CS 106A, Lecture 2
Programming with Karel

suggested reading:

Karel, Ch. 3-4
Plan For Today

• Announcements
• (Re)Meet Karel the Robot
• Control Flow
  – For loops
  – While loops
  – If/else statements
Plan For Today

• Announcements

• (Re)Meet Karel the Robot

• Control Flow
  – For loops
  – While loops
  – If/else statements
Announcements

• Section assignments
• Office Hours
• Lecture Feedback
• Extra Practice
Plan For Today

• Announcements
• (Re)Meet Karel the Robot
• Control Flow
  – For loops
  – While loops
  – If/else statements
Meet Karel the Robot!
Meet Karel the Robot!

Hello, world!
Karel's World
Streets (rows)

Each row is called a street.
Avenues (columns)

Each column is called an avenue.
The intersection of a street and an avenue is a corner.
Karel cannot move through walls.
Beepers mark locations in Karel's world. Karel can pick them up and put them down.
Karel Knows 4 Commands

- move
- turnLeft
- putBeeper
- pickBeeper
Karel Knows 4 Commands

- move
- turnLeft
- putBeeper
- pickBeeper

“methods”
- move makes Karel move forward one square in the direction it is facing.
- `move` makes Karel move forward one square in the direction it is facing.
Commands: turnLeft

- turnLeft makes Karel rotate 90° counter-clockwise.
- There is no turnRight command. (Why not?)
Commands: `turnLeft`

- `turnLeft` makes Karel rotate 90° counter-clockwise.
- There is no `turnRight` command. (Why not?)
- `pickBeeper` makes Karel pick up the beeper at the current corner. Karel can hold multiple beepers at a time in its "beeper bag".
- `pickBeeper` makes Karel pick up the beeper at the current corner. Karel can hold multiple beepers at a time in its "beeper bag".
Commands: putBeeper

- putBeeper makes Karel put a beeper down at its current location.
  - pickBeeper and putBeeper are used to move beepers around.
- `putBeeper` makes Karel put a beeper down at its current location.
  - `pickBeeper` and `putBeeper` are used to move beepers around.
Our First Karel Program

Before

1 2 3 4 5 6

1 2 3 4 5 6
Our First Karel Program

After

[Diagram of a grid with a robot and an area marked off]
Demo
We can make new commands (or methods) for Karel. This lets us decompose our program into smaller pieces that are easier to understand.

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

For example:

```java
private void turnRight() {
    turnLeft();
    turnLeft();
    turnLeft();
}
```
Plan For Today

• Announcements
• (Re)Meet Karel the Robot
• Control Flow
  – 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

Can’t just say:

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

This is too repetitive! Plus, it’s difficult to change (e.g. to 25 beepers).
Instead, use a `for` loop:

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

Repeats the statements in the body `max` times.
Now we can say:

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

This is less repetitive, and is easier to change (e.g. to 25 beepers).
Control Flow: For Loops

Some examples of using for loops:

```java
// turns Karel right
for (int i = 0; i < 3; i++) {
    turnLeft();
}

// Moves Karel in a square
for (int i = 0; i < 4; i++) {
    move();
    turnLeft();
}
```
Plan For Today

• Announcements
• (Re)Meet Karel the Robot
• Control Flow
  – For loops
  – While loops
  – If/else statements
I want Karel to move until it gets to a wall. How do I do this?
Control Flow: While Loops

Can’t just say:

```java
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.
Instead, use a `while` loop:

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

Repeats the statements in the body until `condition` is no longer true. Each time, Karel executes *all statements*, and then checks the condition.
### Possible Conditions

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

*This is Table 1 on page 18 of the Karel coursereader.*
Now we can say:

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

This is less repetitive, and it works in *any size* world!
while loops can have *compound* conditions as well:

```plaintext
// “and”
while (frontIsClear() && beepersPresent()) {
    ...
}

// “or”
while (leftIsClear() || rightIsClear()) {
    ...
}
```
Loops Overview

I want Karel to repeat some commands!

Know how many times

for loop

Don’t know how many times

while loop
Loops Overview

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

8 fence segments, but 9 posts!
The fencepost structure is useful when you want to loop a set of statements, but do one part of that set 1 additional time.

```java
putBeeper(); // post
while (frontIsClear()) {
    move(); // fence
    putBeeper(); // post
}

while (frontIsClear()) {
    putBeeper(); // post
    move(); // fence
}
putBeeper(); // post
```
Plan For Today

• Announcements
• (Re)Meet Karel the Robot
• Control Flow
  – 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:

```java
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.
Instead, use an `if` statement:

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

Runs the statements in the body once if `condition` is true.
If/Else Statements

You can also add an `else` statement:

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

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

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

• Announcements
• (Re)Meet Karel the Robot
• Control Flow
  – For loops
  – While loops
  – If/else statements

Next time: Karel Problem-Solving