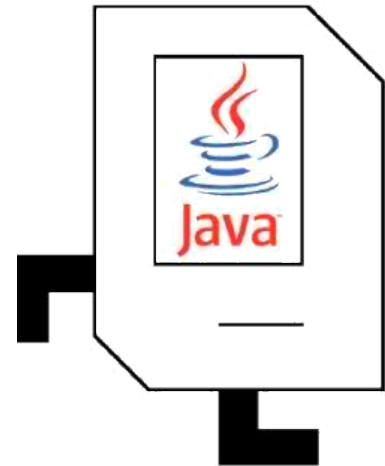




Control Flow

Plan For Today

- Karel the Robot So Far
- Control Flow
 - Reviewing For loops
 - While loops
 - If/else statements



Hi there! I'm Nick.



I'm a Computer Science Lecturer at Stanford.

Plan For Today

- Karel the Robot So Far



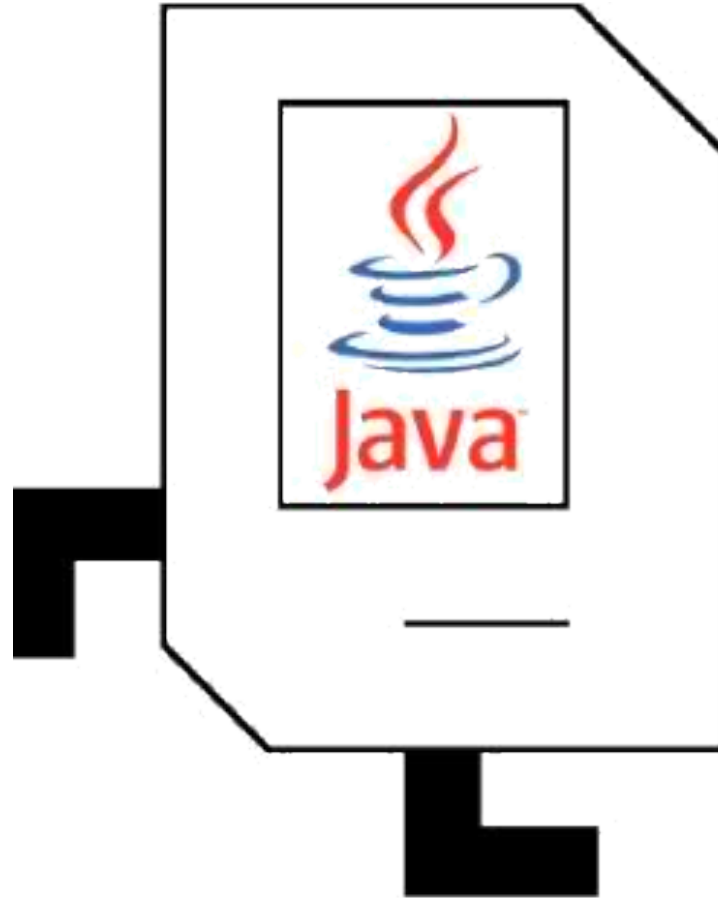
- Control Flow

 - Reviewing For loops

 - While loops

 - If/else statements

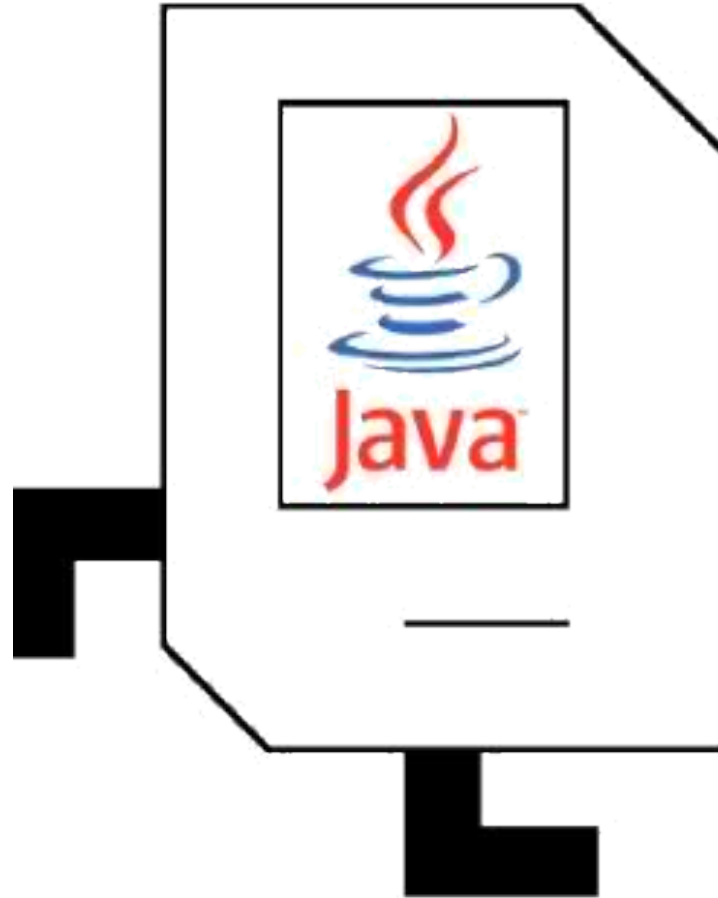
Karel the Robot!



Karel the Robot!



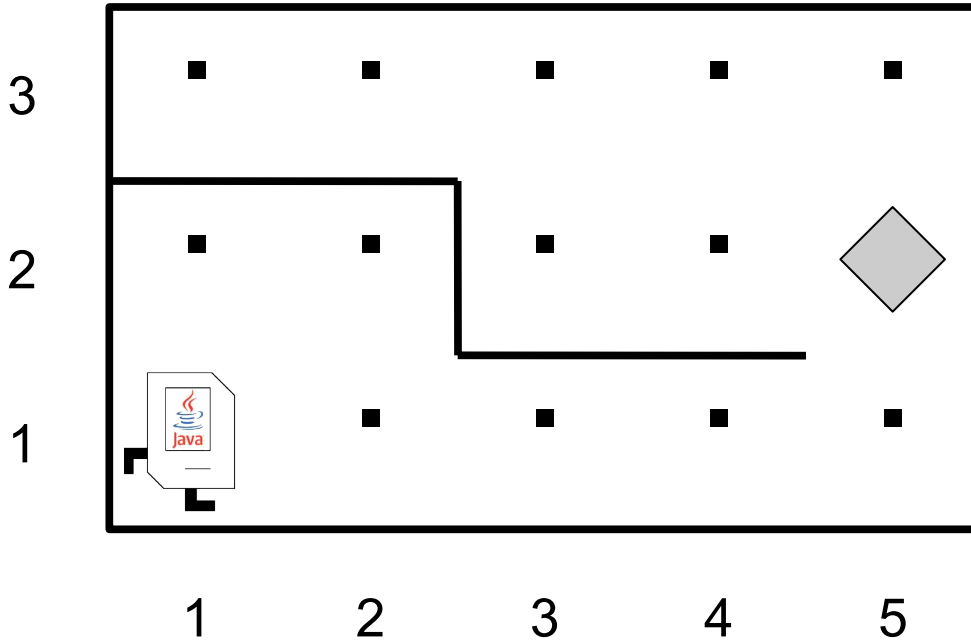
Karel the Robot!



Karel the Robot!



Karel's World



Karel Knows 4 Main Commands



`move`

`turnLeft`

`putBeeper`

`pickBeeper`

Defining New Commands

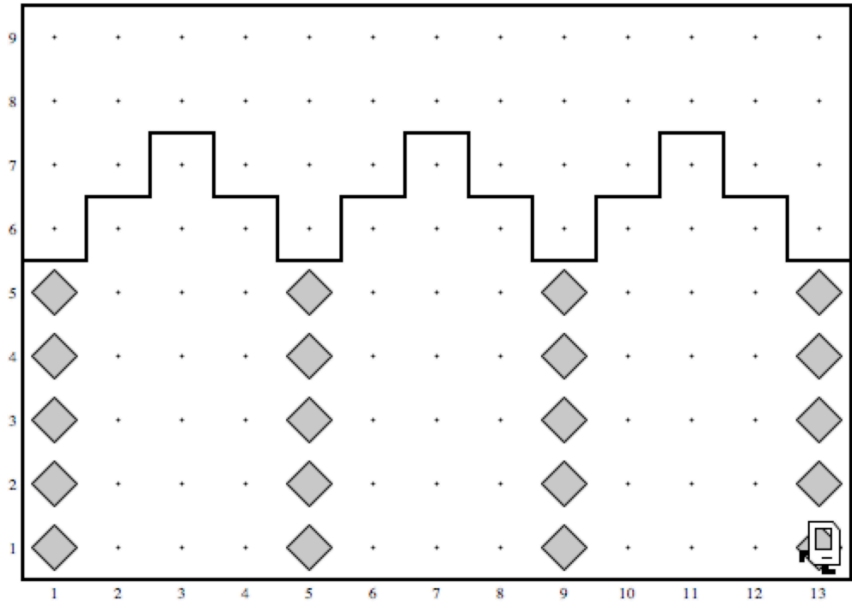
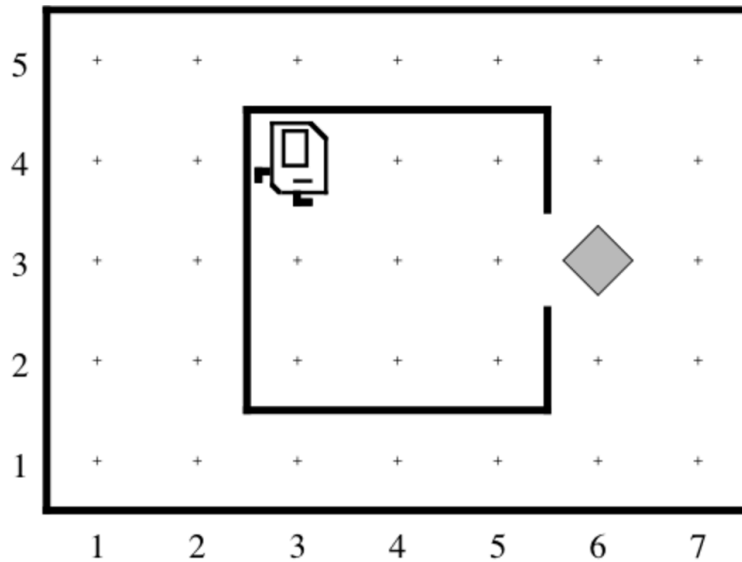
We can make new commands (or **methods**) for Karel. This lets us *decompose* our program into smaller pieces that are easier to understand.

```
private void name() {  
    statement;  
    statement;  
    ...  
}
```

For example:

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

Writing Our First Karel Programs



What did you find most surprising or challenging when working on these programs?

Plan For Today

- Karel the Robot So Far

- **Control Flow**

- Reviewing For loops

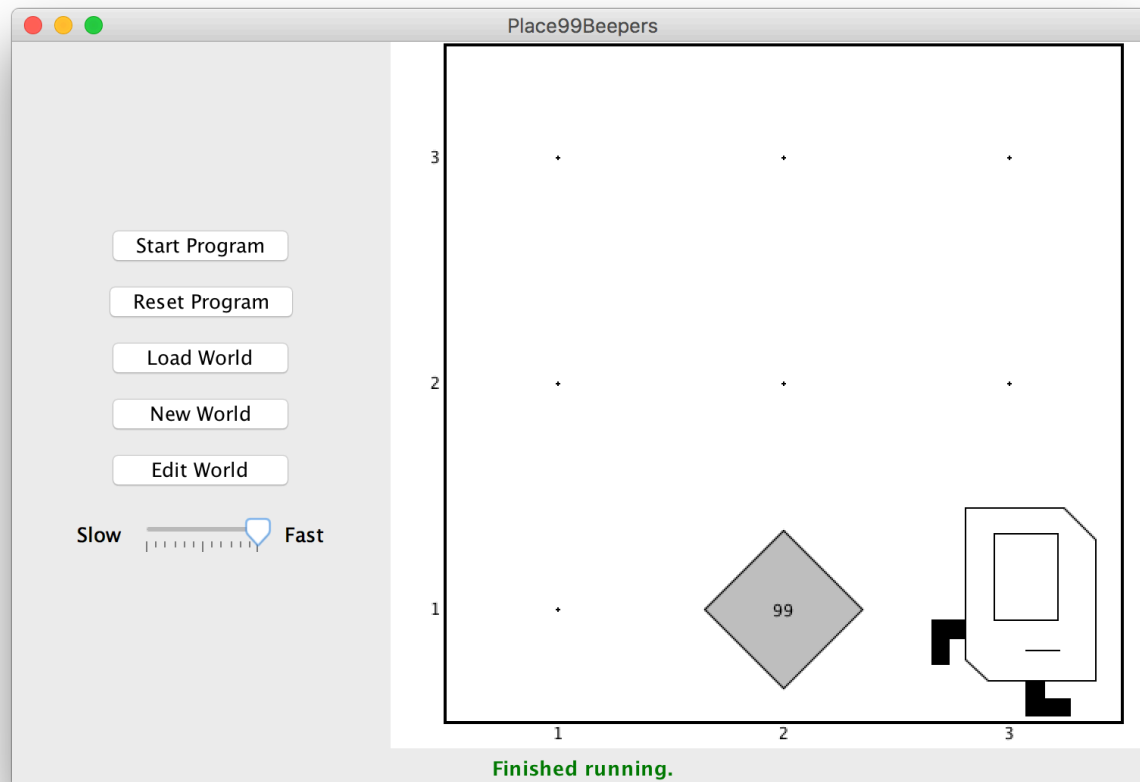


- While loops

- If/else statements

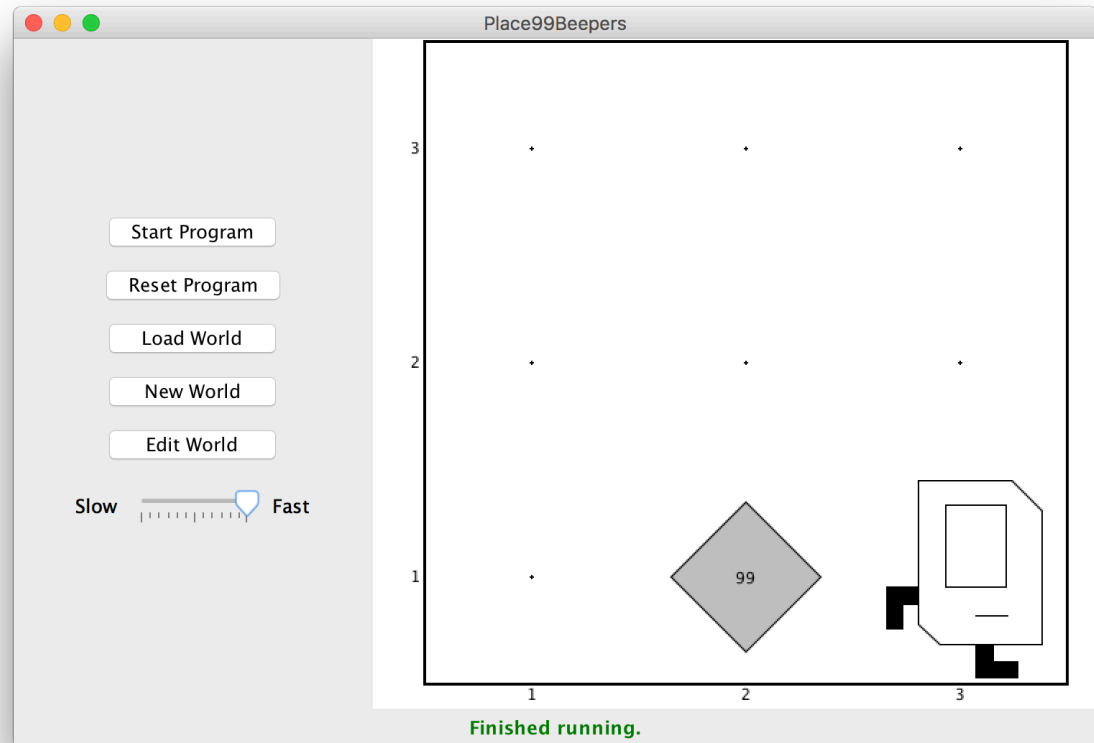
Control Flow: For Loops

I want to make Karel put 99 beepers down on a corner. How do I do this?



Control Flow: For Loops

```
move();  
putBeeper();  
putBeeper();  
putBeeper();  
...  
move();
```



This is too repetitive! Plus, it's difficult to change (e.g. to 25 beepers).

Control Flow: For Loops

Instead, use a **for** loop:

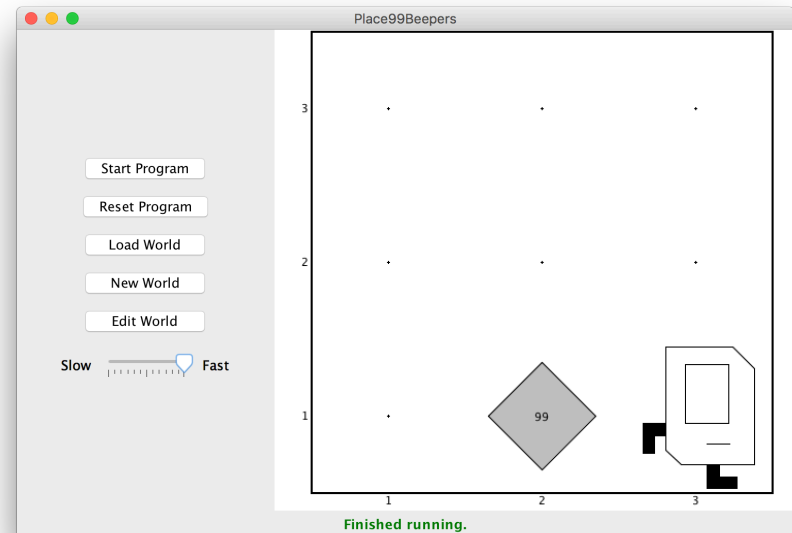
```
for (int i = 0; i < max; i++) {  
    statement;  
    statement;  
    ...  
}
```

Repeats the statements in the body *max* times.

Control Flow: For Loops

Now we can say:

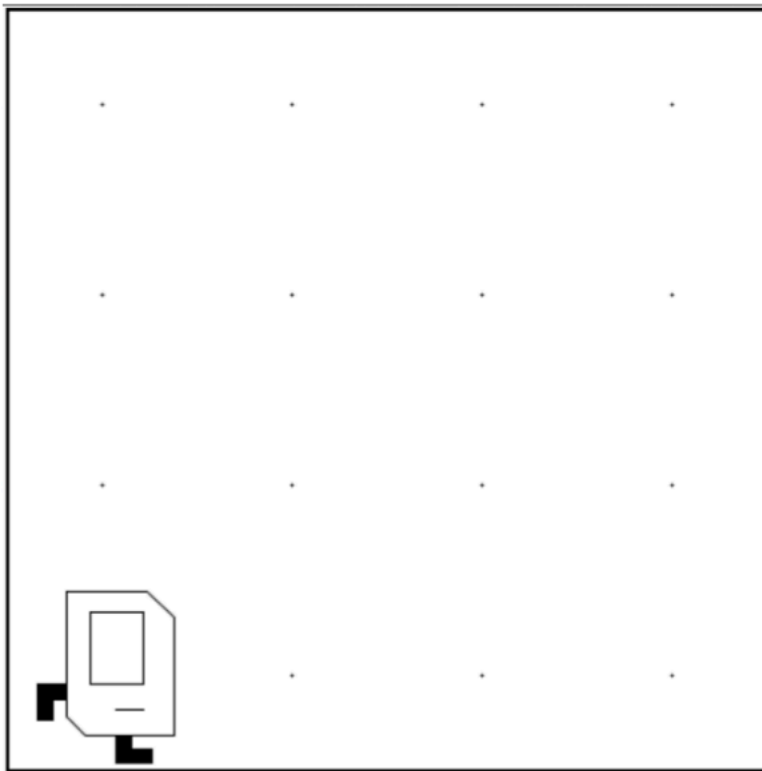
```
move();  
for (int i = 0; i < 99; i++) {  
    putBeeper();  
}  
move();
```



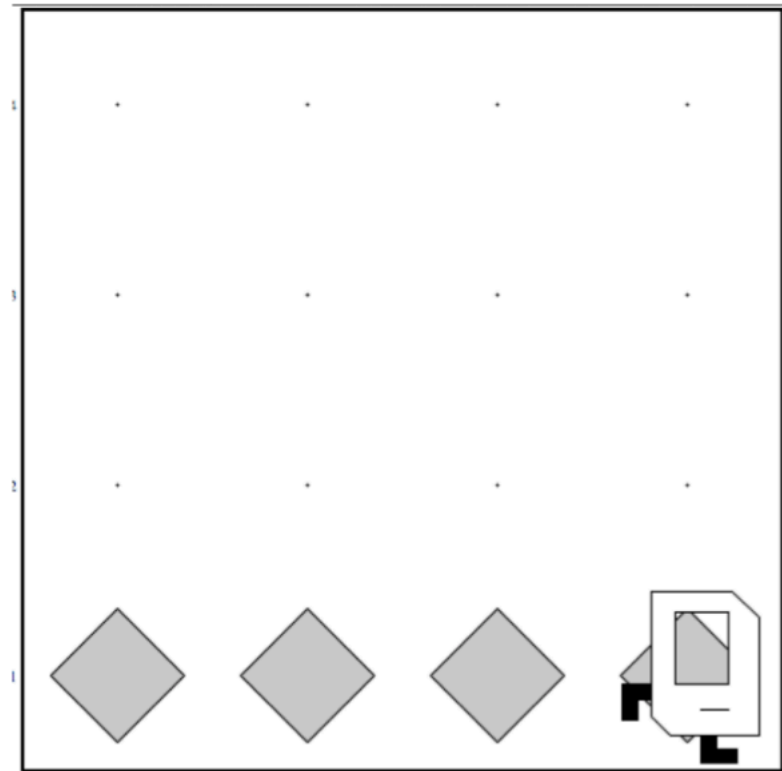
This is less repetitive, and is easier to change (e.g. to 25 beepers).

Next: Place Beeper Line

Before



After



Place Beeper Line

```
import stanford.karel.*;

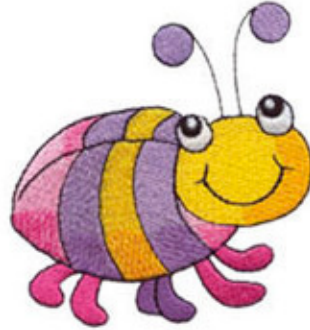
public class BeeperLine extends SuperKarel {

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

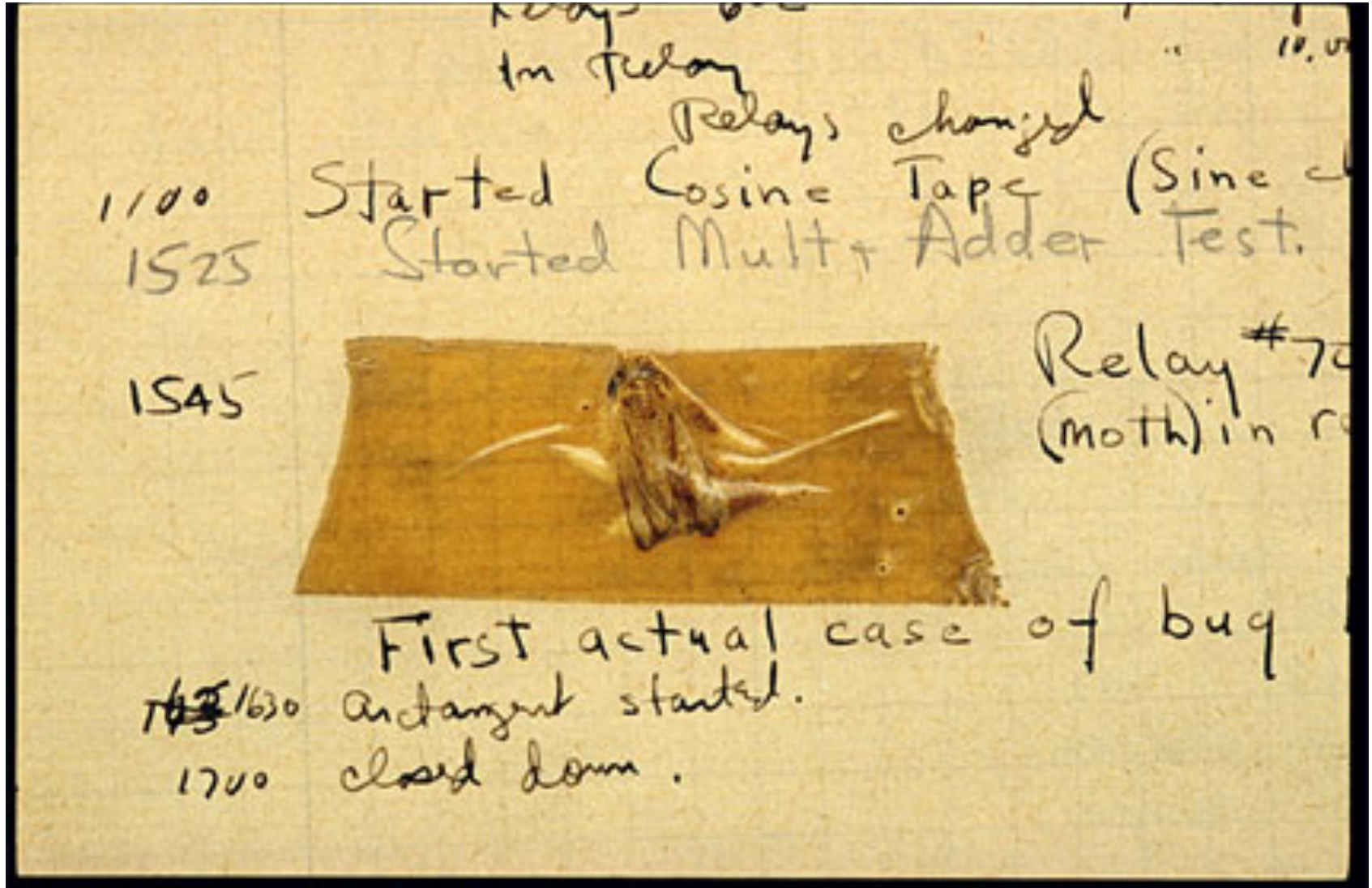


We Have A Bug!

- A **bug** is an issue where our program does not work the way we want it to.



Actual Bug From Mark II

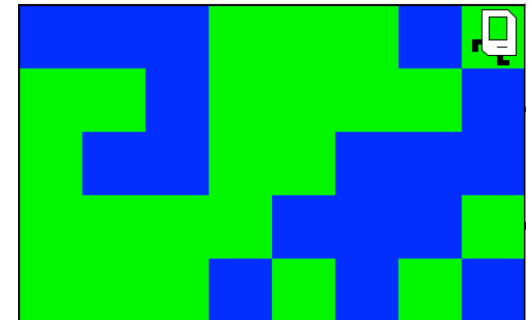
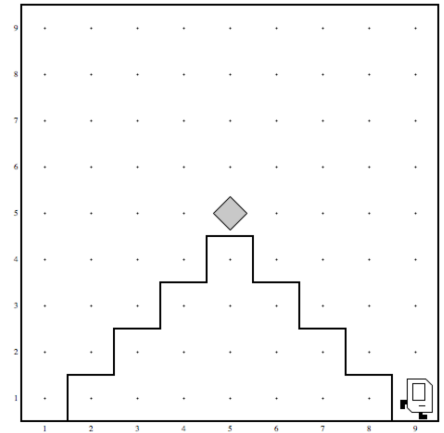
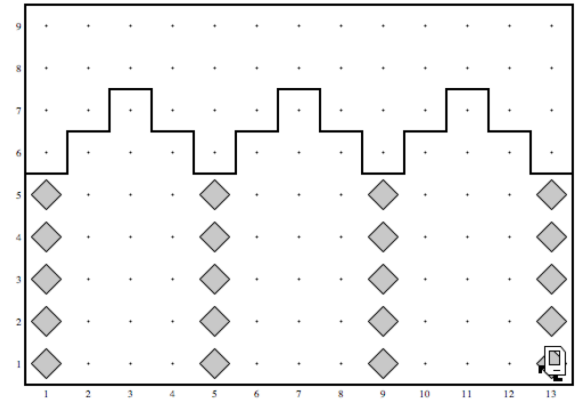


Grace Hopper




Rest Of Today

- Implement your own Build Charles Bridge program that you talked about in section
- Write a program “Mountain Karel” where Karel climbs a mountain of any size and places a beeper on top
- If you have time, write a program “Random Painter” where Karel paints any world randomly with green and blue squares.



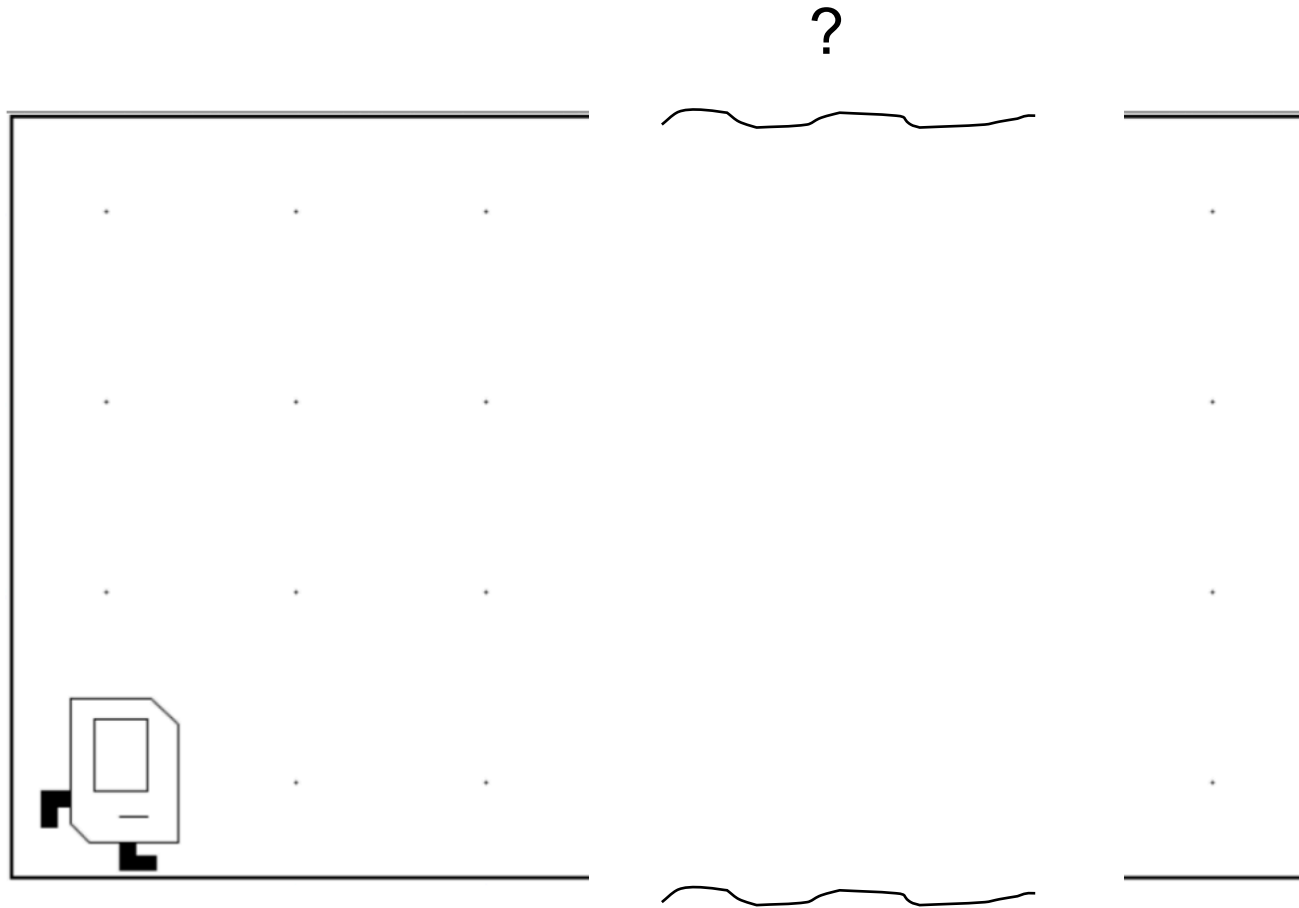
Plan For Today

- Karel the Robot So Far
- Control Flow
 - Reviewing For Loops
 - While loops 
 - If/else statements

Control Flow

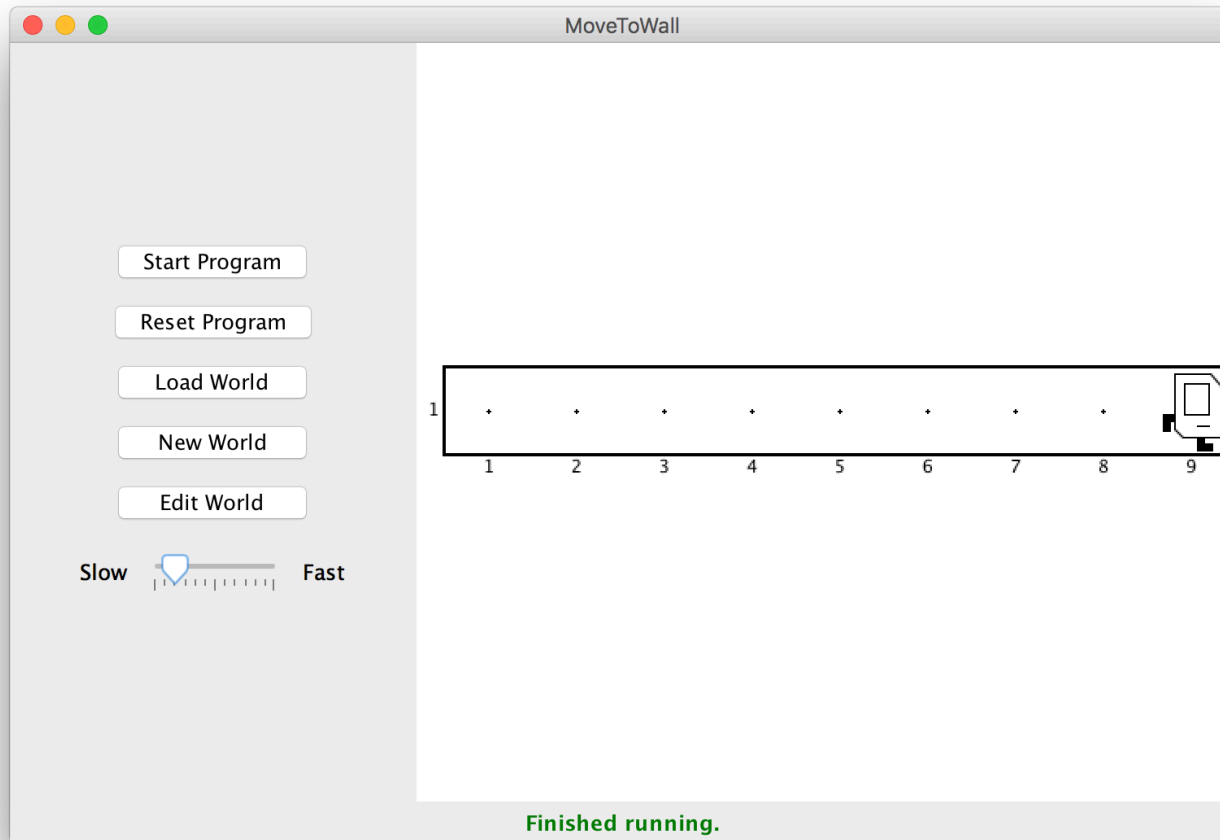
- So far, we have written programs that behave the same every time.
- More complex programs behave differently in different worlds or cases.
- We are going to learn how to write Karel programs that can do things based on what the world is like!

Generalized Programs



Control Flow: While Loops

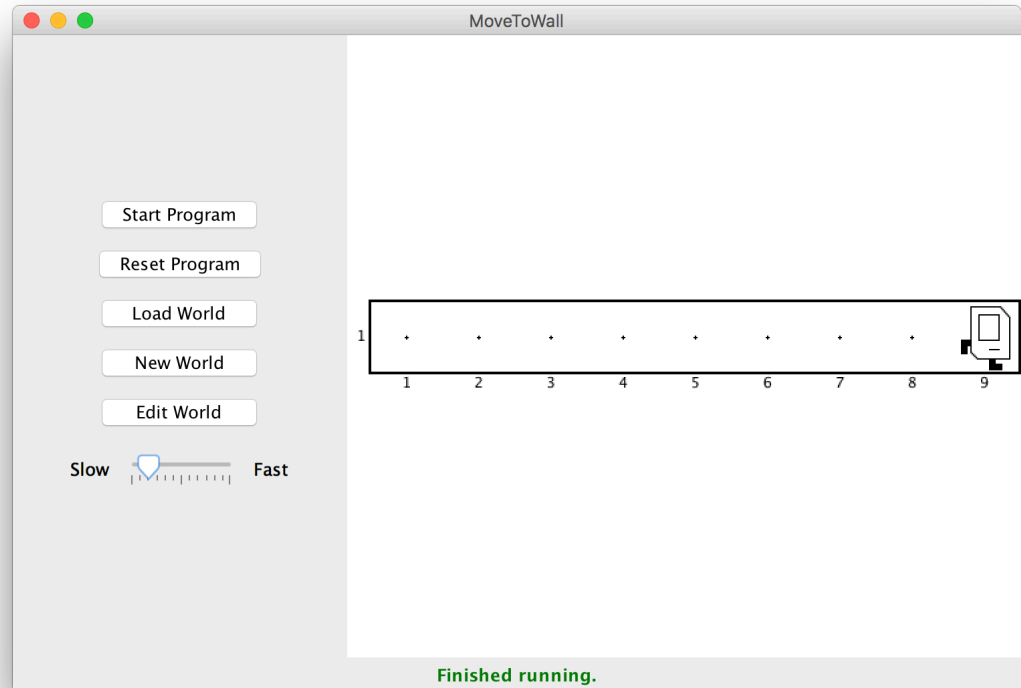
I want Karel to move until reaching a wall. How do I do this?



Control Flow: While Loops

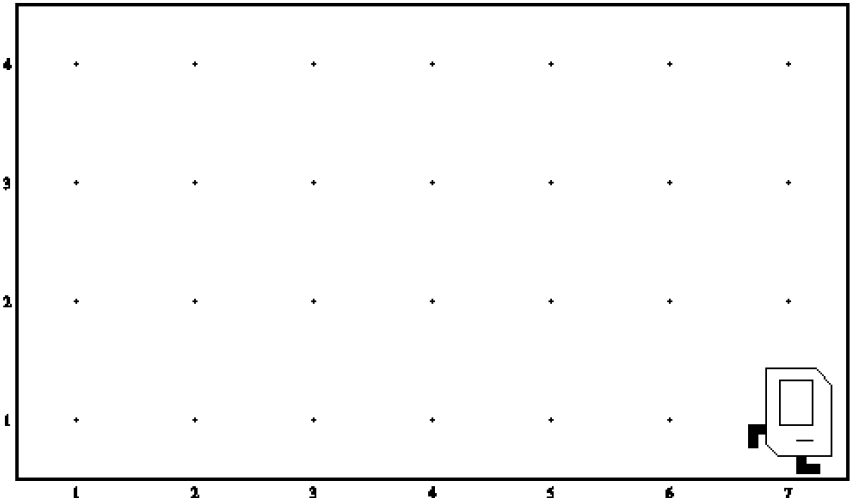
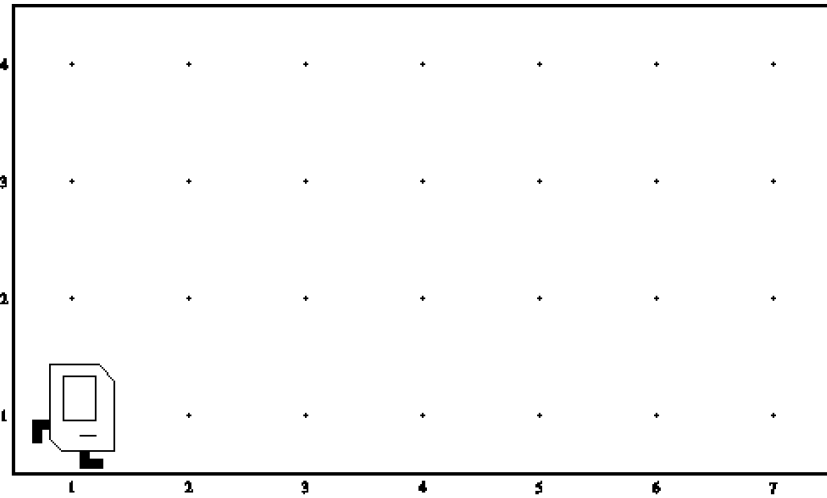
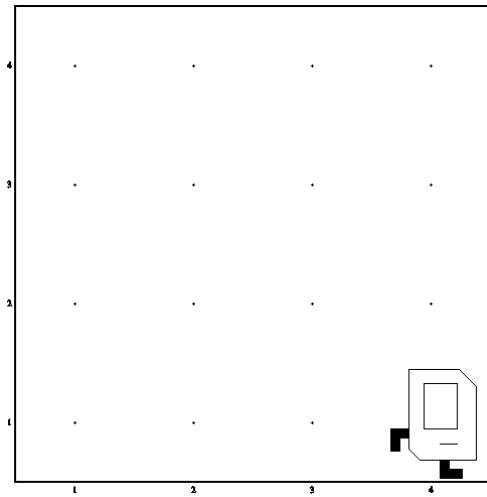
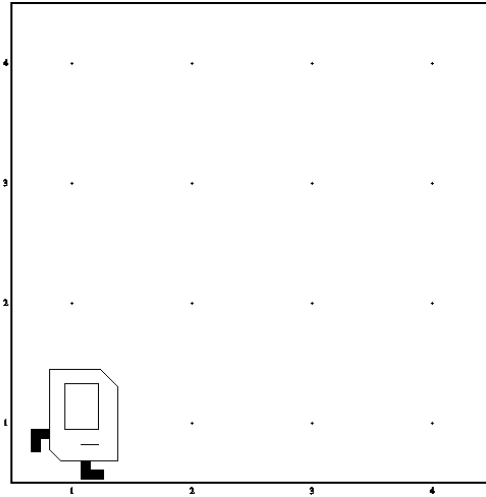
Can't just say:

```
move( );  
move( );  
move( );  
move( );  
...
```



This is too repetitive! Also, we might not know how far away a wall is. Plus, we want our program to be as *generalized* as possible and work in many different worlds.

Working In Any World



Control Flow: While Loops

Instead, use a **while** loop:

```
while (condition) {  
    statement;  
    statement;  
    ...  
}
```

Repeats the statements in the body until ***condition*** is no longer true.

Each time, Karel checks the condition, and then executes *all statements before checking the condition again*.

Possible Conditions

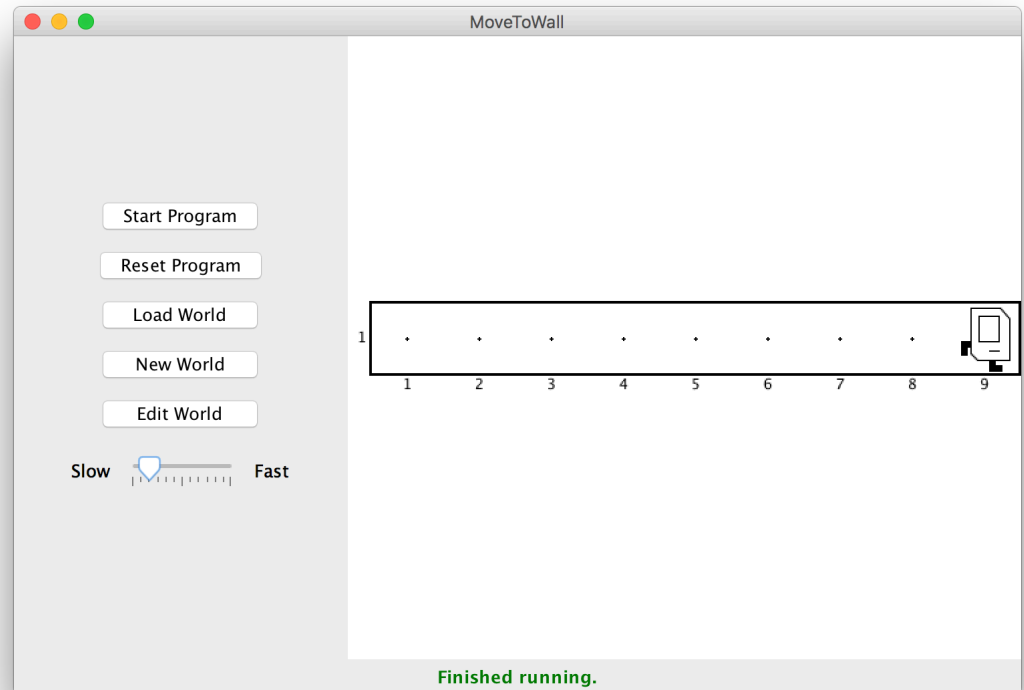
<i>Test</i>	<i>Opposite</i>	<i>What it checks</i>
<code>frontIsClear()</code>	<code>frontIsBlocked()</code>	Is there a wall in front of Karel?
<code>leftIsClear()</code>	<code>leftIsBlocked()</code>	Is there a wall to Karel's left?
<code>rightIsClear()</code>	<code>rightIsBlocked()</code>	Is there a 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 on the course website, under the "Handouts" tab.

Control Flow: While Loops

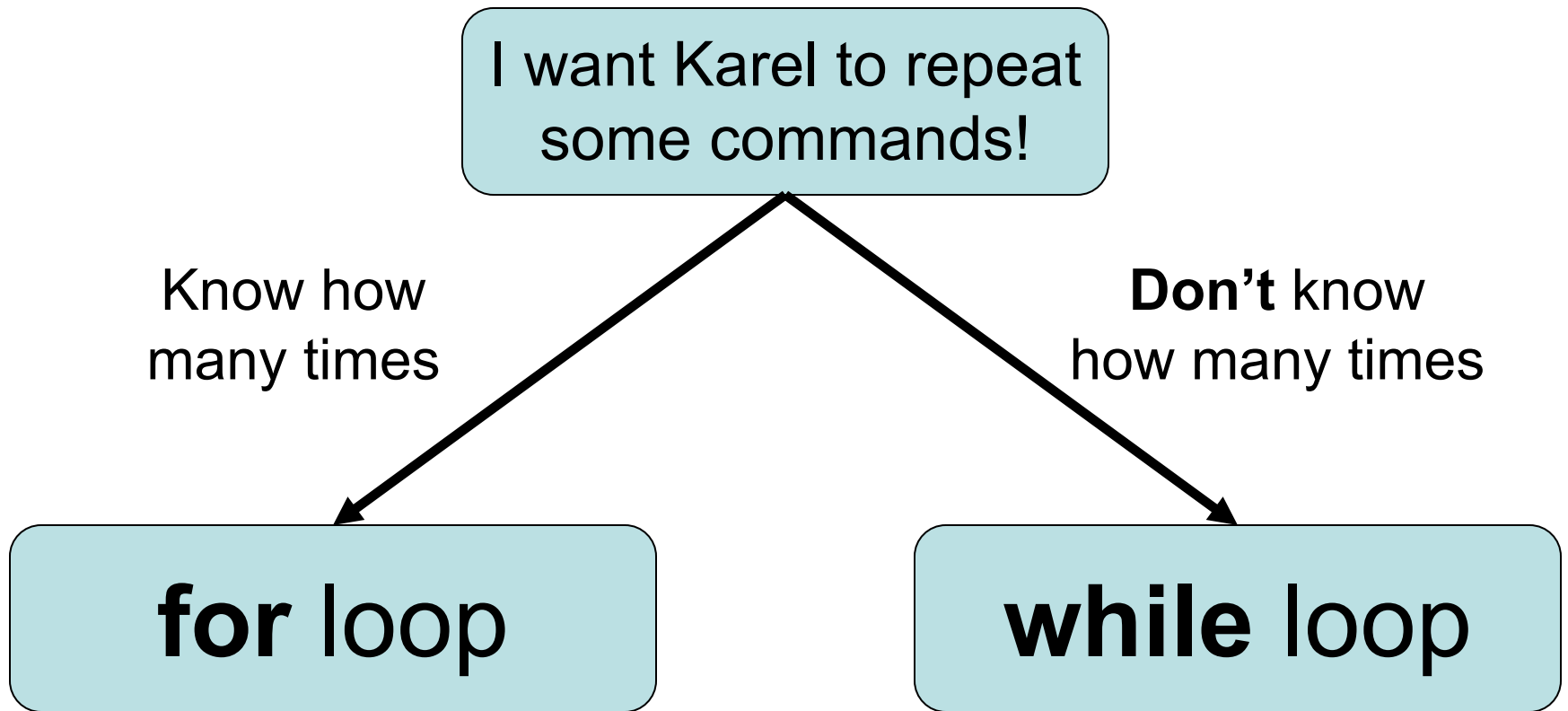
Now we can say:

```
while (frontIsClear()) {  
    move();  
}
```



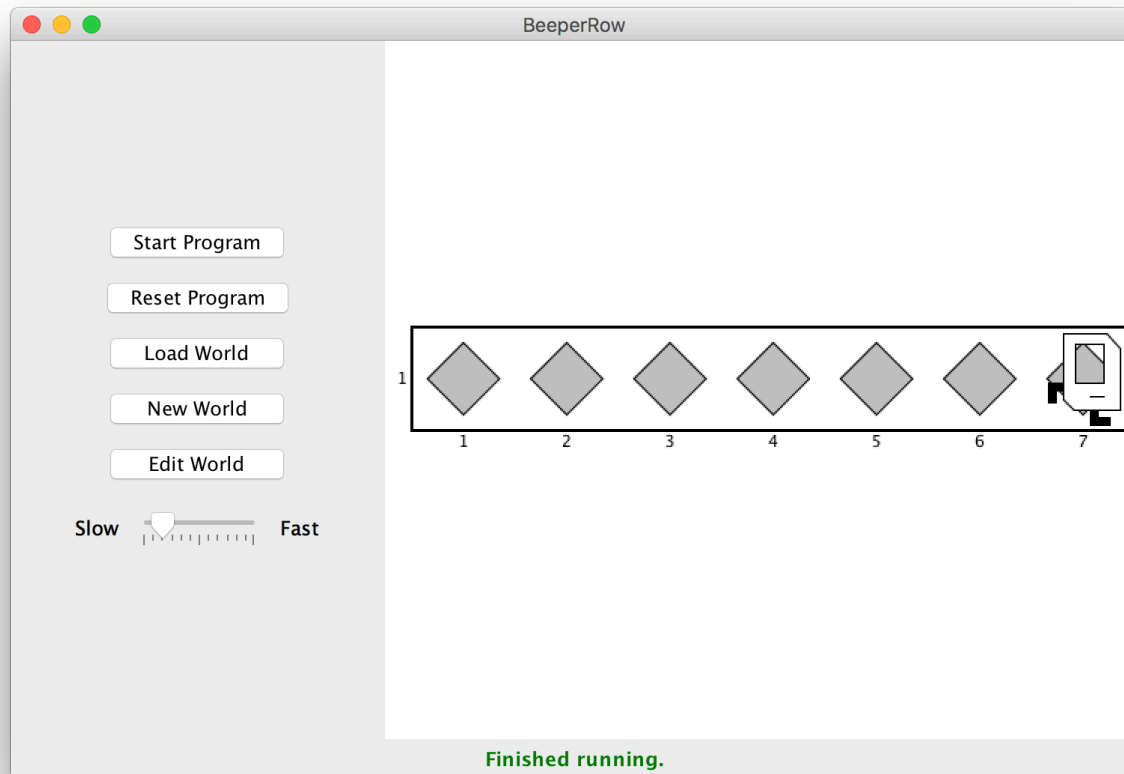
This is less repetitive, and it works in *any size* world!

Loops Overview



Loops Overview

I want Karel to put down a row of beepers until it reaches a wall. How do I do this?



Demo: BeeperLine

BeeperLine Solution

```
public void run() {  
    while (frontIsClear()) {  
        putBeeper();  
        move();  
    }  
  
    // put one last beeper once we reach a wall  
    putBeeper();  
}
```

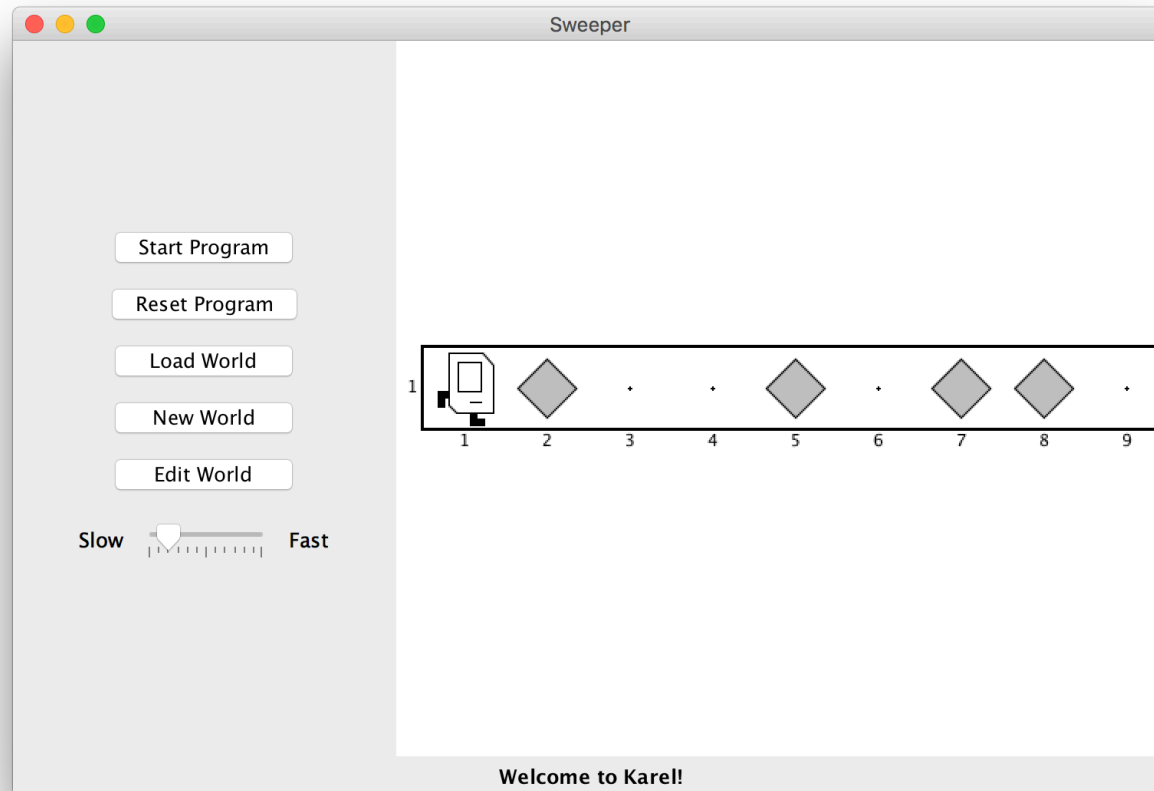
Plan For Today

- Karel the Robot So Far
- **Control Flow**
 - Reviewing For Loops
 - While loops
 - **If/else statements**



If/Else Statements

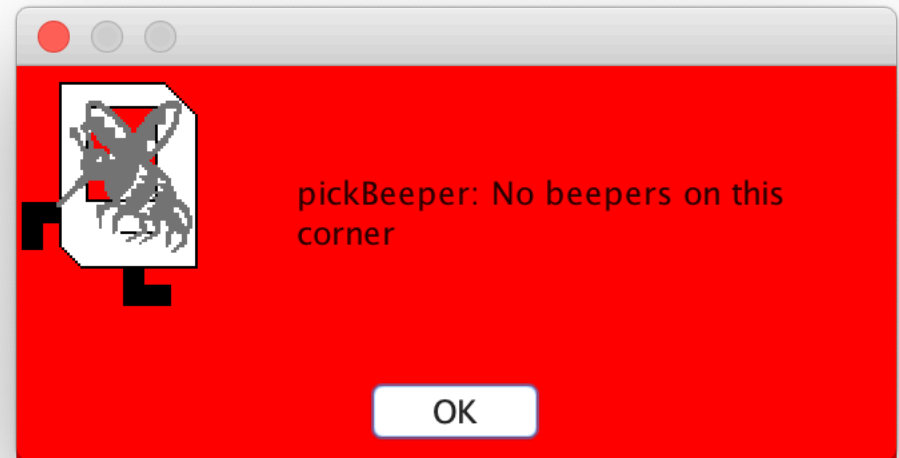
I want to make Karel clean up all beepers in front of it until it reaches a wall. How do I do this?



If/Else Statements

Can't just say:

```
while (frontIsClear()) {  
    move();  
    pickBeeper();  
}
```



This may crash, because Karel *cannot pick up beepers if there aren't any*. We don't **always** want Karel to pick up beepers; just when there is a beeper to pick up.

If/Else Statements

Instead, use an **if** statement:

```
if (condition) {  
    statement;  
    statement;  
    ...  
}
```

Runs the statements in the body *once* if *condition* is true.

If/Else Statements

You can also add an **else** statement:

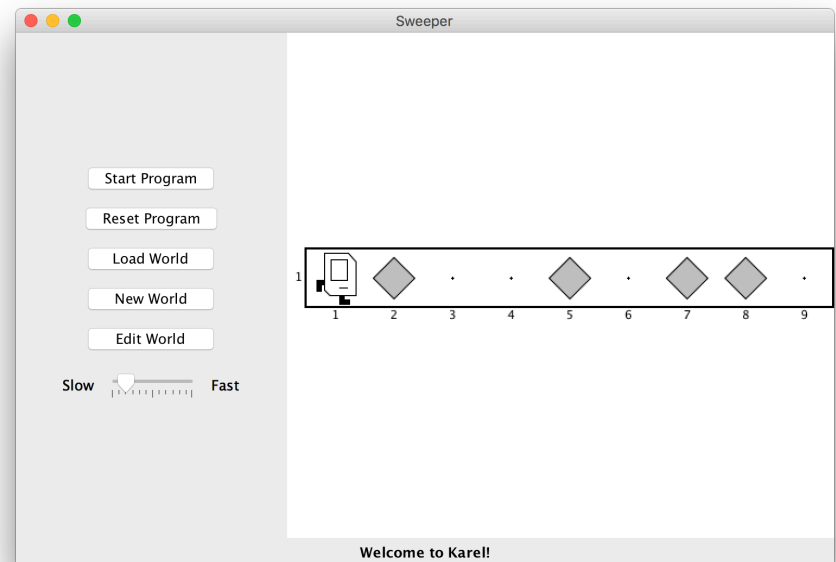
```
if (condition) {  
    statement;  
    statement;  
    ...  
} else {  
    statement;  
    statement;  
    ...  
}
```

Runs the first group of statements if ***condition*** is true; otherwise, runs the second group of statements.

If/Else Statements

Now we can say:

```
while (frontIsClear()) {  
    move();  
    if (beepersPresent()) {  
        pickBeeper();  
    }  
}
```



Now, Karel won't crash because it will only pickBeeper if there is one.

What Does This Do?

```
import stanford.karel.*;



public class IfExample extends SuperKarel {

    public void run() {
        if (beeperPresent()) {
            pickBeeper();
        } else {
            putBeeper();
        }
    }
}
```

Karel Resources

CS Bridge Handouts ▾ Projects ▾ Ex

- Karel Reader
- Karel Reference

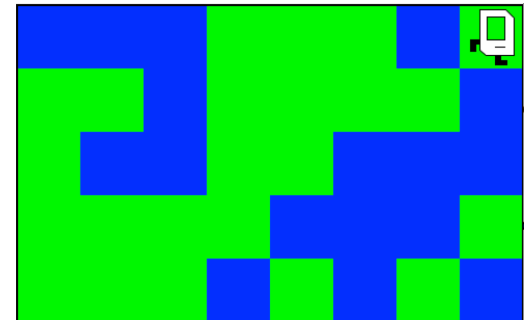
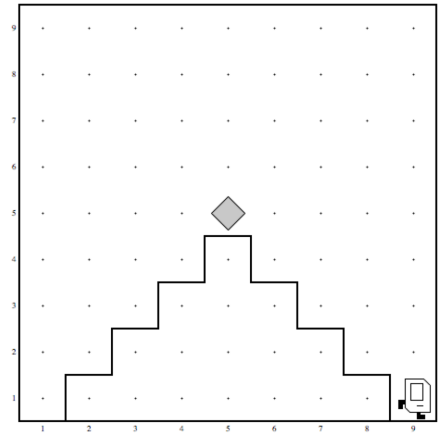
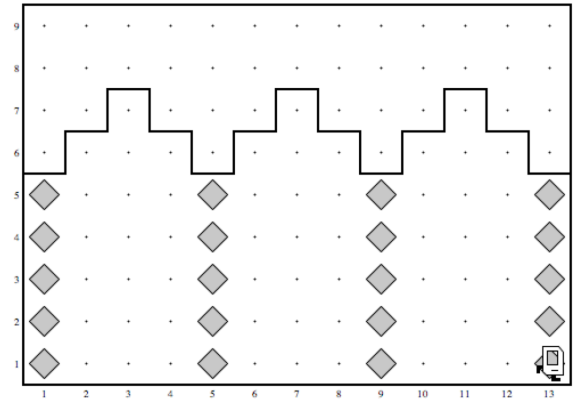


Micro to C

Summer 2019
July 9th to July 19th a

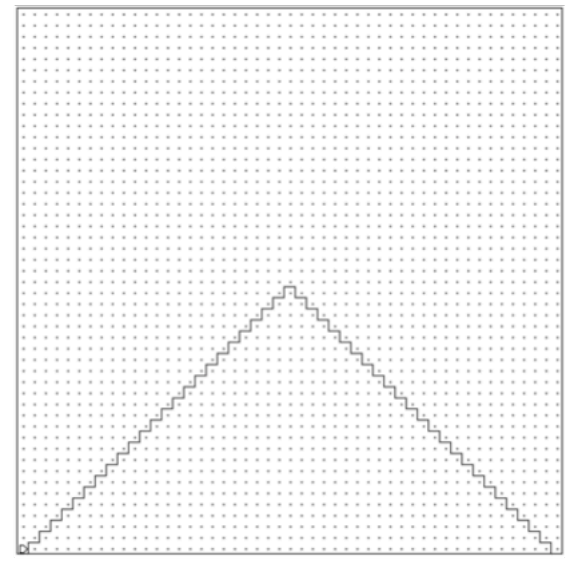
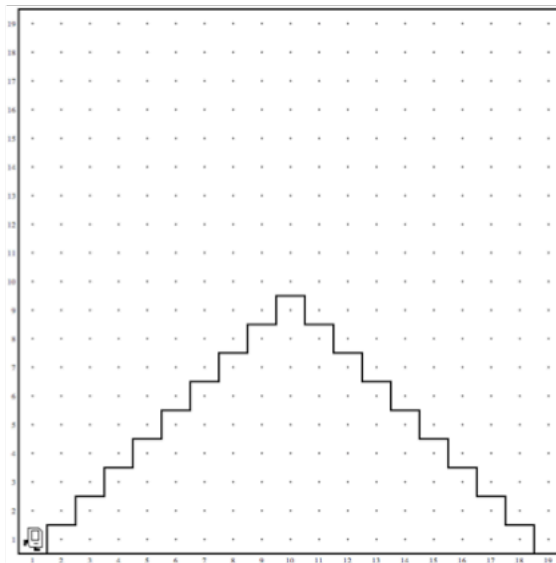
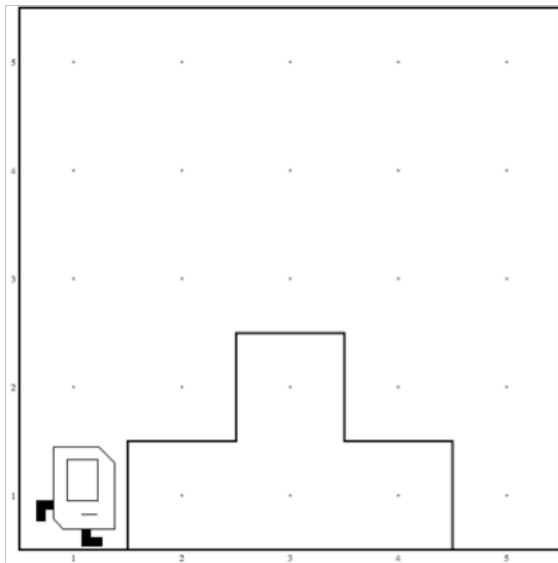
Rest Of Today

- Implement your own Build Charles Bridge program that you talked about in section
- Write a program “Mountain Karel” where Karel climbs a mountain of any size and places a beeper on top
- If you have time, write a program “Random Painter” where Karel paints any world randomly with green and blue squares.

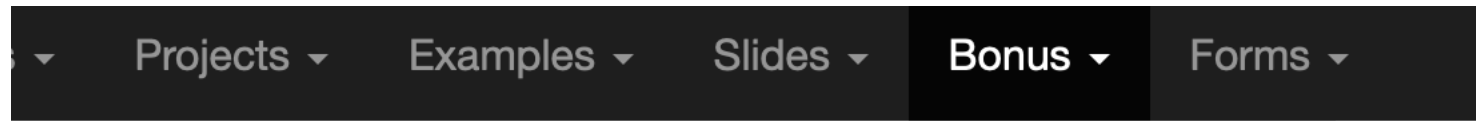


Mountain Karel

These programs should work on worlds of any size, as long as they obey the requirements given in the problem.



What If I Finish Early?



Bonus projects

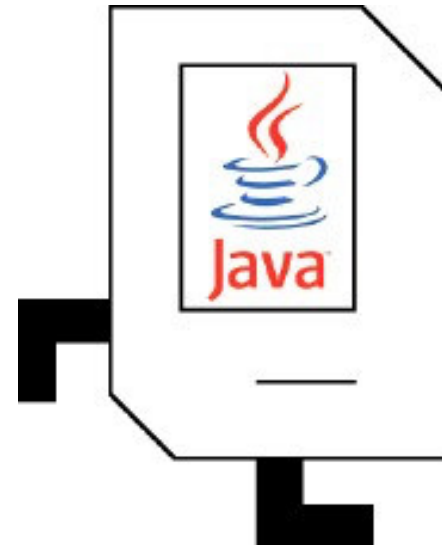
Intro to Computer Science

Summer 2019

July 9th to July 19th at Czech Technical University, Prague

Recap

- Karel the Robot So Far
- Control Flow
 - Reviewing For Loops
 - While loops
 - If/else statements



Tomorrow: transitioning from Karel to Java