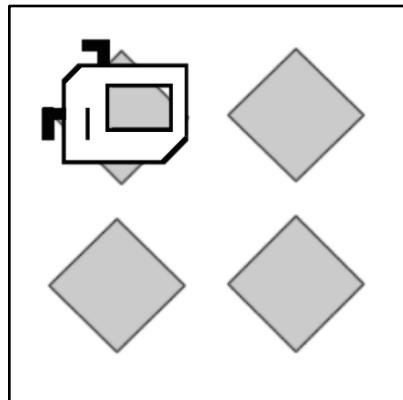


Fourth time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

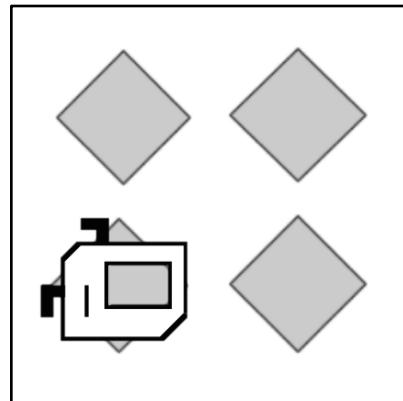


Fourth time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeepers();  
            move();  
            turnLeft();  
        }  
    }  
}
```

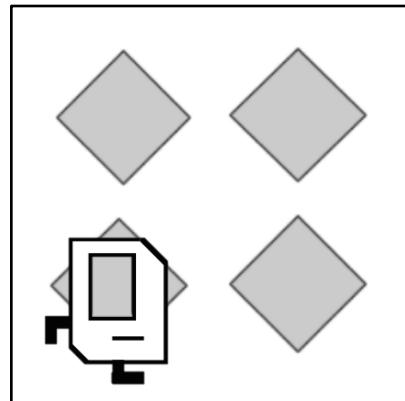


Fourth time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```

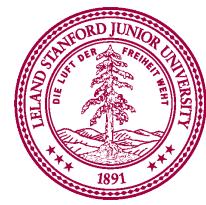
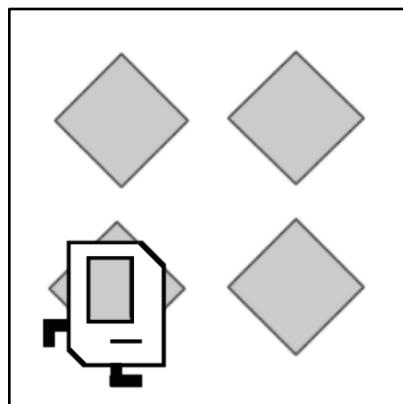


Fourth time
through the
loop



Place Beeper Square

```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```



Exciting!

Aside: Super Karel

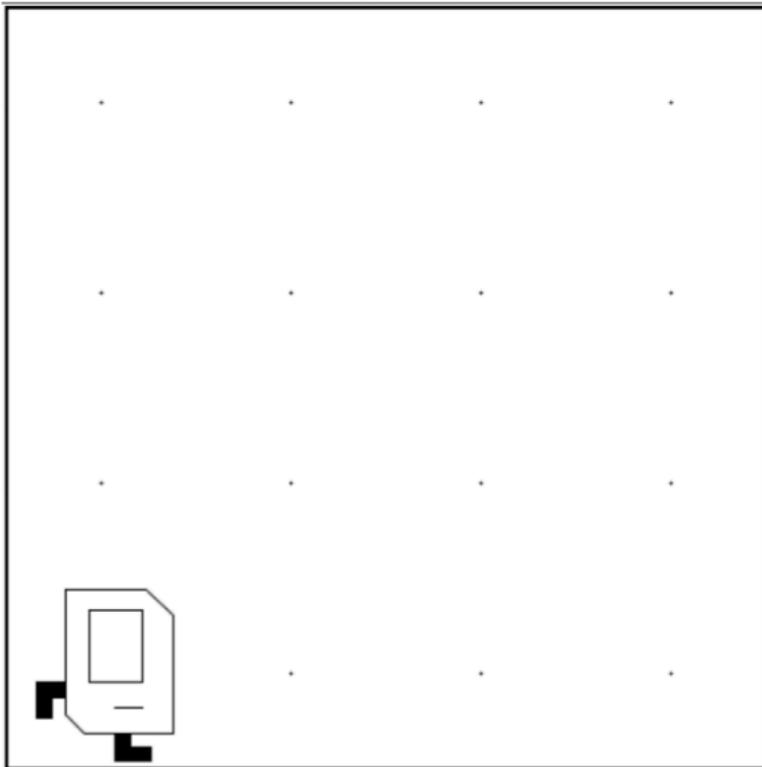
```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // super karel has a few more commands  
        turnRight();  
        turnAround();  
        paintCorner(BLUE);  
  
        putBeeper();  
        move();  
    }  
  
}
```



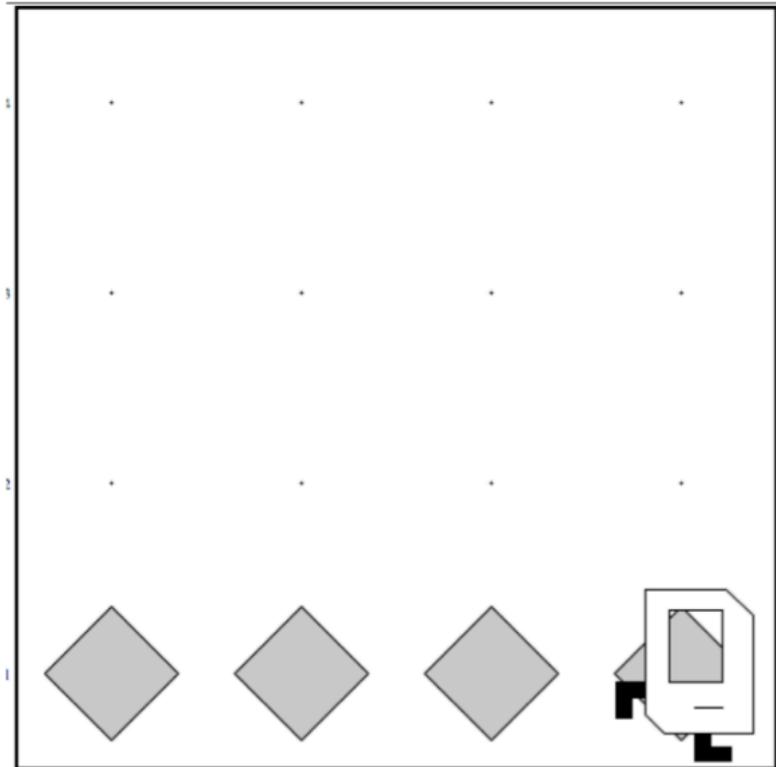
Next task

Place Beeper Line

Before



After



Place Beeper Line

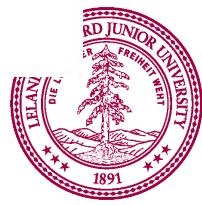
```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
        }  
    }  
}
```





Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
        }  
    }  
}
```

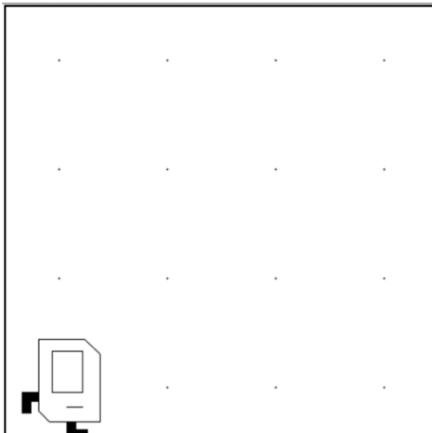




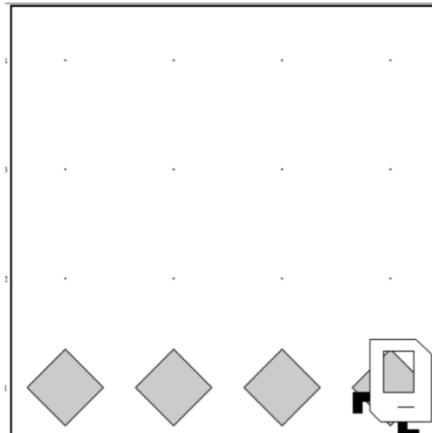
Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
        }  
    }  
}  
}  
What we want
```

Before

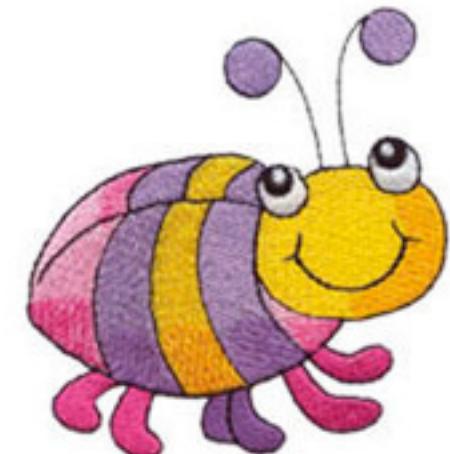
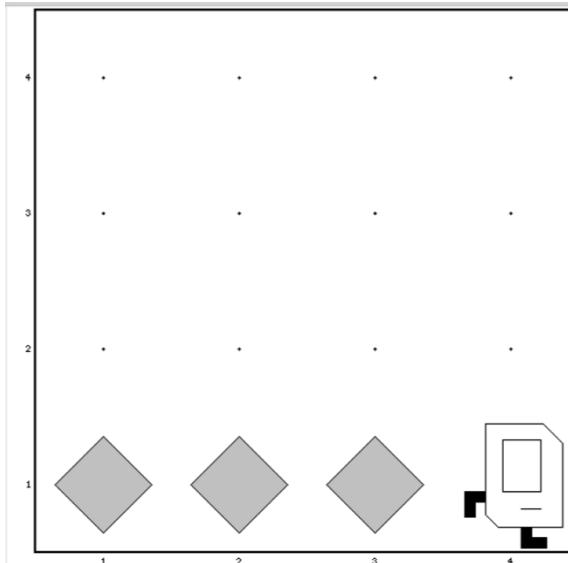


After



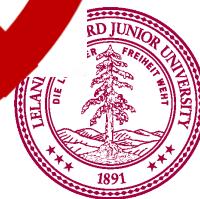
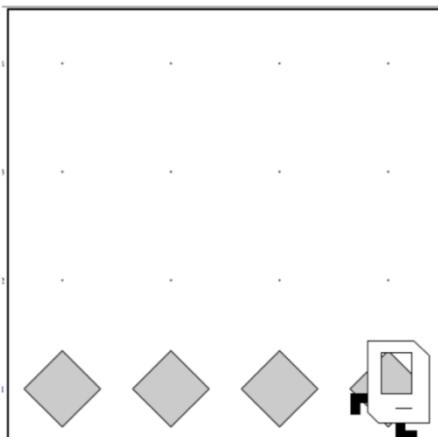
Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        for(int i = 0; i < 3; i++) {  
            putBeeper();  
            move();  
        }  
    }  
}
```



Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        for(int i = 0; i < 3; i++) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}
```



Actual Bug from Marc II

1100 Started Cosine Tape (Sine -
1525 Started Multi Adder Test.
 Relays changed
 in relay

1545



Relay #70
(moth) in re

First actual case of bug

~~1630~~ 1630 Antilog start.

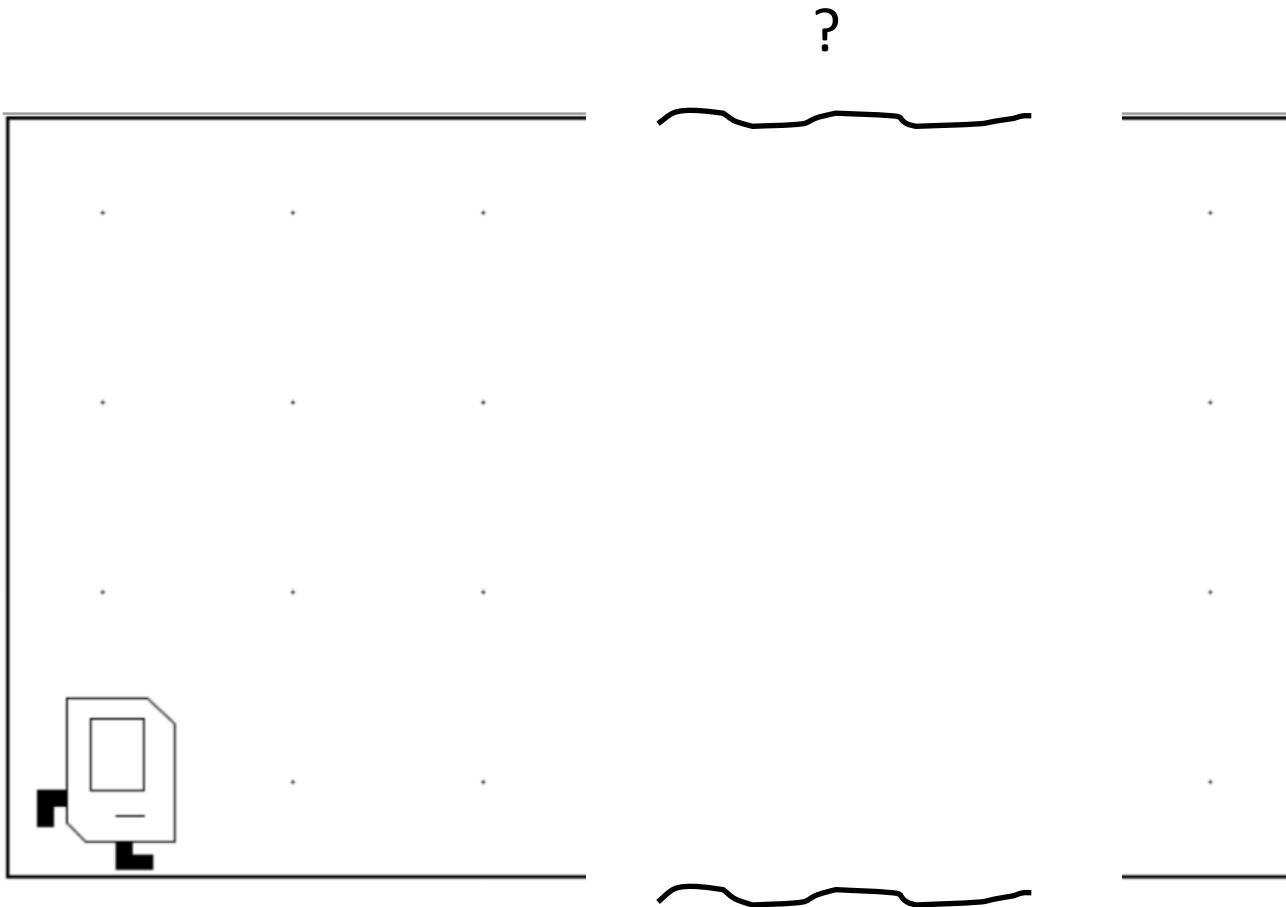
1700 closed down.



Grace Hopper

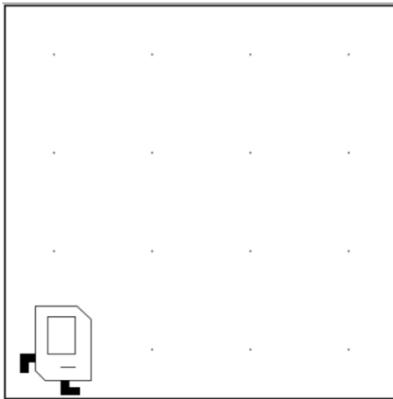


Don't Know World Size

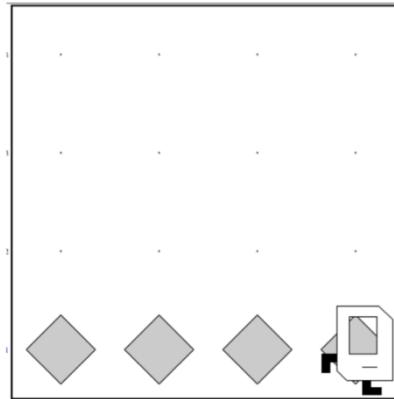


Work in Any World

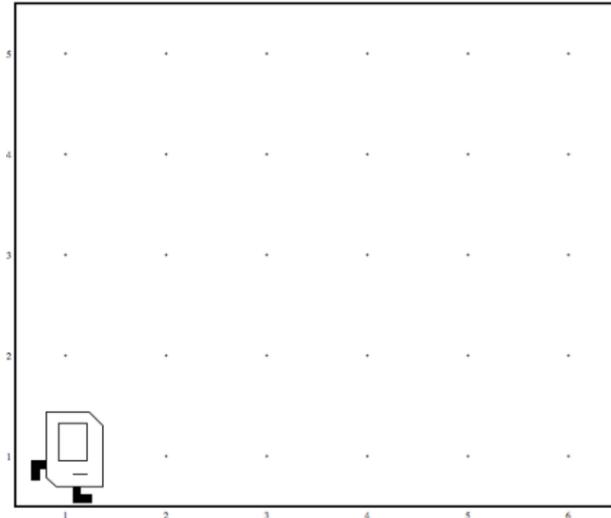
Before



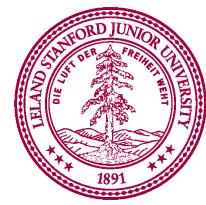
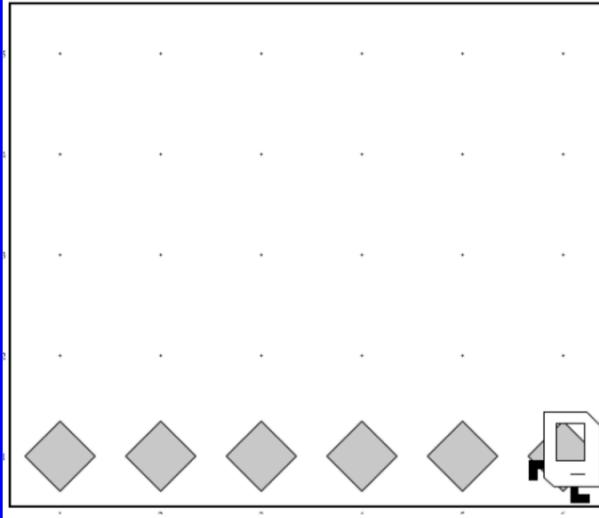
After



Before



After



While Loop

While Loop

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(condition) {  
            code to repeat  
        }  
    }  
}
```



Place Beeper Line

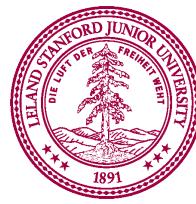
```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

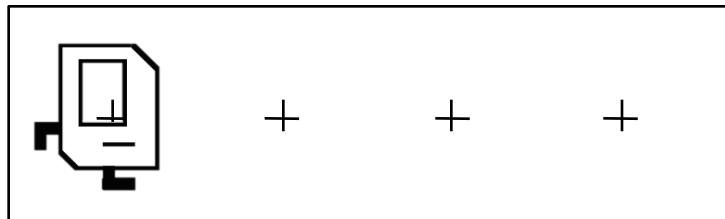
        // example while loop
        while(frontIsClear()) {
            move();
        }
        // extra put beeper
        putBeepers();
    }

}
```



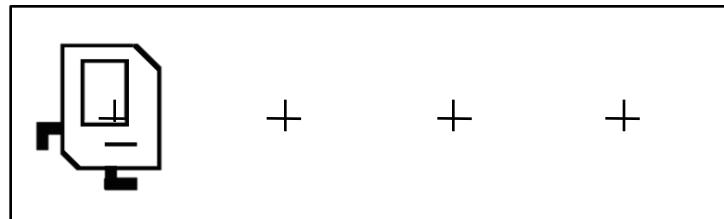
Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}  
}
```



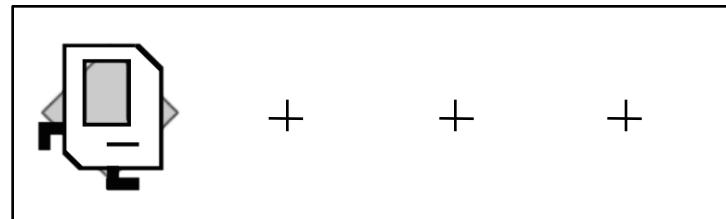
Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}  
}
```



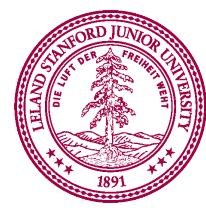
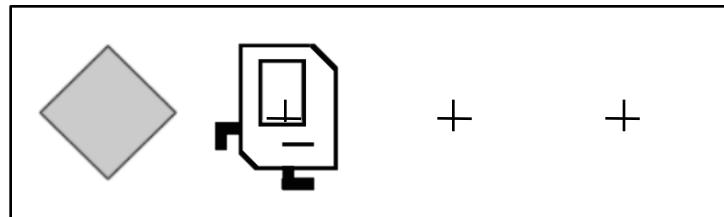
Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}
```



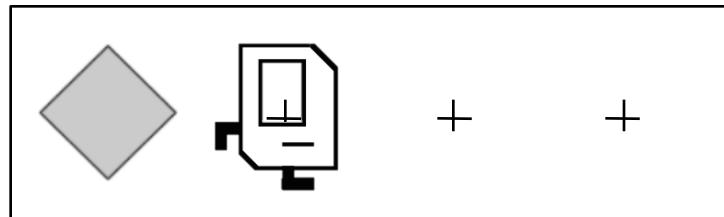
Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}
```



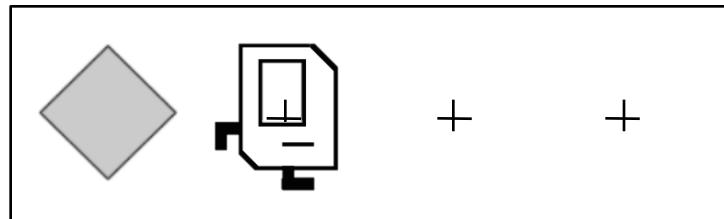
Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}
```



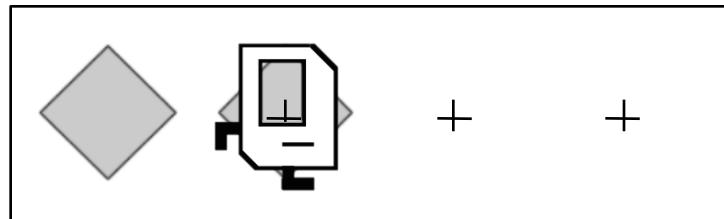
Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}  
}
```



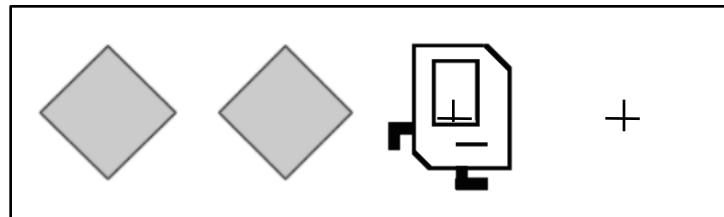
Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}  
}
```



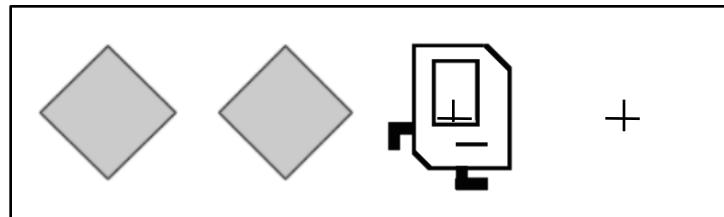
Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}
```



Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}
```



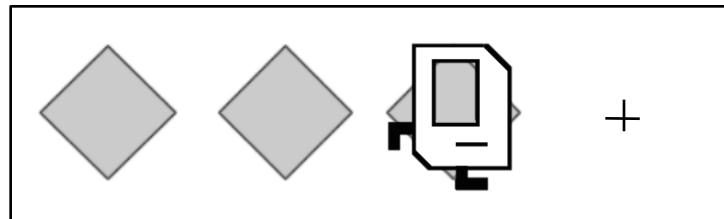
Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

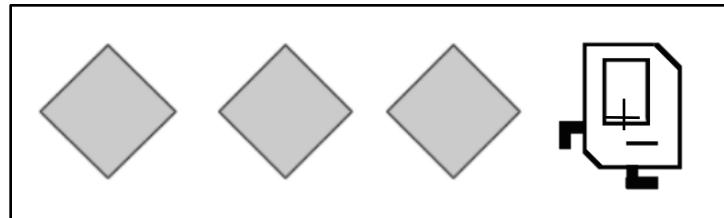
    public void run() {

        // example while loop
        while(frontIsClear()) {
            putBeeper();
            move();
        }
        // extra put beeper
        putBeeper();
    }
}
```



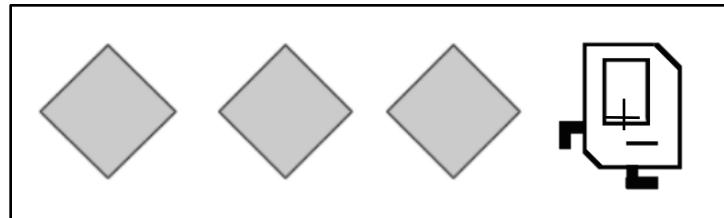
Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}
```



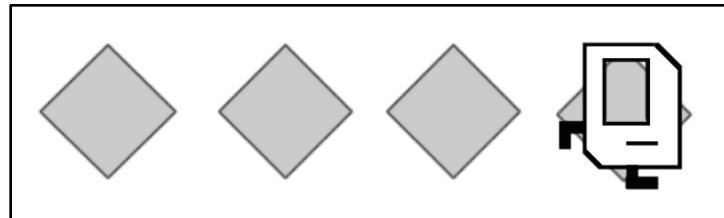
Place Beeper Line

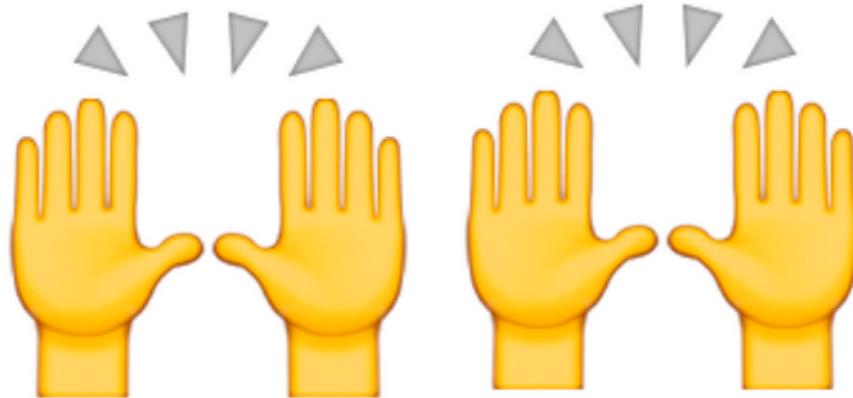
```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}  
}
```



Place Beeper Line

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
  
        // example while loop  
        while(frontIsClear()) {  
            putBeeper();  
            move();  
        }  
        // extra put beeper  
        putBeeper();  
    }  
}
```





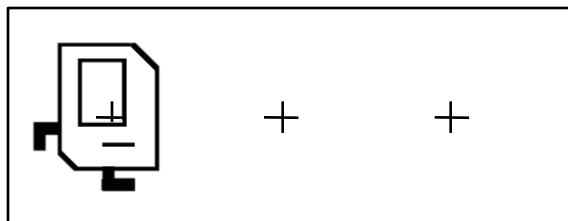
Piech, CS106A, Stanford University



Common misconception:

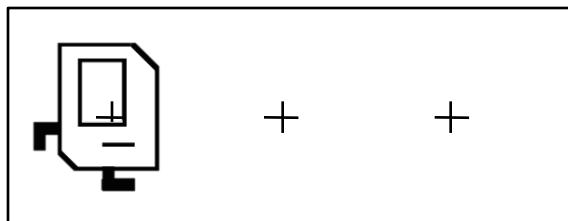
Place Beeper Line: Redux

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```



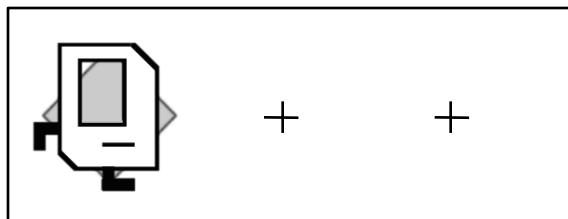
Place Beeper Line: Redux

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```



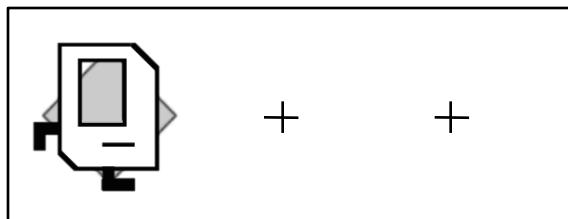
Place Beeper Line: Redux

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```



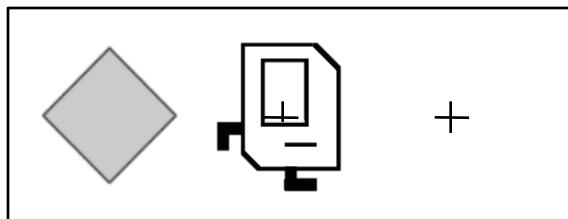
Place Beeper Line: Redux

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```



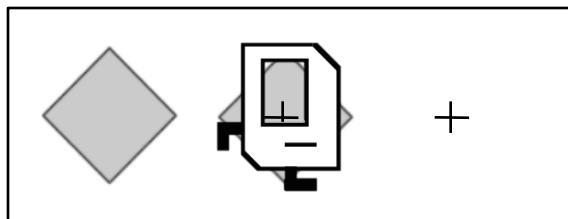
Place Beeper Line: Redux

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```



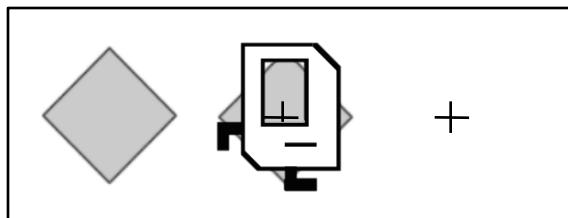
Place Beeper Line: Redux

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```



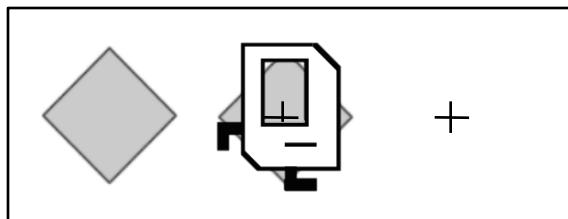
Place Beeper Line: Redux

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```



Place Beeper Line: Redux

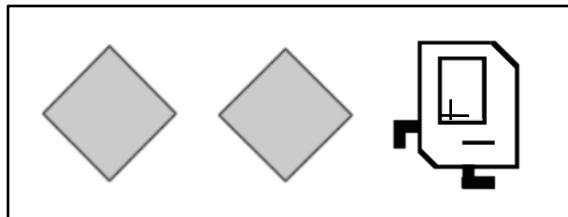
```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```



Place Beeper Line: Redux

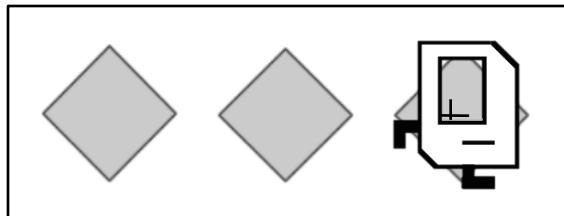
```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```

This is
incredibly
important!



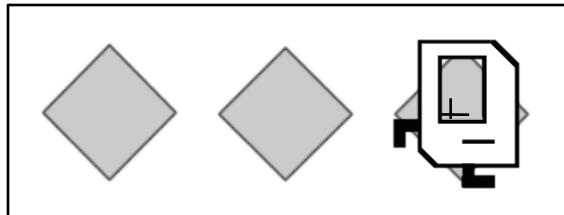
Place Beeper Line: Redux

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```



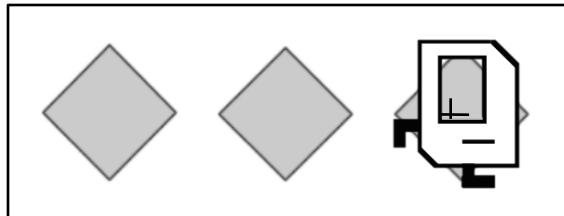
Place Beeper Line: Redux

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```



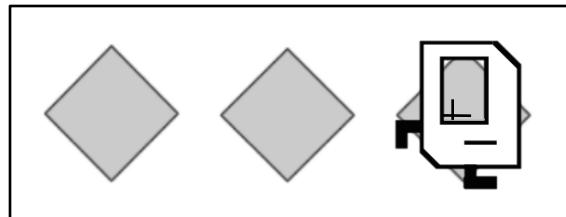
Place Beeper Line: Redux

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```

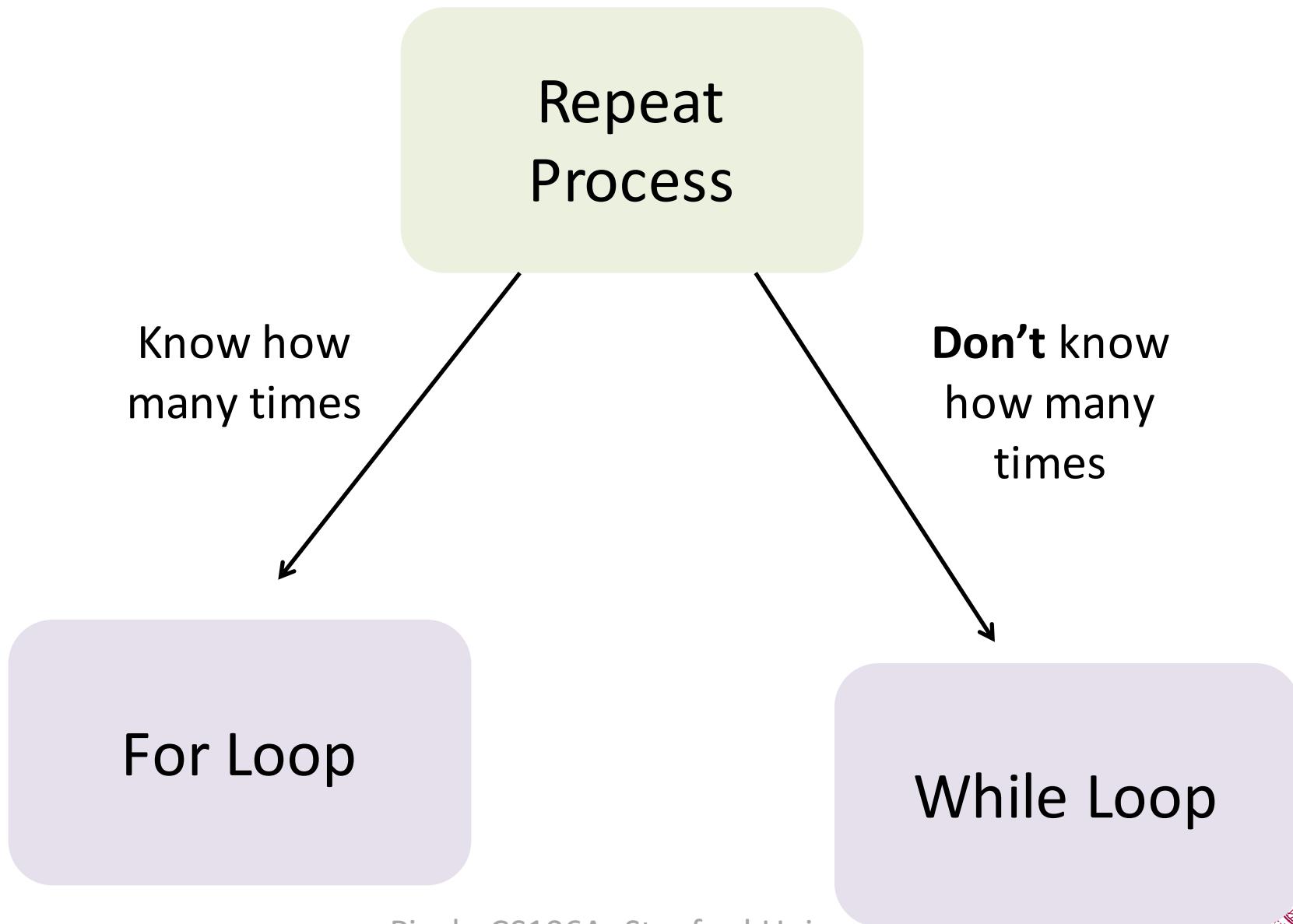


Place Beeper Line: Redux

```
import stanford.karel.*;  
  
public class BeeperLine extends SuperKarel {  
  
    public void run() {  
        // place a first beeper  
        putBeeper();  
  
        // example while loop  
        while(frontIsClear()) {  
            move();  
            putBeeper();  
        }  
    }  
}
```



Which Loop



What if you only want to repeat
one time?

If statement

If Statement

```
import stanford.karel.*;  
  
public class IfExample extends SuperKarel {  
  
    public void run() {  
  
        // example of an if statement  
        if(condition) {  
            code to run if condition is true  
        }  
  
    }  
  
}
```



If Statement

```
import stanford.karel.*;  
  
public class IfExample extends Pretend{  
  
    public void run() {  
  
        // example of an if statement  
        if(youLikeBeyonce()) {  
            makeSomeNoise();  
        }  
    }  
}
```



If Statement

```
import stanford.karel.*;  
  
public class IfExample extends SuperKarel{  
  
    public void run() {  
        safeMove();  
    }  
  
    private void safeMove() {  
        if(frontIsClear()) {  
            move();  
        }  
    }  
}  
}
```



If / Else Statement

```
import stanford.karel.*;  
  
public class IfExample extends SuperKarel{  
  
    public void run() {  
        invertBeeper();  
    }  
  
    private void invertBeeper() {  
        if(beepersPresent()) {  
            pickBeeper();  
        } else {  
            putBeeper();  
        }  
    }  
}  
}
```



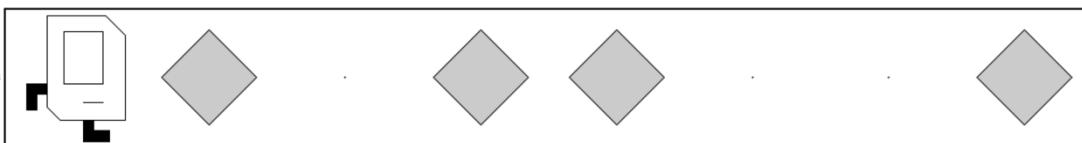
The Full Karel

<p>Built-in Karel commands:</p> <pre>move(); turnLeft(); putBeeper(); pickBeeper();</pre>	<p>Conditional statements:</p> <pre>if (condition) { statements executed if condition is true } if (condition) { statements executed if condition is true } else { statements executed if condition is false }</pre>																		
<p>Karel program structure:</p> <pre>/* * Comments may be included anywhere in * the program between a slash-star and * the corresponding star-slash characters. */ import stanford.karel.*; /* Definition of the new class */ public class name extends Karel { public void run() { statements in the body of the method } definitions of private methods }</pre>	<p>Iterative statements:</p> <pre>for (int i = 0; i < count; i++) { statements to be repeated } while (condition) { statements to be repeated }</pre>																		
<p>Karel condition names:</p> <table> <tbody> <tr> <td>frontIsClear()</td> <td>frontIsBlocked()</td> </tr> <tr> <td>leftIsClear()</td> <td>leftIsBlocked()</td> </tr> <tr> <td>rightIsClear()</td> <td>rightIsBlocked()</td> </tr> <tr> <td>beepersPresent()</td> <td>noBeepersPresent()</td> </tr> <tr> <td>beepersInBag()</td> <td>noBeepersInBag()</td> </tr> <tr> <td>facingNorth()</td> <td>notFacingNorth()</td> </tr> <tr> <td>facingEast()</td> <td>notFacingEast()</td> </tr> <tr> <td>facingSouth()</td> <td>notFacingSouth()</td> </tr> <tr> <td>facingWest()</td> <td>notFacingWest()</td> </tr> </tbody> </table>	frontIsClear()	frontIsBlocked()	leftIsClear()	leftIsBlocked()	rightIsClear()	rightIsBlocked()	beepersPresent()	noBeepersPresent()	beepersInBag()	noBeepersInBag()	facingNorth()	notFacingNorth()	facingEast()	notFacingEast()	facingSouth()	notFacingSouth()	facingWest()	notFacingWest()	<p>New commands in the SuperKarel class:</p> <pre>turnRight(); turnAround(); paintCorner(color);</pre> <p>New conditions in the SuperKarel class:</p> <pre>random() random(p) cornerColorIs(color)</pre>
frontIsClear()	frontIsBlocked()																		
leftIsClear()	leftIsBlocked()																		
rightIsClear()	rightIsBlocked()																		
beepersPresent()	noBeepersPresent()																		
beepersInBag()	noBeepersInBag()																		
facingNorth()	notFacingNorth()																		
facingEast()	notFacingEast()																		
facingSouth()	notFacingSouth()																		
facingWest()	notFacingWest()																		



Random Painter

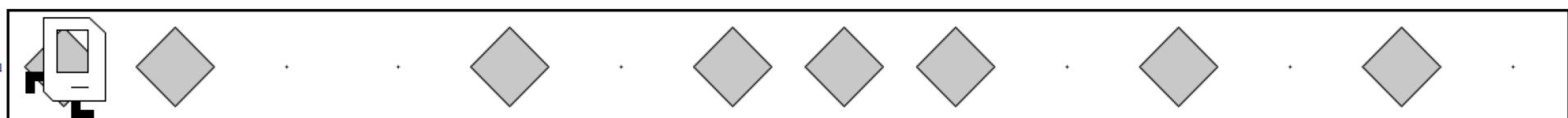
Before:



After:



Before:



After:



You just learned most of
programming “control flow”