Karel The Robot

The way I solved and optimized the problem.

Duaa Alsaaideh
Java and DevOps Training - Fall 2021

Supervisor
Motasem Aldiab & Fahed Jubair
November 2021

Table of contents

List of Figures

Introducing Karel the Robot

Karel is a very simple robot living in a very simple world. By giving Karel a set of commands, you can direct it to perform certain tasks within its world. Karel's world contains streets (horizontal lines) and avenues (vertical lines). The intersection of a street and an avenue is called a corner. Karel's bag contains an infinite number of beepers, so karel can use them as shown in figure 1.1 below. Karel can only be positioned on corners and must be facing one of the four standard compass directions (north, south, east, west).

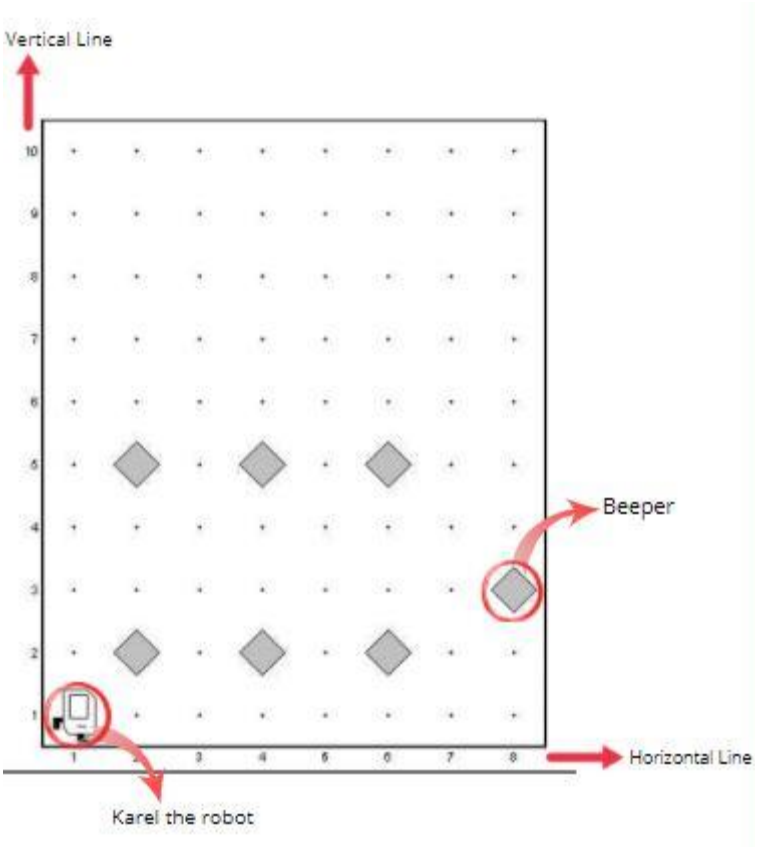


Figure 1.1: information about karel

Definition of the problem

Karel must divide any given world of any size into four equal chambers using beepers. when it is impossible to divide them into four equal chambers Karel can divide them into two equal chambers. If the map has an odd number of vertical and horizontal lines ex: 7 * 7 or 9 * 9, then Karel needs to put beepers on the diagonals also. Karel should return to the origin point after achieving its task.

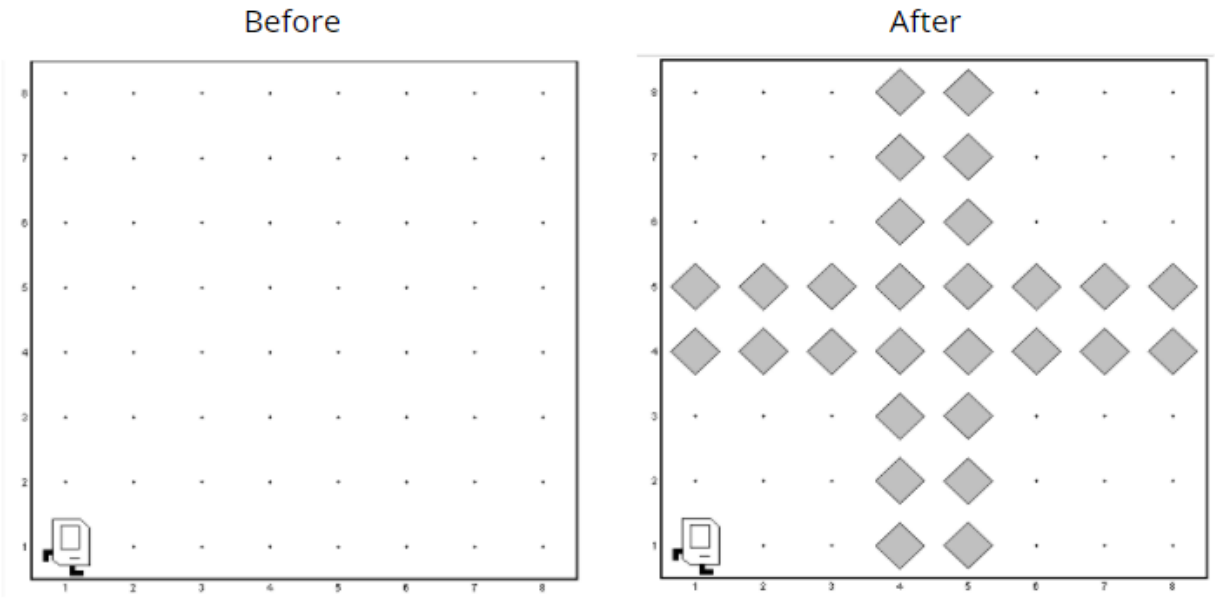


Figure 1.2: Solution Example

This problem should be solved along with the following constraints:

1. Karel should achieve his task with the lowest number of moves.
2. Karel should use the minimum number of resources (beepers) from his own beepers.
3. The code should be minimized to the lowest possible number of lines.

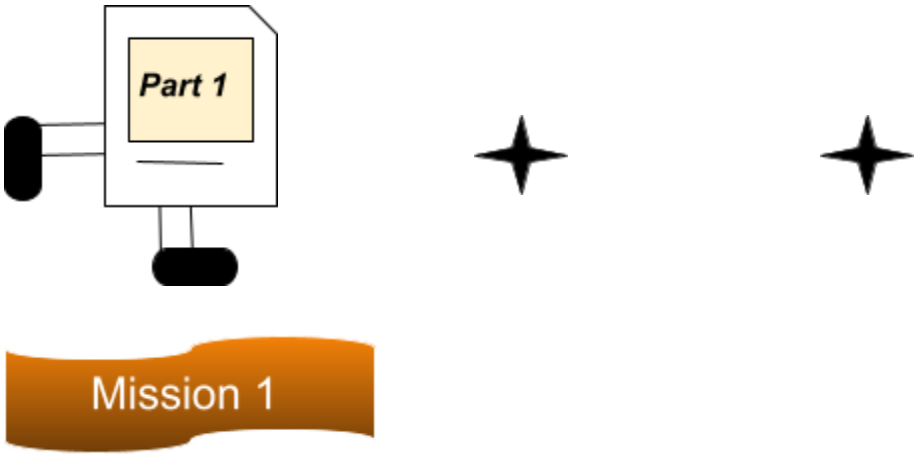
Solution of the problem

Each problem should be divided into subproblems and each subproblem should perform a conceptually simple task. And this does not mean that it will not be complex and may require many lines of code, most of the time it will. Even so, it should end up achieving a task that is easy to describe and is as general as possible. And from this concept, I divided Karel's problem into subproblems.

Firstly, the problem consisted of two parts:

- **Part 1:**
  - Assume the map is not clean from beepers and Karel's first mission will be cleaning the map.
  - Mission two Karel should get the world dimensions, so the solution can be applied to any other world.
  - Mission three Karel will divide the map into equal chambers along with using logic. Some world's dimensions will require deciding whether to use four or two chambers or less.
  - Mission four will be returning Karel to its original point.
- **Part 2:**
  - Assume the map is cleaned from beepers. So Karel's first mission will be getting the world dimensions.
  - Mission two will be dividing the map.
  - Mission three will be returning Karel to its original point.

Assuming that Karel is located at the corner of 1st Street and 1st Avenue, facing east. The run() method calls three methods which are the setBeepersInBag() which initiates beepers in Karel's bag, part1() method and part2() method. Part1() method calls at first the cleanMap() method.



Cleaning the map:

1. **Calling cleanRow() method.**
    - It cleans row by row by picking all the present beepers on Karel's way and by calling two methods checkBeepers() that checks if beepers are present to pick them and counts the picked beepers, and the keepPicking() method which keeps picking beepers while moving.
2. **Commands in the cleanMap method.**
    - Using while loop and some if statements along with calling two methods TurnFaceToWest() which turns Karel's face if it's left is clear and then

clears the other row and TurnFaceToEast() which turns Karel's face if it's right is clear and cleans the other row.

3. **Returning home.**

- After cleaning the whole map Karel returns to its origin point, by calling backHome() method, which keepMoving and turning left if Karel is facing west until it back to the origin point facing east. The backHome() method calls another method called keepMoving() which keeps moving if the karel front is clear, in addition to this keepMoving() method which calls countMoves() method that applies the move command and counts Karel's moves.

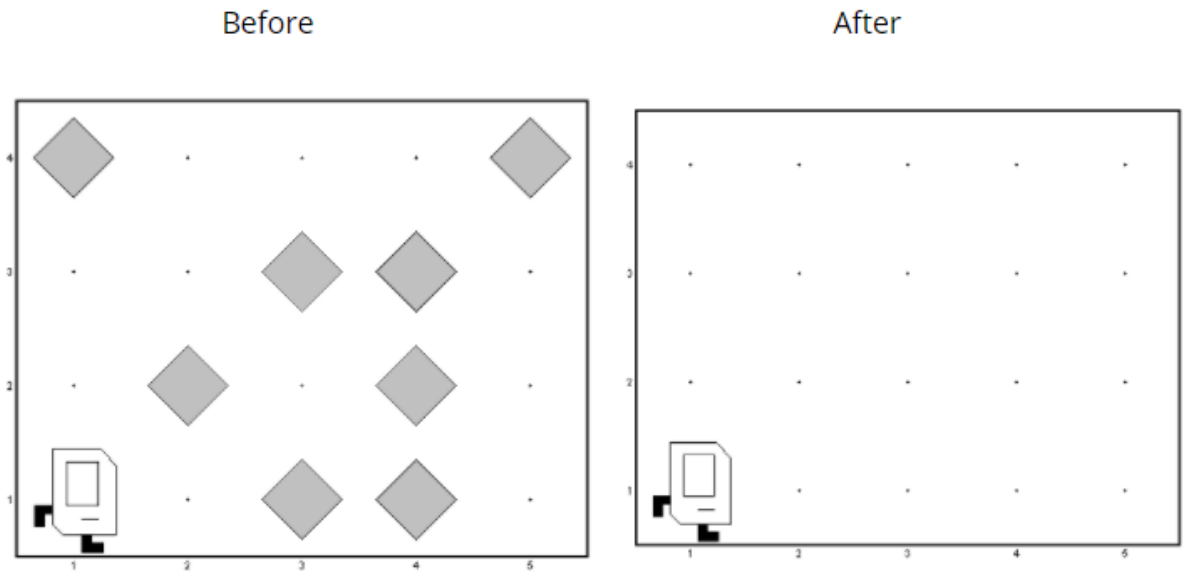


Figure 1.3: Clean Map example

The second method that part1() calls is doTask() which applies mission2 and mission3. The doTask() method calls get_world_dimensions() method and stores the result in a world_dimensions array, after printing on screen the world dimensions it applies the backHome() method and then applies mission three which contains calling the divide_map() method.

Getting the world dimensions:

1. **get_world_dimensions() method.**
    - This method returns an integer array called dimensions, by counting Karel's steps on the x-axis and the y-axis.

★ Karel returns home after getting the world dimensions.

Dividing the map:

1. **The doTask() method calls divide_map() method and sends the world dimensions array as a parameter.**
    - Contains many if and else statements that specify the even and odd dimensions with some special cases, each conditional statement calls a specific method to divide either x-axis or y-axis, for example: if the x-axis is odd and bigger than two its will call a method called divide_odd_x(x-axis).

- The divide_odd_x(x-axis) method call a method called stepsToMove() which takes an integer as a parameter and keeps Karel moving using a for loop starting from zero to the required steps, it also calls another method called putLeftLine() which keep putting beepers on the left line by calling other methods.

2. **The divide_map() method calls a divide_diagonal() method if both x and y axes are odd and equal.**
    - divide_diagonal() method takes one of the axes as a parameter and with other commands it let Karel puts beepers on diagonal intersections also, it calls a two methods rowDowenLeft() and rowDownRight() both let Karel move a row down from left and right.

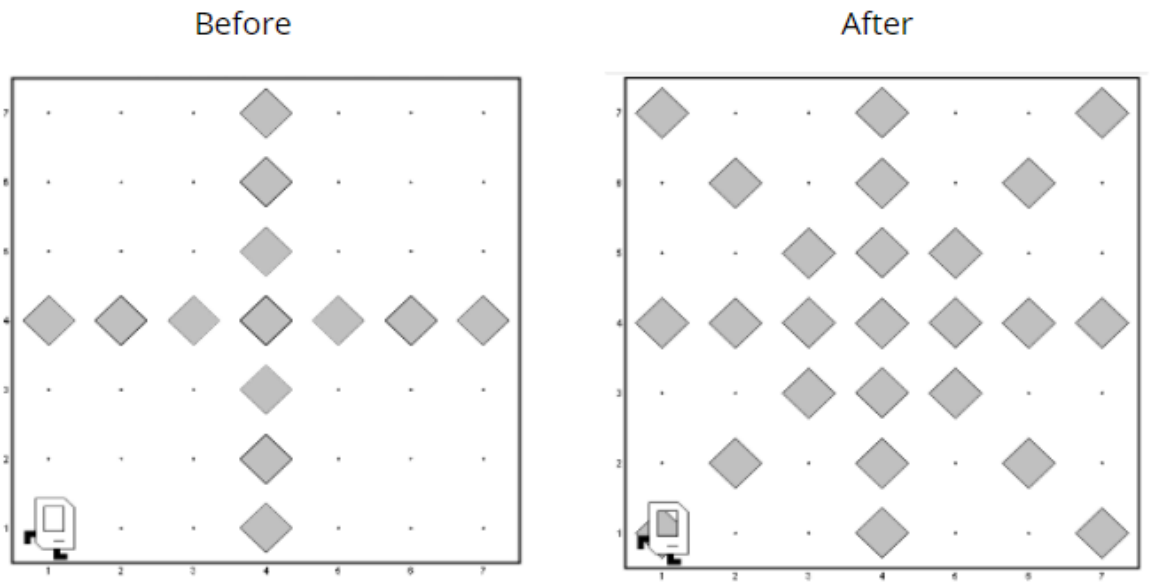


Figure 1.4: Fill diagonal example

Mission 4

Returning home:

- After dividing the map Karel returns to its origin point, by calling backHome() method.

Part 2 has three missions just like the last three missions in part 1, so the method part2() will call the method doTask() and this method will do the same as explained before. As mentioned before, assuming that the map is clean from beepers.

Optimizations that have been done:

- Karels moves have been decreased to their lowest number, and this means that every move Karel has done was for a purpose. At first, Karel started to move row by row in order to check for beepers and clean the map. Secondly, Karel moved

on the axes in order to get the world dimensions, and lastly, Karel moved to divide the map and return home.

- The picked beepers can be reused again in dividing the map.
- Reusable functions were written, to reduce the number of code lines.
- In my code, I did not find recursion useful in optimizing my code.

Conclusion

These kinds of problems have changed my way of thinking a lot and give me the chance to try many ways till obtaining the best. I recommend to use Karel the robot in teaching programming everywhere.

References

1. The full code: Duaa Alsaaideh 2021, Google Drive, uploaded 6 November 2021, <Homework.java>
2. Eric, R (2005, September). Karel the robot learns java, *Department of Computer Science, Stanford University.*