



# Control Flow

Chris Piech

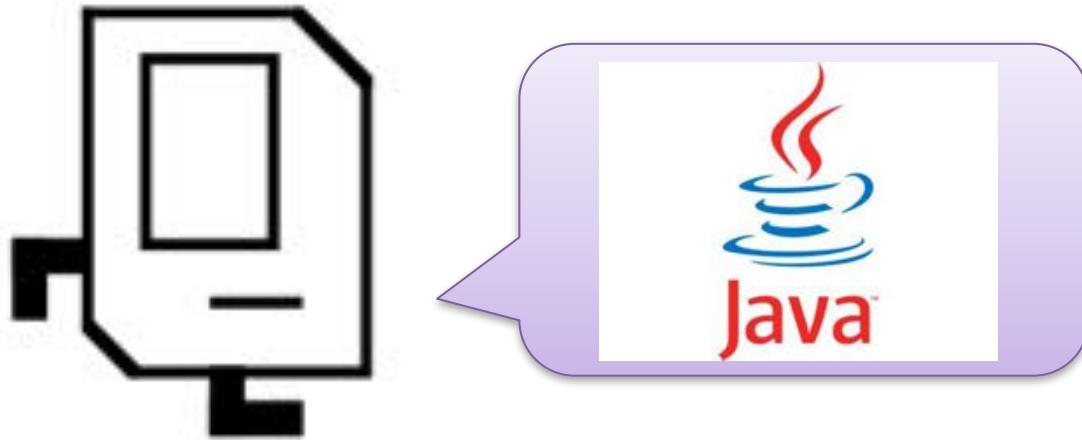
CS106A, Stanford University

PREVIOUSLY ON

GAME OF THRONES

**CS106A**

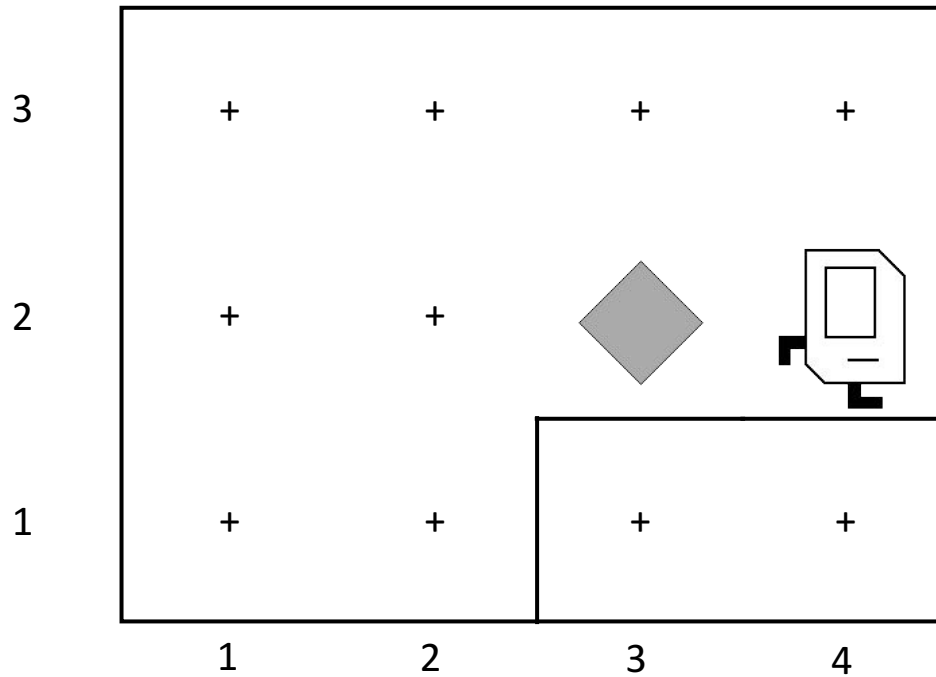
# Karel the Robot



- \* 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();  
    }  
}
```

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

```
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();  
    }  
}
```

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

```
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();  
    }  
}
```

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

```
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();
    }
}
```

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();  
    }  
}
```

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

```
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();  
    }  
}
```

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

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

```
}
```





# Method Definition



This adds a new  
command to Karels  
vocabulary

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

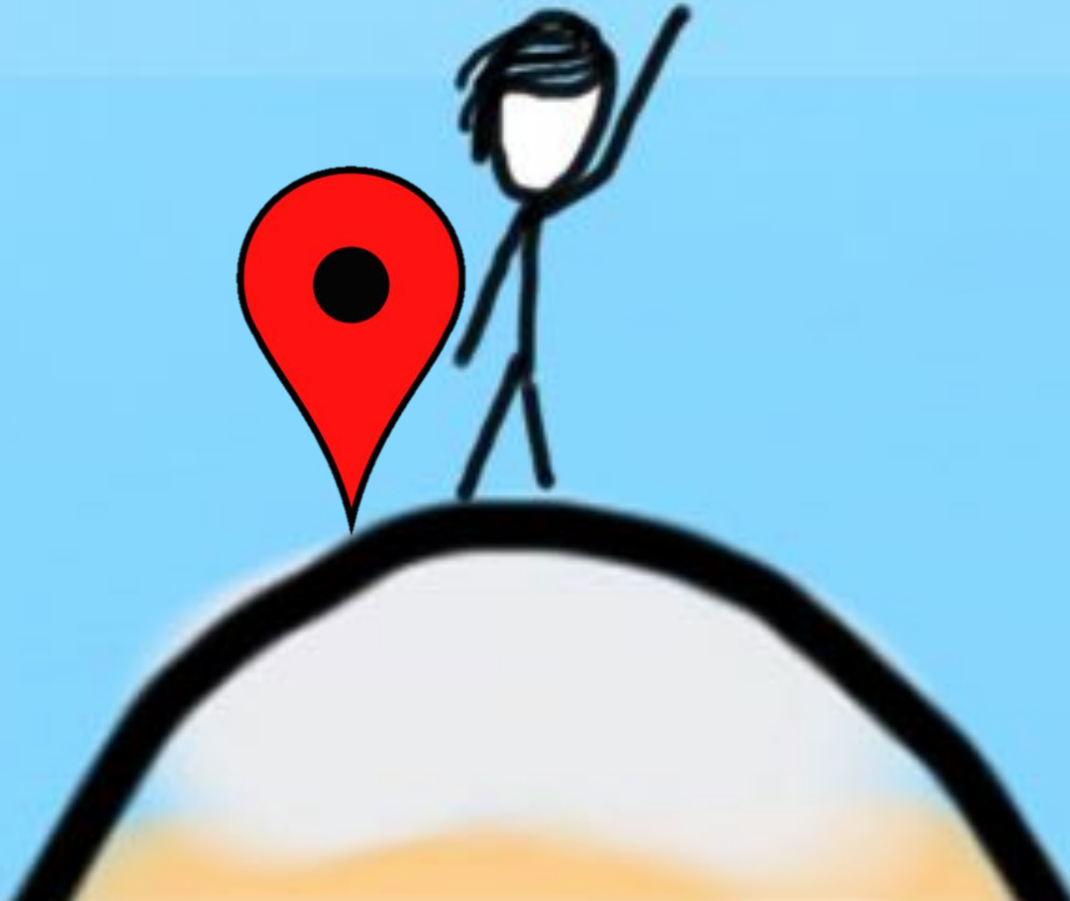


# Decomposition of Hard Tasks

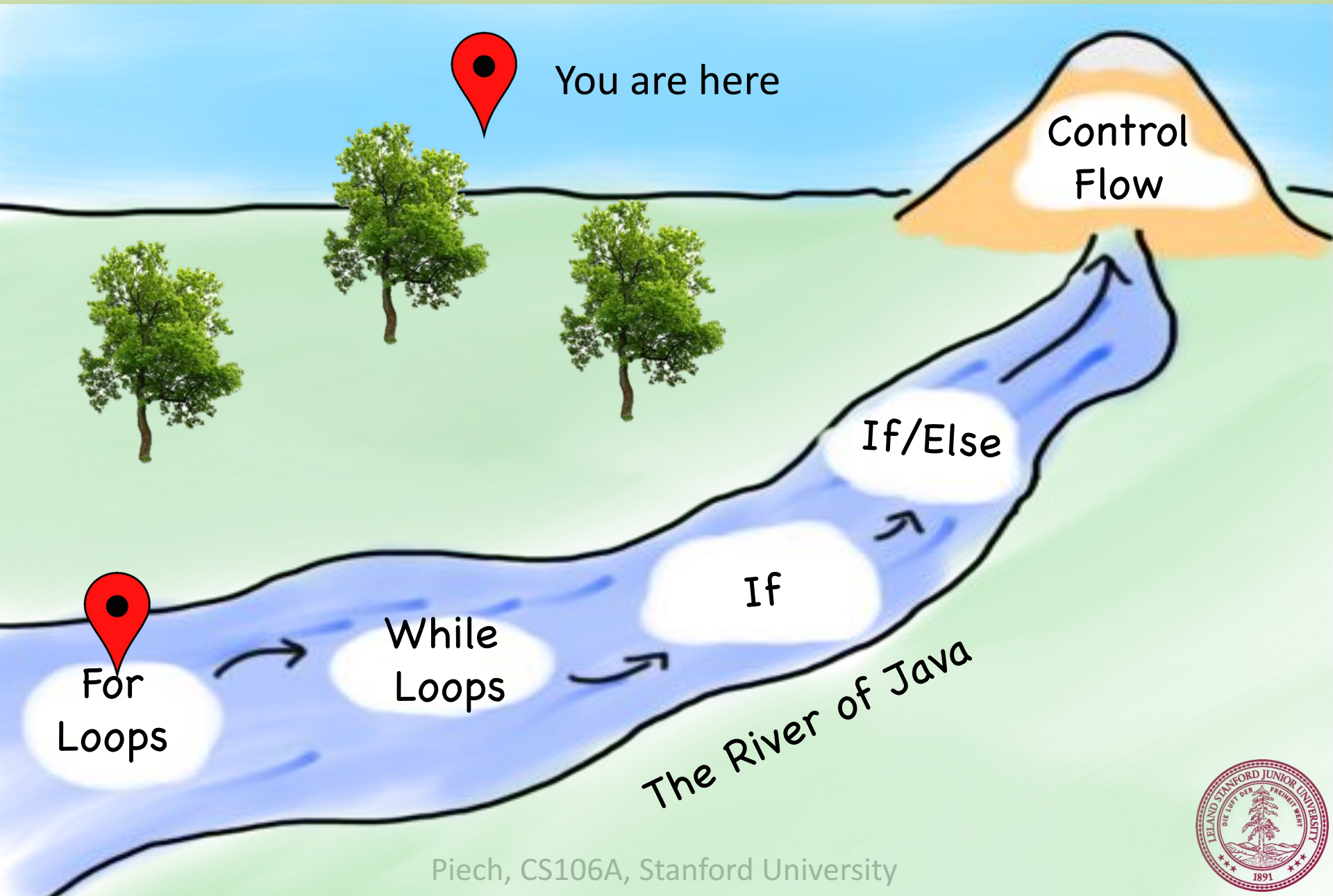


# Today's Goal

1. Code using loops and conditions
2. Trace programs that use loops and conditions



# Today's Route



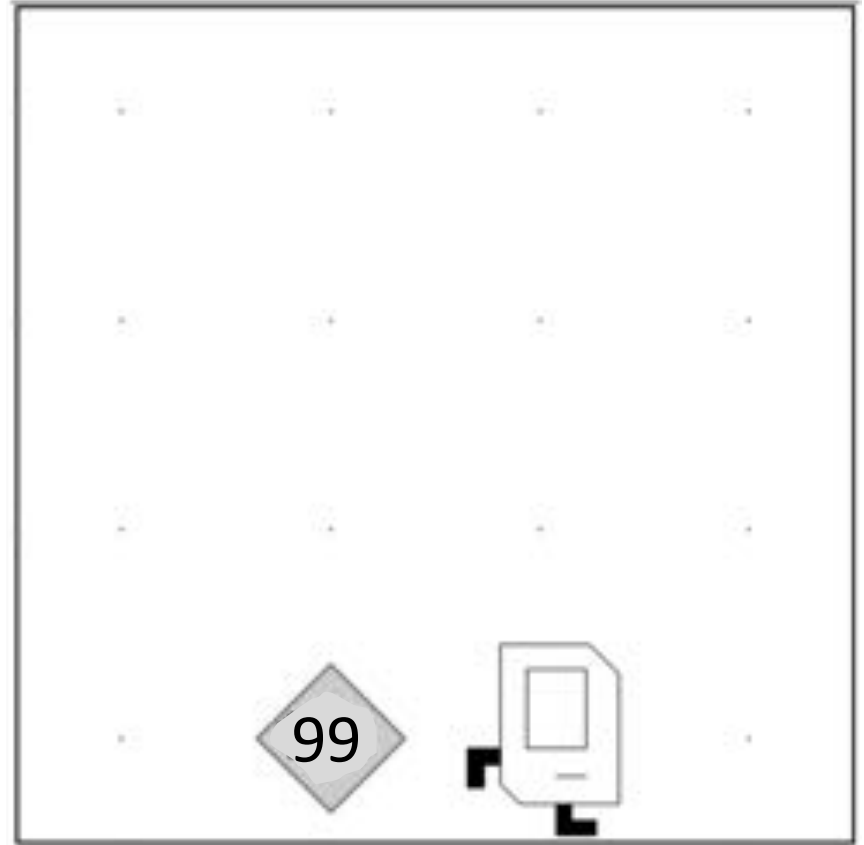
For loops,  
While loops,  
If/Else statements

# Place 99 beepers?

Before



After



# Place 99 beepers

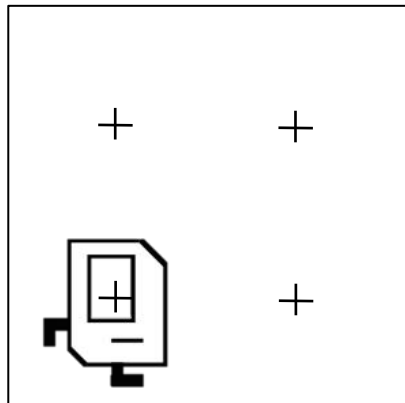
```
public class Place99Beepers extends SuperKarel {  
    public void run() {  
        move();  
        for(int i = 0; i < 99; i++) {  
            putBeeper();  
        }  
        move();  
    }  
}
```

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



# Place Beeper Square

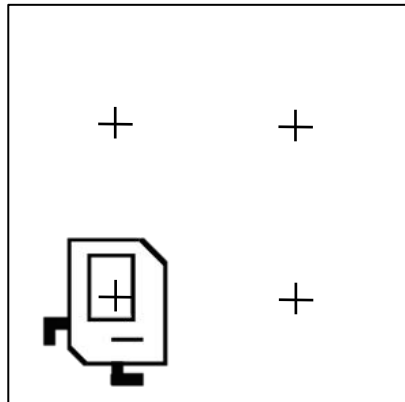
```
public class BeeperSquare extends SuperKarel {  
    public void run() {  
        for(int i = 0; i < 4; i++) {  
            putBeeper();  
            move();  
            turnLeft();  
        }  
    }  
}
```





# Place Beeper Square

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

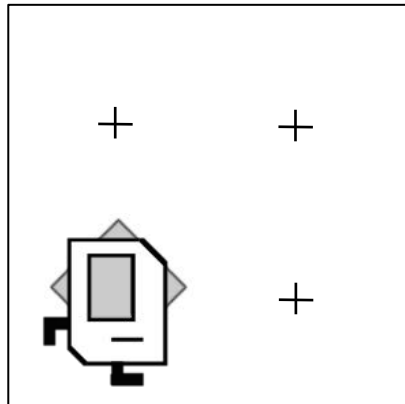


First time  
through the  
loop



# Place Beeper Square

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

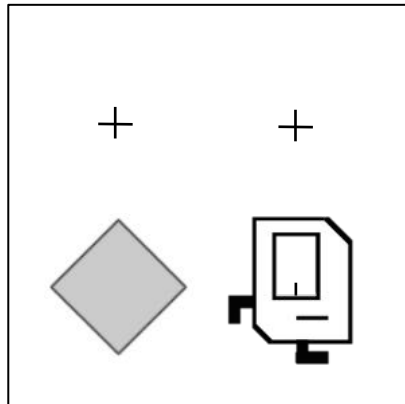


First time  
through the  
loop



# Place Beeper Square

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

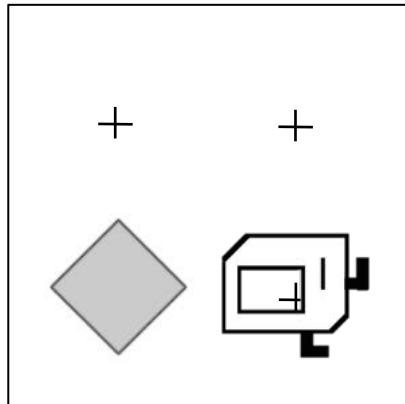


First time  
through the  
loop



# Place Beeper Square

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

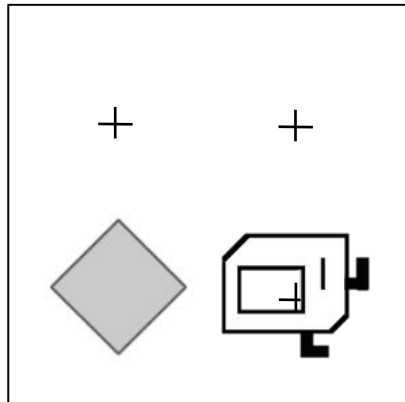


First time  
through the  
loop



# Place Beeper Square

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

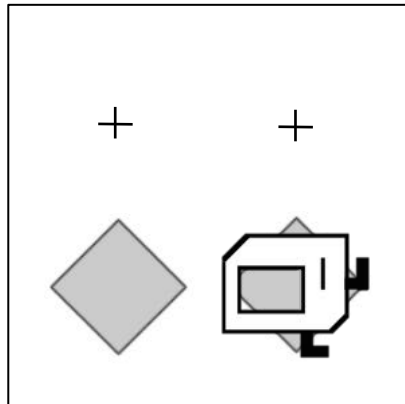


Second time  
through the  
loop



# Place Beeper Square

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

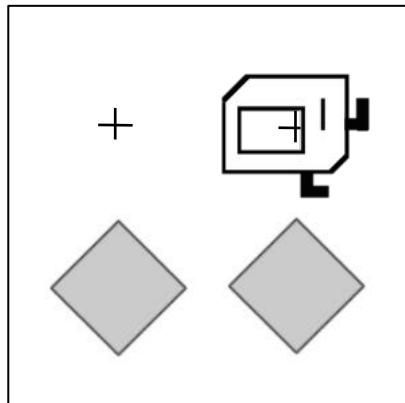


Second time through the loop



# Place Beeper Square

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

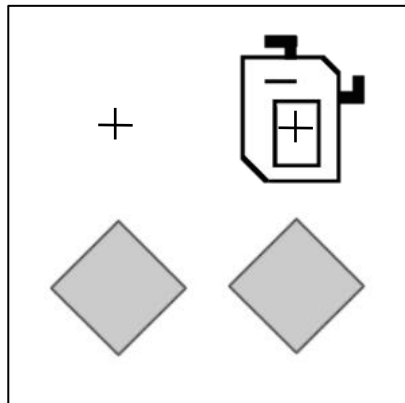


Second time  
through the  
loop



# Place Beeper Square

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



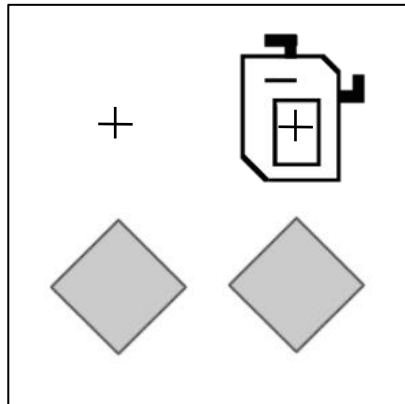
Second time  
through the  
loop





# Place Beeper Square

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

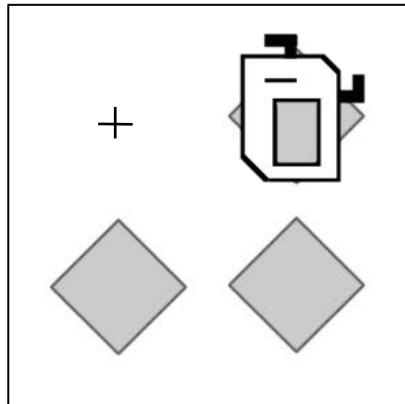


Third time  
through the  
loop



# Place Beeper Square

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

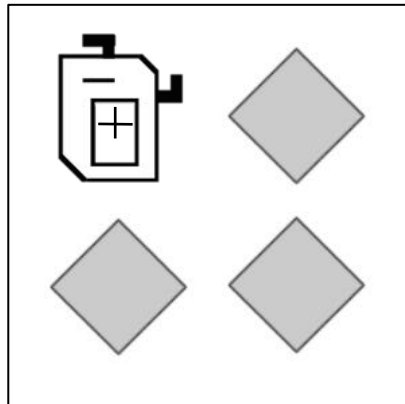


Third time through the loop



# Place Beeper Square

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

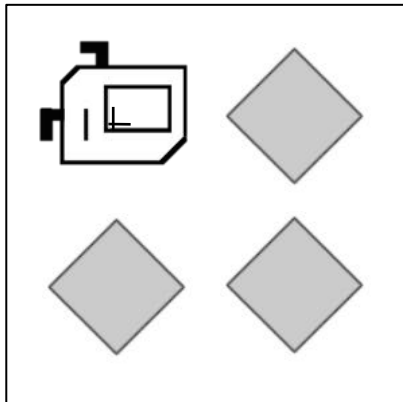


Third time  
through the  
loop



# Place Beeper Square

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

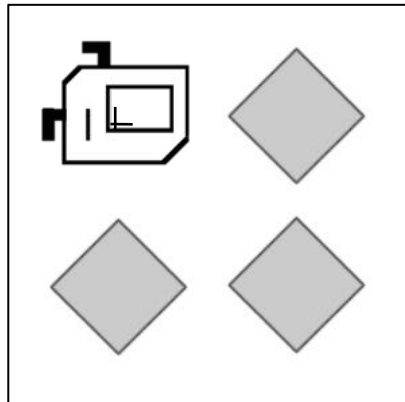


Third time through the loop



# Place Beeper Square

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

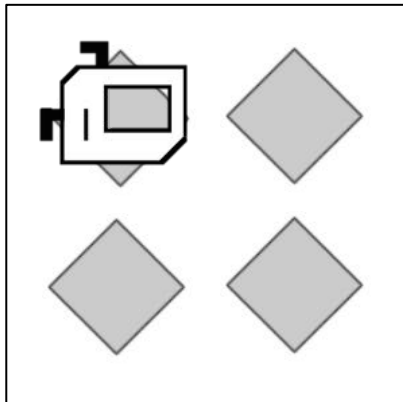


**Fourth** time  
through the  
loop



# Place Beeper Square

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

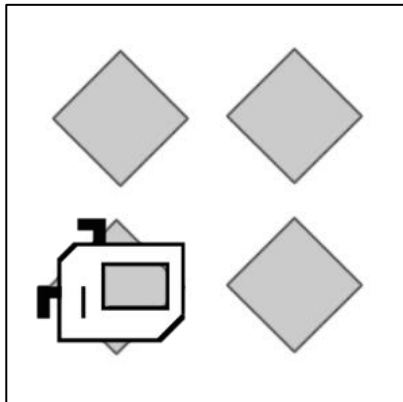


**Fourth time**  
through the  
loop



# Place Beeper Square

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

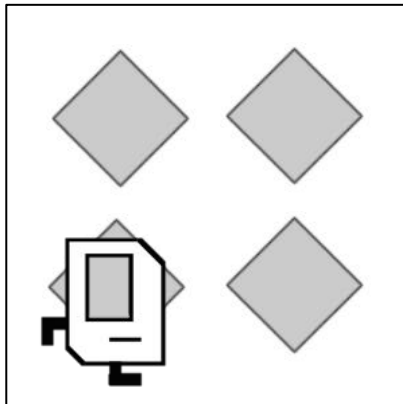


**Fourth** time  
through the  
loop



# Place Beeper Square

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



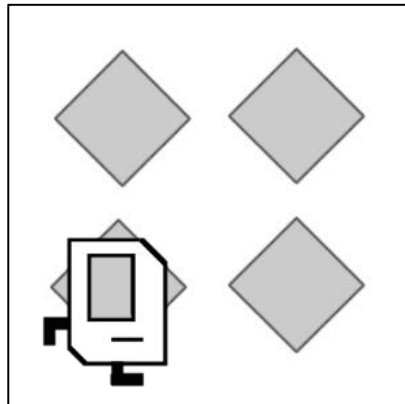
**Fourth** time through the loop





# Place Beeper Square

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



Exciting!

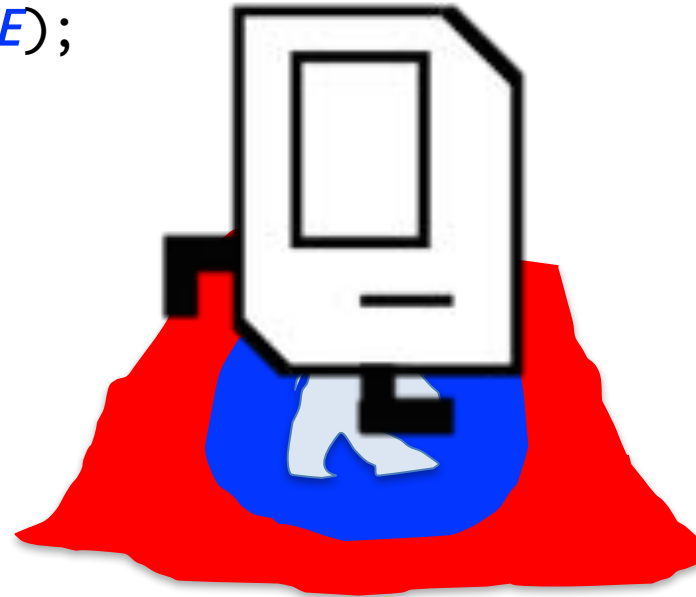
# Aside: Super Karel

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



# Aside: Super Karel

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

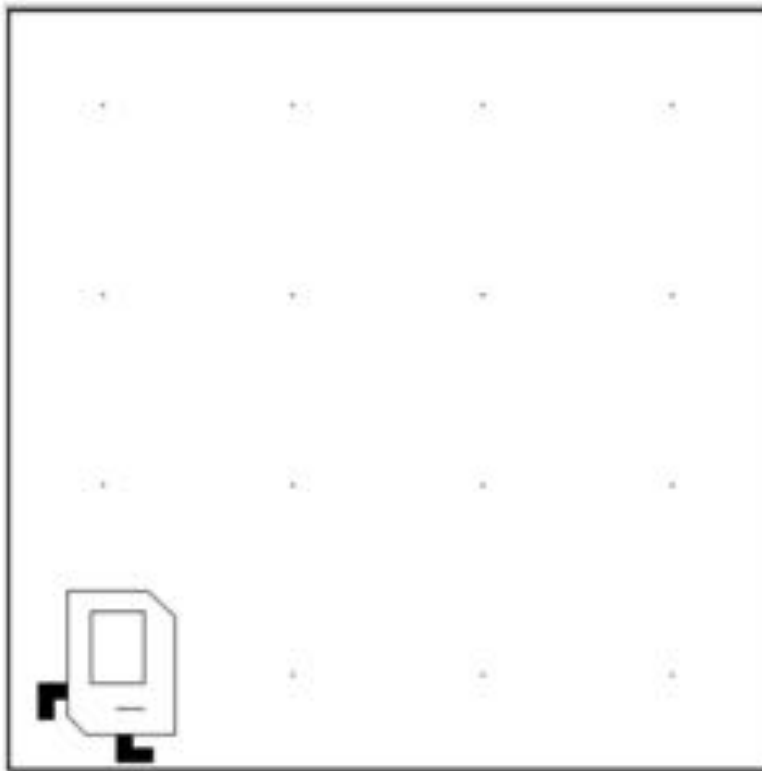


Next task

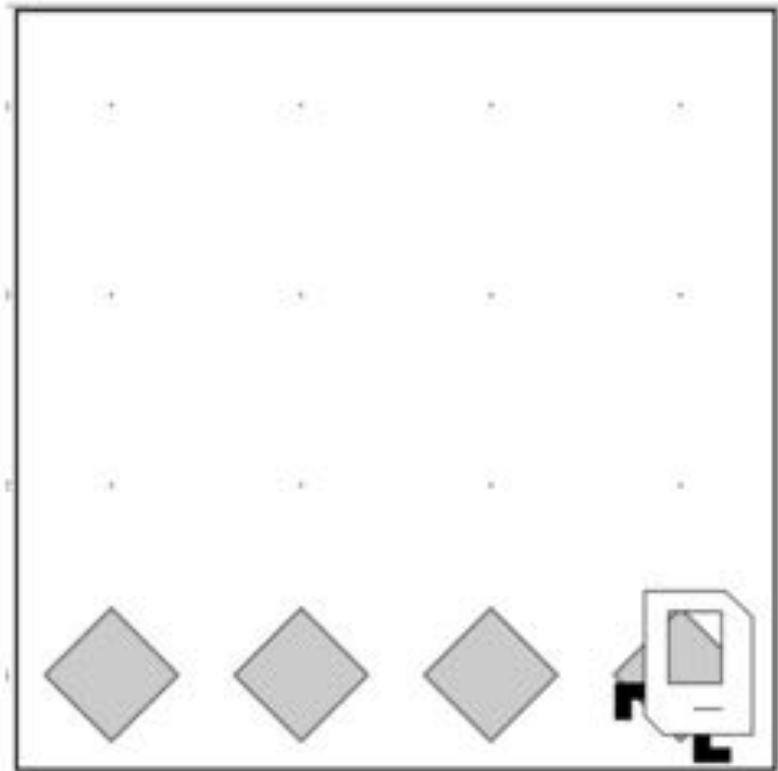
# Place Beeper Line

*Try and solve it!*

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();
        }
    }
}
```





221B

# Place Beeper Line

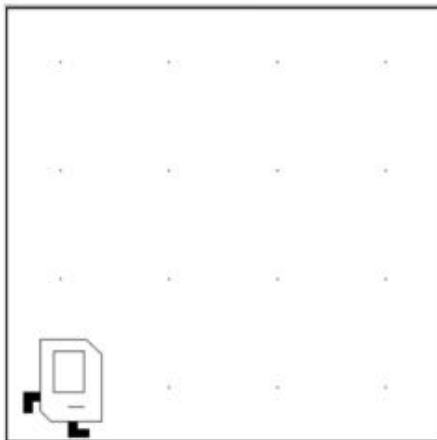
```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

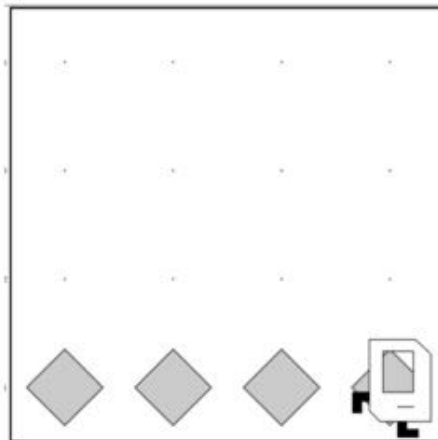
    public void run() {
        for(int i = 0; i < 4; i++) {
            putBeeper();
            move();
        }
    }
}
```

What we want

Before



After

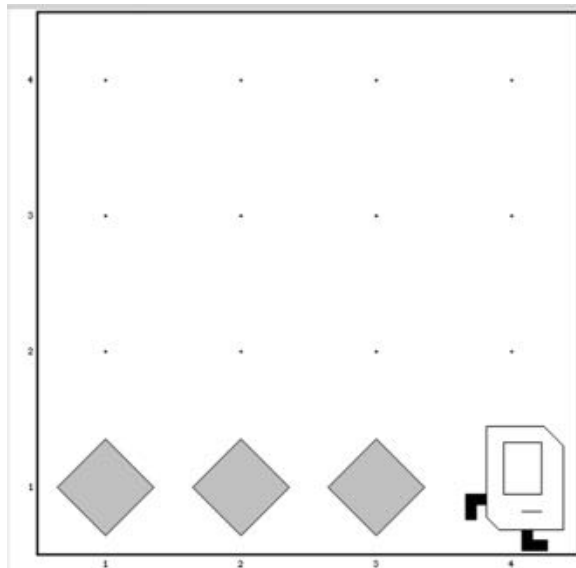


# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        for(int i = 0; i < 3; i++) {
            putBeeper();
            move();
        }
    }
}
```

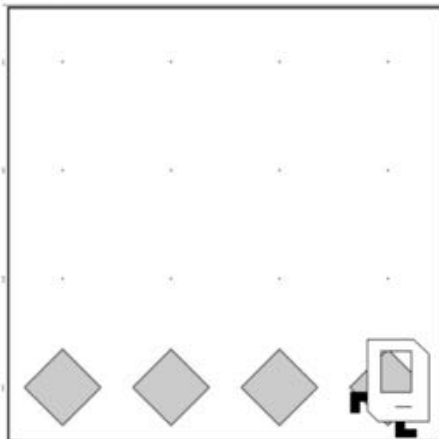


# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        for(int i = 0; i < 3; i++) {
            putBeeper();
            move();
        }
        // extra put beeper
        putBeeper();
    }
}
```



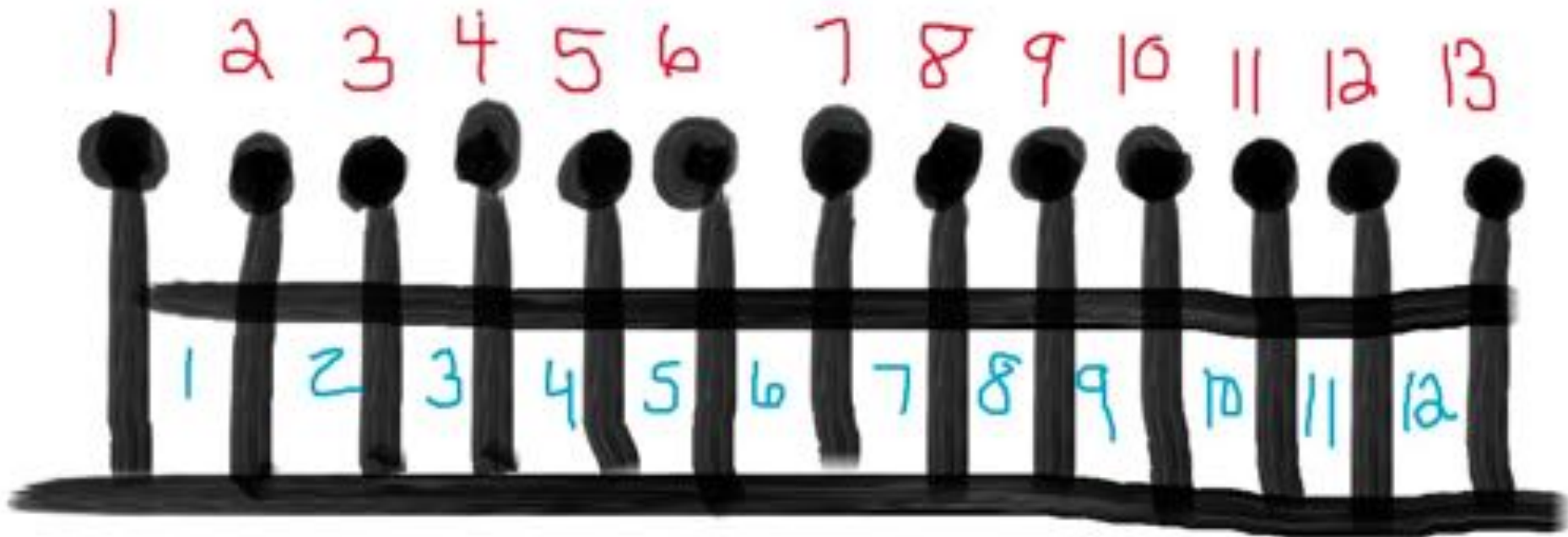
# Actual Bug from Marc II



# Grace Hopper



# Fence Post Problem

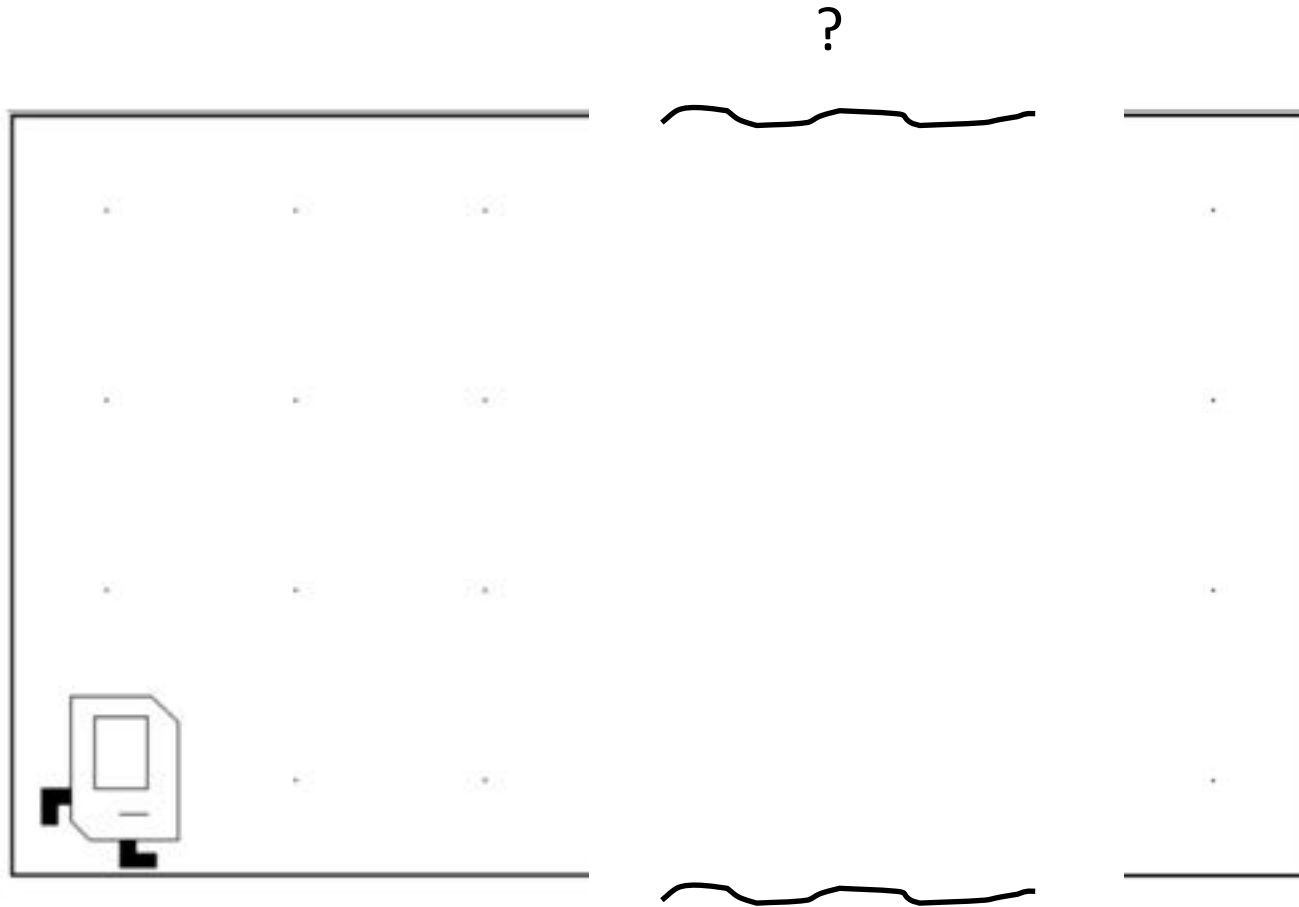


\* Also sometimes called an Off By One Error



Unstoppable

# Don't Know World Size



While Loop

# While Loop

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

        // example while loop
        while(condition) {
            code to repeat
        }

    }

}
```



# Possible Conditions

<i>Test</i>	<i>Opposite</i>	<i>What it checks</i>
<code>frontIsClear()</code>	<code>frontIsBlocked()</code>	Is there a wall in front of Karel?
<code>leftIsClear()</code>	<code>leftIsBlocked()</code>	Is there a wall to Karel's left?
<code>rightIsClear()</code>	<code>rightIsBlocked()</code>	Is there a wall to Karel's right?
<code>beepersPresent()</code>	<code>noBeepersPresent()</code>	Are there beepers on this corner?
<code>beepersInBag()</code>	<code>noBeepersInBag()</code>	Any there beepers in Karel's bag?
<code>facingNorth()</code>	<code>notFacingNorth()</code>	Is Karel facing north?
<code>facingEast()</code>	<code>notFacingEast()</code>	Is Karel facing east?
<code>facingSouth()</code>	<code>notFacingSouth()</code>	Is Karel facing south?
<code>facingWest()</code>	<code>notFacingWest()</code>	Is Karel facing west?

This is **Table 1** on page 18 of the Karel courser reader.



# Move to Wall

```
import stanford.karel.*;

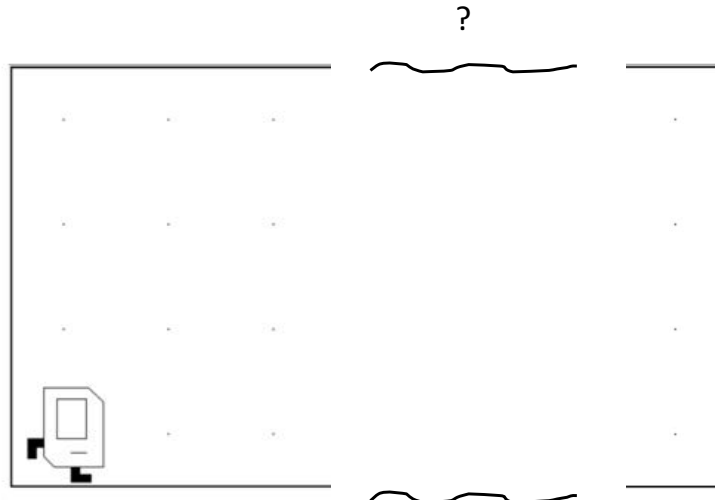
public class BeeperLine extends SuperKarel {

    public void run() {

        // example while loop
        while(condition) {
            code to repeat
        }

    }

}
```



# Move to Wall

```
import stanford.karel.*;

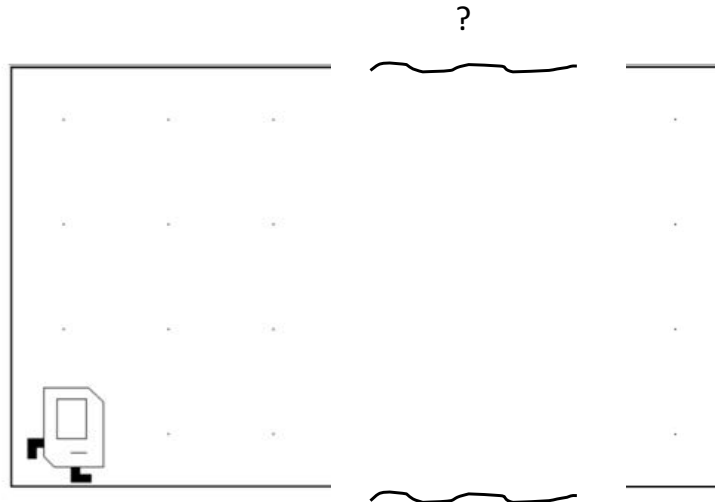
public class BeeperLine extends SuperKarel {

    public void run() {

        // example while loop
        while(frontIsClear()) {
            code to repeat
        }

    }

}
```



# Move to Wall

```
import stanford.karel.*;

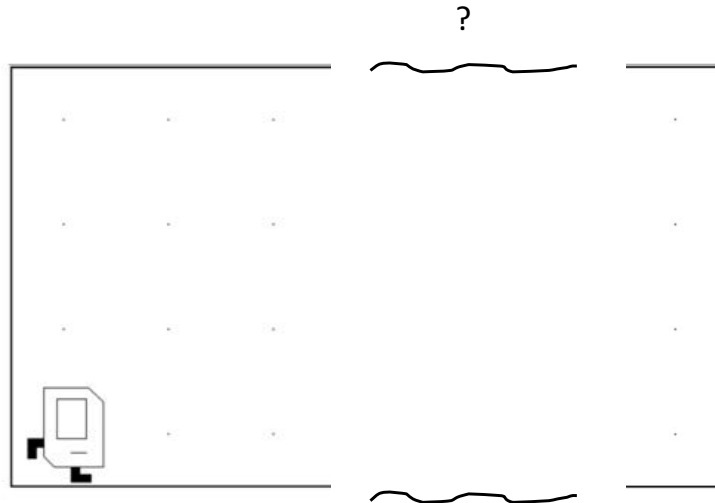
public class BeeperLine extends SuperKarel {

    public void run() {

        // example while loop
        while(frontIsClear()) {
            move();
        }

    }

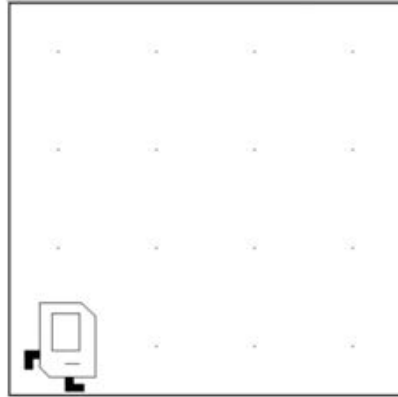
}
```



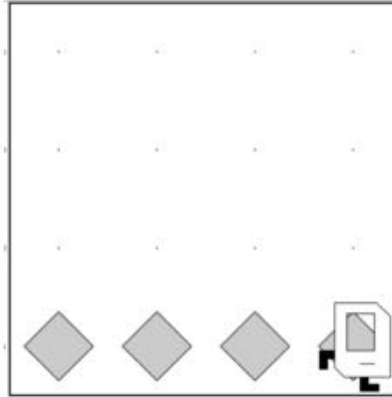


# Code that Works in Any World

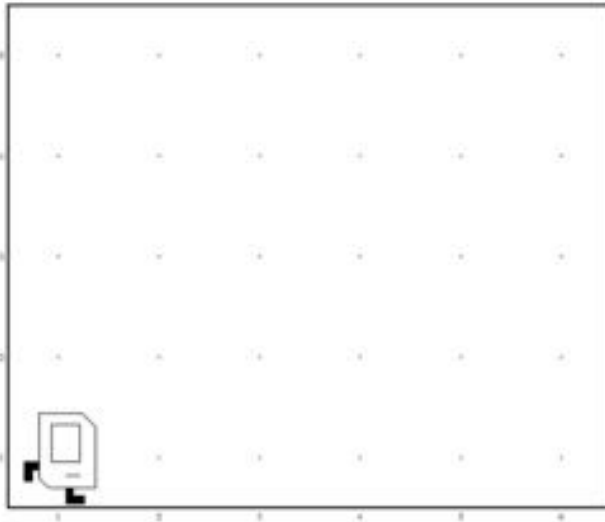
Before



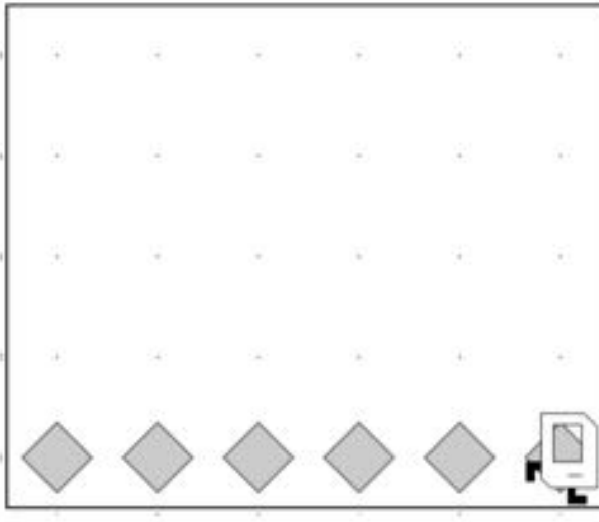
After



Before



After



# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

        // example while loop
        while(frontIsClear()) {
            move();
        }
    }
}
```



# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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



# 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();
    }

}
```



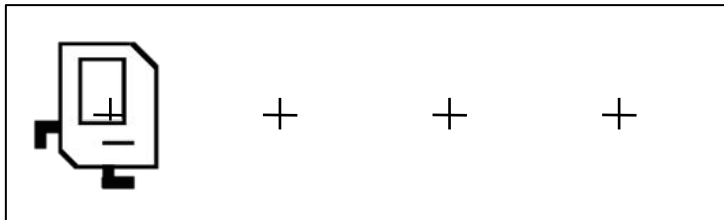
# Place Beeper Line

```
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

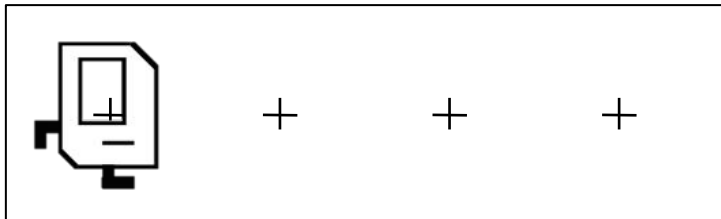
```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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

}
```



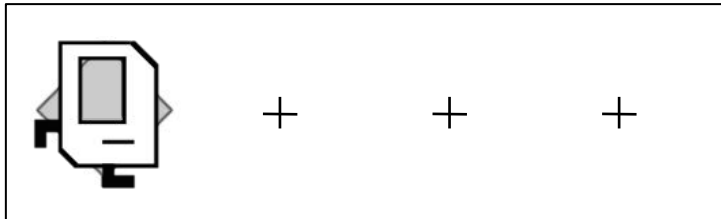
# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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



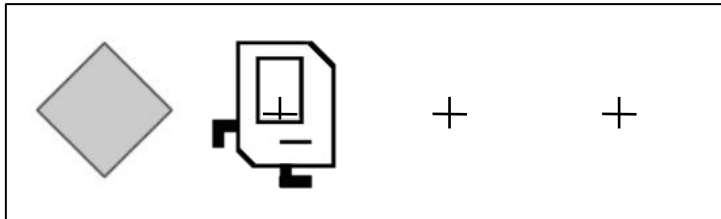
# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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





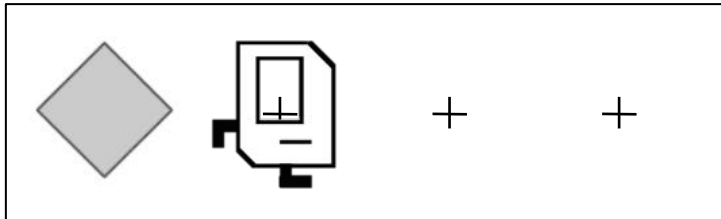
# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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



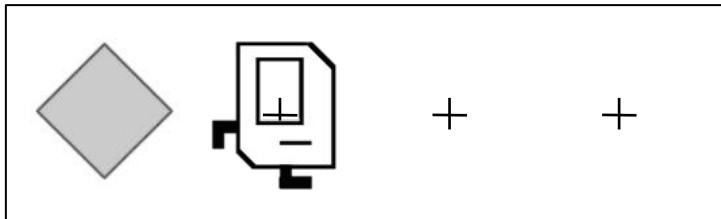
# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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



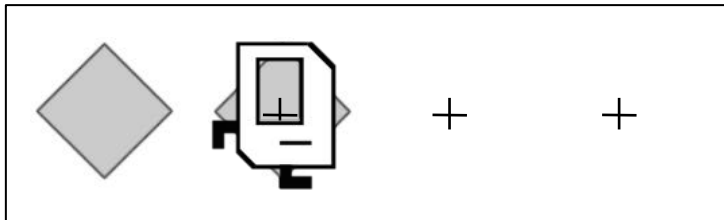
# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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



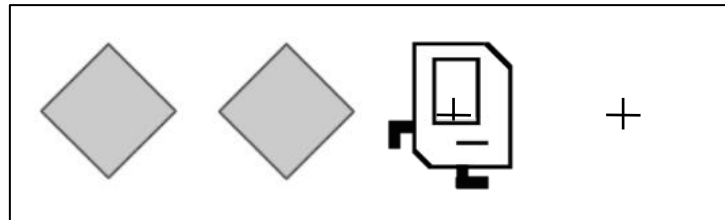
# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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



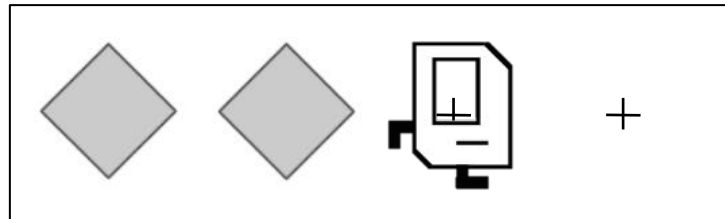
# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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



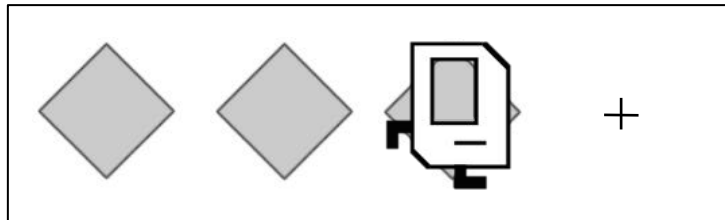
# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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



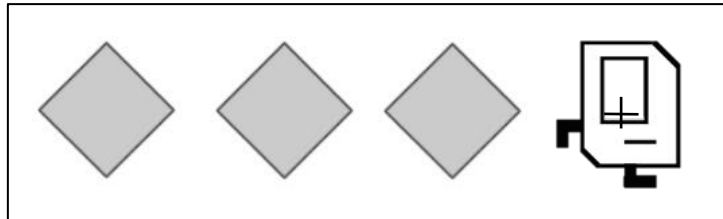
# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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



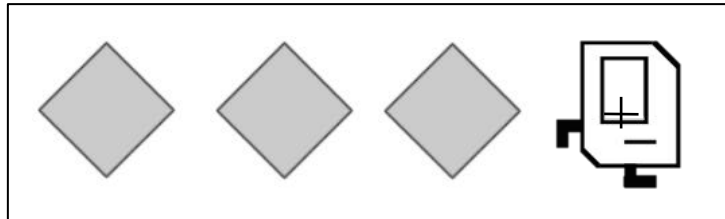
# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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





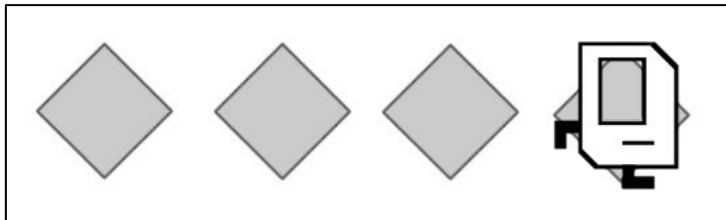
# Place Beeper Line

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {

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





Common misconception:

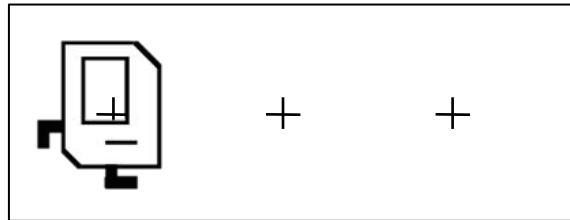
# Place Beeper Line: Redux

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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



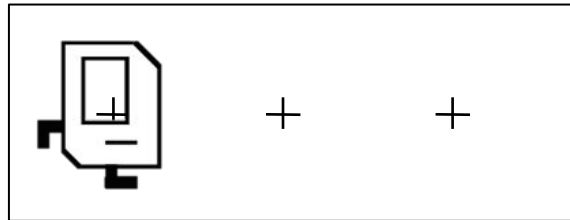
# Place Beeper Line: Redux

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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



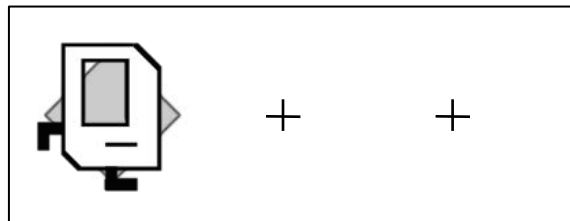
# Place Beeper Line: Redux

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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



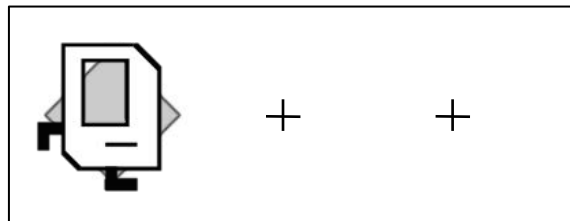
# Place Beeper Line: Redux

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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



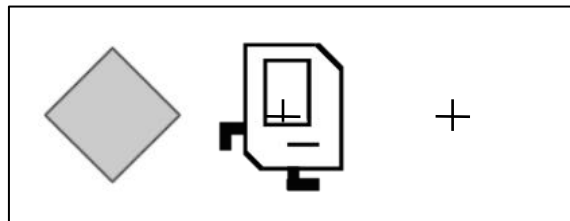
# Place Beeper Line: Redux

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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





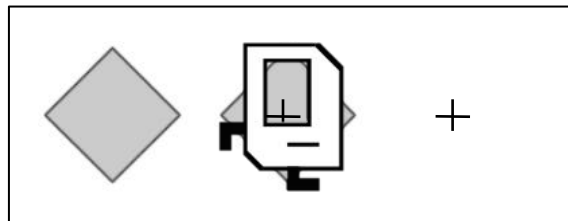
# Place Beeper Line: Redux

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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



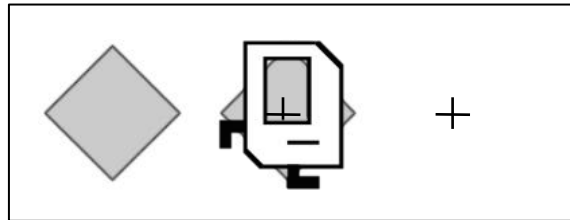
# Place Beeper Line: Redux

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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



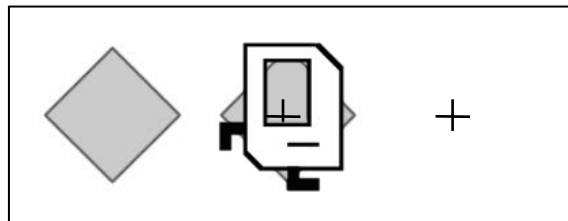
# Place Beeper Line: Redux

```
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

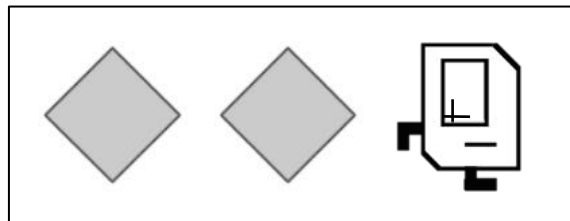
```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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

This is  
incredibly  
important!



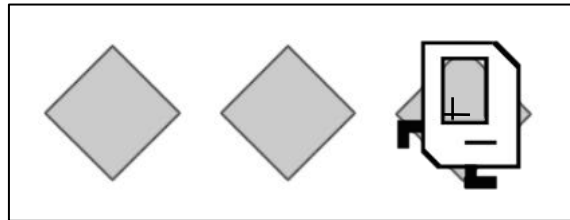
# Place Beeper Line: Redux

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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



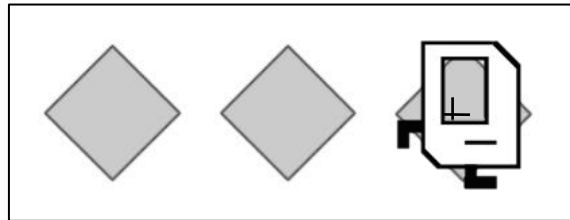
# Place Beeper Line: Redux

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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



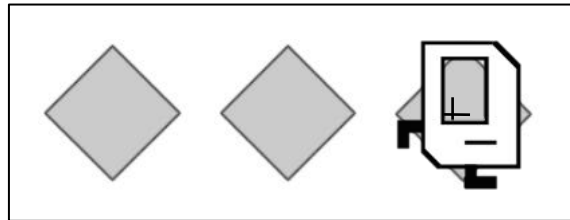
# Place Beeper Line: Redux

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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



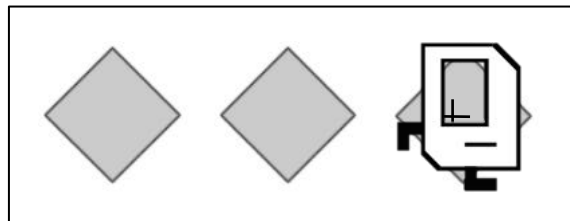
# Place Beeper Line: Redux

```
import stanford.karel.*;

public class BeeperLine extends SuperKarel {

    public void run() {
        // place a first beeper
        putBeeper();

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





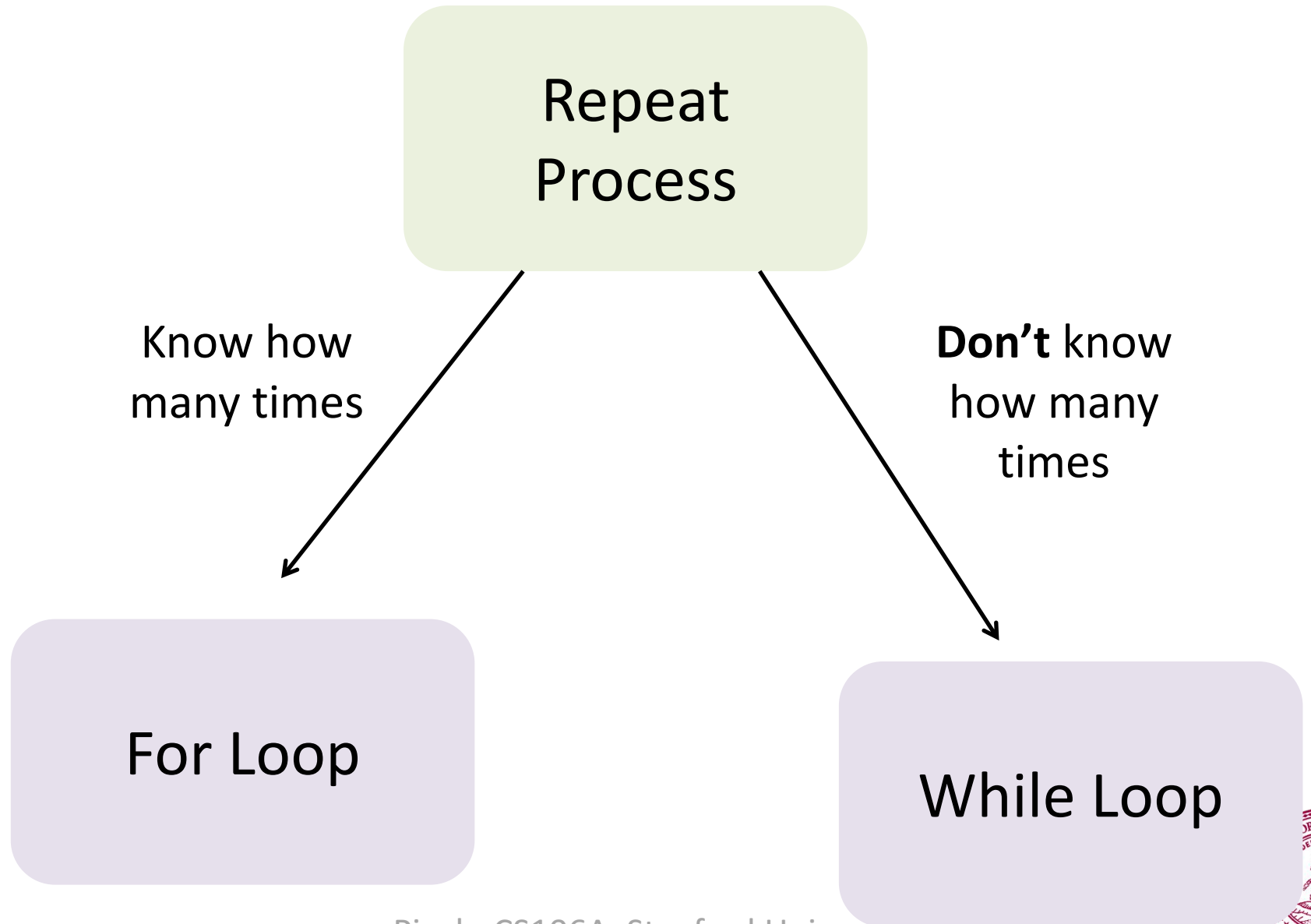


A program executes one line at a time.

The while loop checks its condition only at the start of the code block and before repeating.



# 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();
        }
    }
}
```





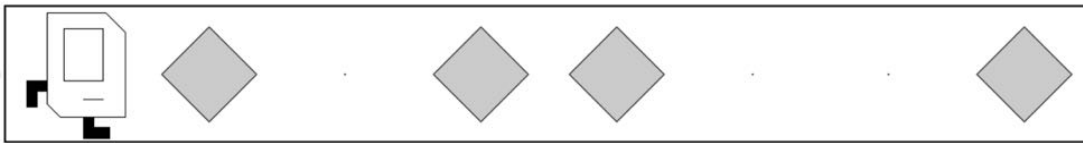
# The Full Karel

<p><b>Built-in Karel commands:</b></p> <pre> move(); turnLeft(); putBeeper(); pickBeeper(); </pre>	<p><b>Conditional statements:</b></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><b>Karel program structure:</b></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><b>Iterative statements:</b></p> <pre> for (int i = 0; i &lt; count; i++) {     statements to be repeated }  while (condition) {     statements to be repeated } </pre>
<p><b>Karel condition names:</b></p> <pre> frontIsClear()    frontIsBlocked() leftIsClear()     leftIsBlocked() rightIsClear()    rightIsBlocked() beepersPresent() noBeepersPresent() beepersInBag()   noBeepersInBag() facingNorth()    notFacingNorth() facingEast()     notFacingEast() facingSouth()    notFacingSouth() facingWest()     notFacingWest() </pre>	<p><b>Method definition:</b></p> <pre> private void name () {     statements in the method body } </pre> <p><b>New commands in the SuperKarel class:</b></p> <pre> turnRight(); turnAround(); paintCorner (color); </pre> <p><b>New conditions in the SuperKarel class:</b></p> <pre> random() random(p) cornerColorIs (color) </pre>



# Random Painter

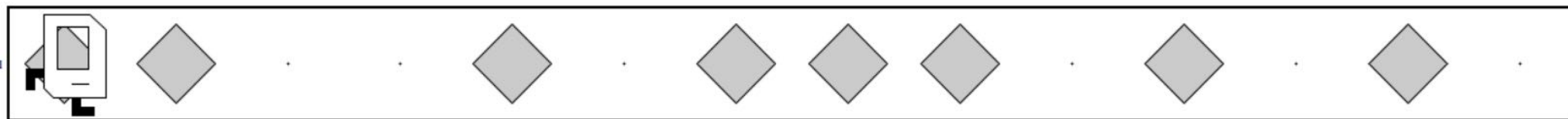
Before:



After:



Before:



After:



You just learned most of  
programming "control flow"

# Today's Goal

1. Code using loops and conditions
2. Trace programs that use loops and conditions



