Eric Roberts
CS 106A

Handout #12
January 11, 2016

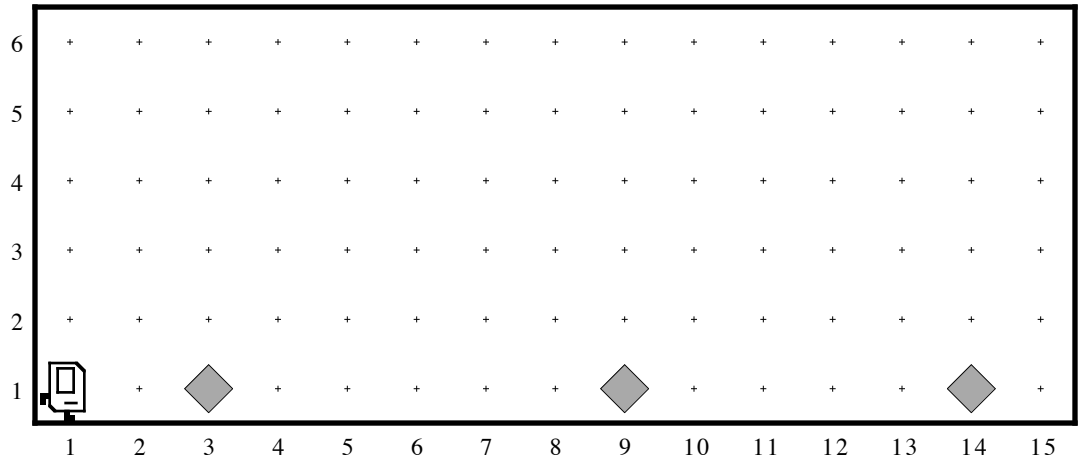
Section Handout #1—Karel and Simple Java

For this week's section, your goal is to solve a Karel problem that emphasizes stepwise refinement and to write some extremely simple Java programs, just to get you started.

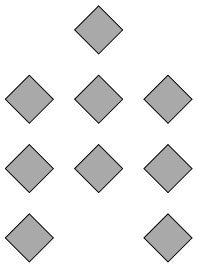
1. Repairing damage from Katrina

More than a decade after Hurricane Katrina, considerable damage remains along the Gulf Coast, and many communities have yet to be rebuilt. As part of its plans to improve the nation's infrastructure, the government has established a new program named Katrina Automated RELief (or KAREL) whose mission is to dispatch house-building robots to repair the damaged area. Your job is to program those robots.

Each robot begins at the west end of a street that might look like this:

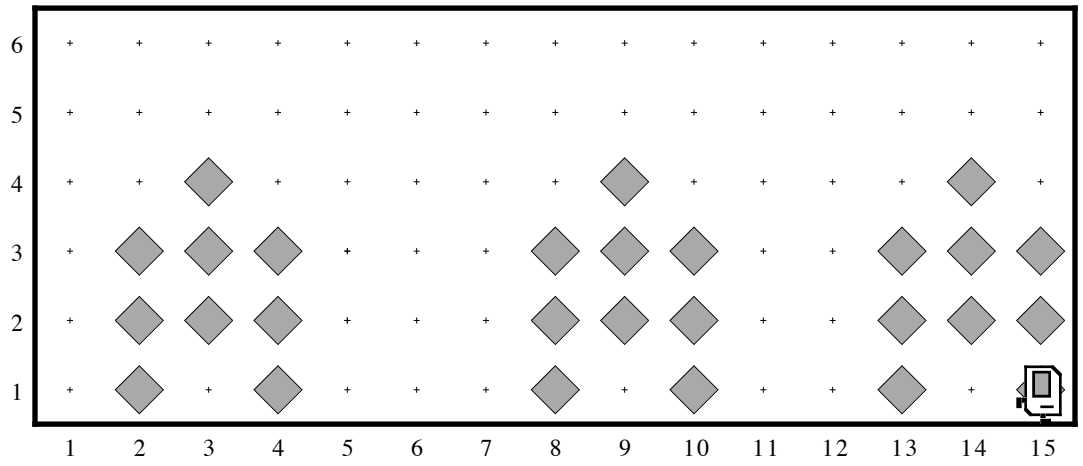


Each beeper in the figure represents a pile of debris where a house once stood. Karel's job is to walk along the street and build a new house in the place marked by each beeper. Those houses, moreover, need to be raised on stilts to avoid damage from the next storm. Each house, in fact, should look exactly like this:



The new house should be centered at the point at which the bit of debris was left, which means that the first house in the diagram above will be constructed with its left edge along 2nd Avenue.

At the end of the run, Karel should be at the east end of the street having created a set of houses that look like this for the initial conditions shown:

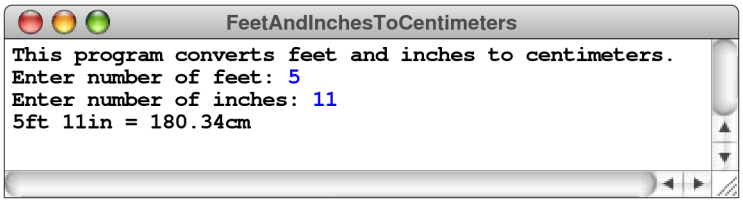


In solving this problem, you can count on the following facts about the world:

- Karel starts off facing east at the corner of 1st Street and 1st Avenue with an infinite number of beepers in its beeper bag.
- The beepers indicating the positions at which houses should be built will be spaced so that there is room to build the houses without overlapping or hitting walls.
- Karel must end up facing east at the southeast corner of the world. Moreover, Karel should not run into a wall if it builds a house that extends into that final corner.

2. Simple expressions (Chapter 3, exercise 1, page 91)

Extend the `InchesToCentimeters` program given in Figure 3-2 so that it reads in two input values: the number of feet, followed on a separate line by the number of inches. Here is a sample run of the program:



3. Precedence (Chapter 3, exercise 6, page 92)

In Norton Juster's *The Phantom Tollbooth,* the Mathemagician gives Milo the following problem to solve:

```
4 + 9 - 2 * 16 + 1 / 3 * 6 - 67 + 8 * 2 - 3 + 26 - 1 / 34 + 3 / 7 + 2 - 5
```

According to Milo's calculations, which are corroborated by the Mathemagician, this expression "all works out to zero." If you do the calculation, however, the expression comes out to zero only if you start at the beginning and apply all the operators in strict left-to-right order. What would the answer be if the Mathemagician's expression were evaluated using Java's precedence rules?