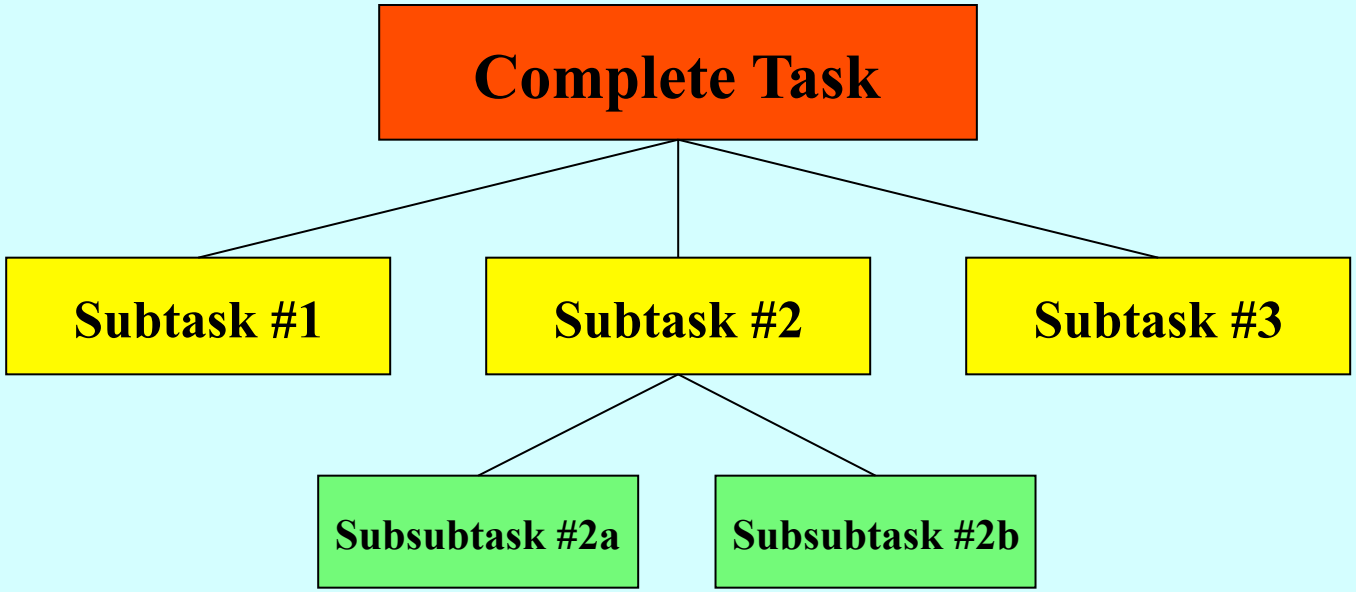
Stepwise Refinement

Jerry Cain
CS 106J
April 7, 2017

Stepwise Refinement

- The most effective way to solve a complex problem is to break it down into successively simpler subproblems.
- You start by breaking the whole task down into simpler parts.
- Some of those tasks may themselves need subdivision.
- This process is called ***stepwise refinement*** or ***decomposition***.



Criteria for Choosing a Decomposition

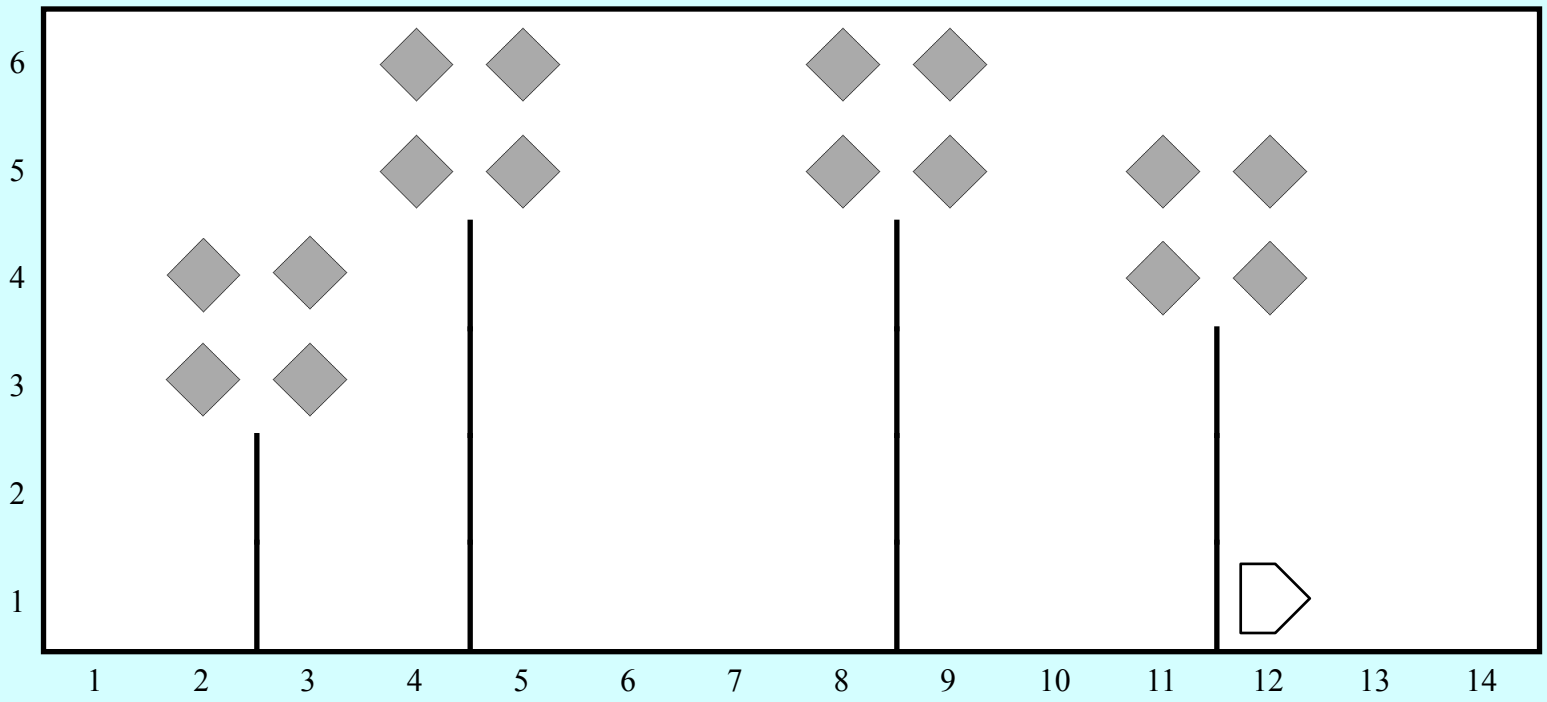
1. ***The proposed steps should be easy to explain.*** One indication that you have succeeded is being able to find simple names.

2. ***The steps should be as general as possible.*** Programming tools get reused all the time. If your methods perform general tasks, they are much easier to reuse.

3. ***The steps should make sense at the level of abstraction at which they are used.*** If you have a method that does the right job but whose name doesn't make sense in the context of the problem, it is probably worth defining a new method that calls the old one.

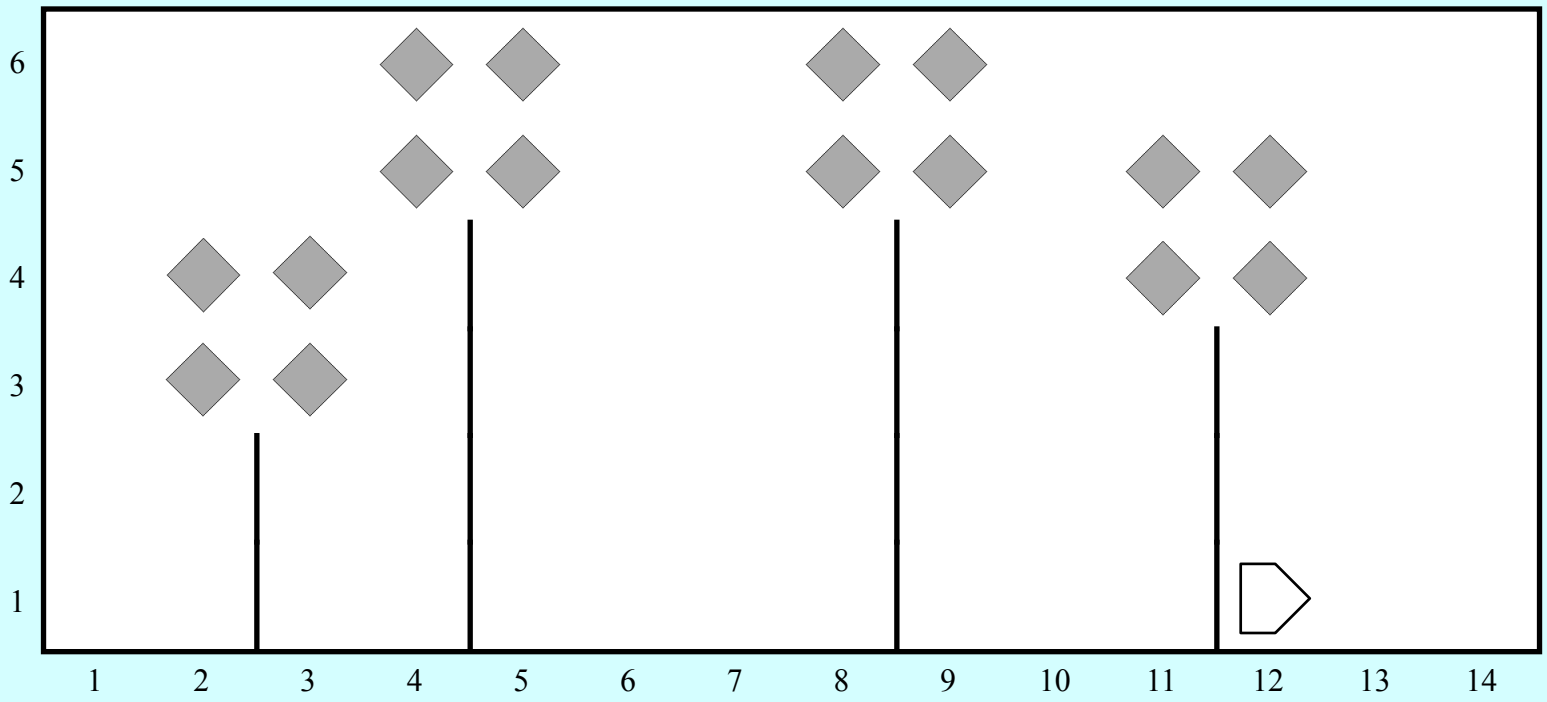
Exercise: Banishing Winter

- In this problem (which is described in detail in Handout #4), Karel is supposed to usher in springtime by placing bundles of leaves at the top of each "tree" in the world.
- Given this initial world, the final state should look like this:



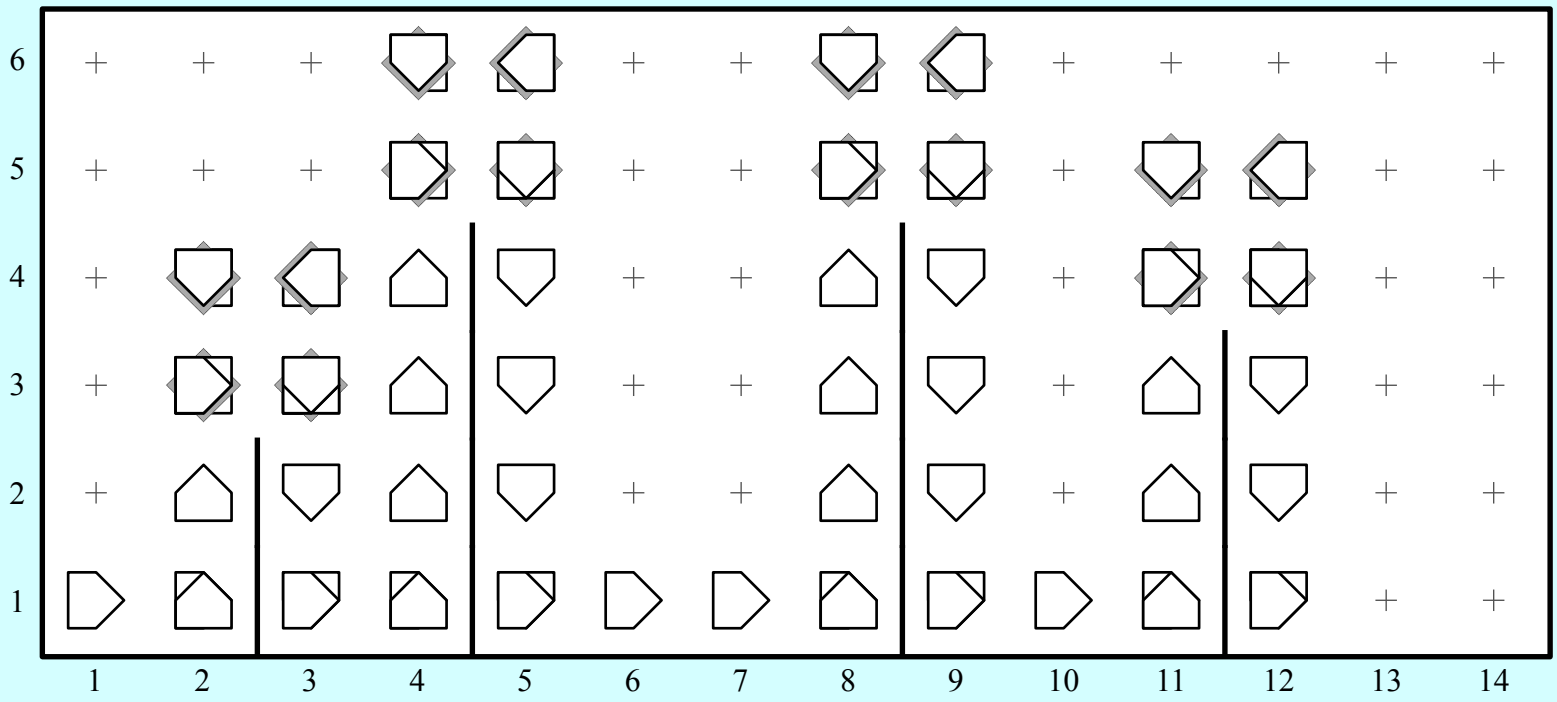
Understanding the Problem

- One of the first things you need to do given a problem of this sort is to make sure you understand all the details.
- According to the handout, Karel stops when it runs out of beepers. Why couldn't it just stop at the end of 1st Street?



The Top-Level Decomposition

- You can break this program down into two tasks that are executed repeatedly:
  - ☞ – Find the next tree.
  - ☞ – Decorate that tree with leaves.

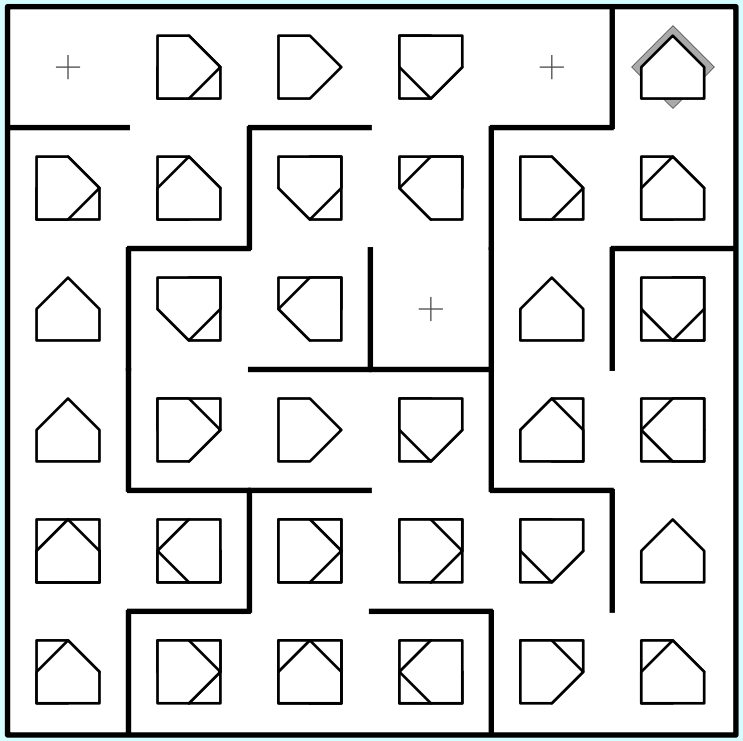


Preconditions and Postconditions

- Many of the bugs that you are likely to have come from being careless about the conditions under which you use a particular method.
- As an example, it would be easy to forget the **`turnLeft`** call at the end of the **`addLeavesToTree`** method.
- To reduce the likelihood of such errors, it is useful to define pre- and postconditions for every method you write.
  - A ***precondition*** specifies something that must be true before a method is called.
  - A ***postcondition*** specifies something that must be true after the method call returns.

Algorithmic Programming in Karel

- In computer science, a well-specified strategy for solving a problem is called an ***algorithm***.
- The goal for the rest of this lecture is to write a program that solves a maze using the ***right-hand rule***:



The End