Problem-solving and Optimizations for Dividing a Map in Karel

Name: Zaid saadedin

Position: Trainer

Organization: Atypon

Date: 2023/5/31

Overview

Dear DR Motasem, I am pleased to submit the following detailed report for highlighting the problem-solving steps and optimizations algorithm that I used to divide a map of Karel world, as requested. This report provides a comprehensive overview of the solution approach, including the code snippets and the optimizations made to ensure efficiency.

Introduction

The provided code is a Java program that aims to solve the problem of dividing a given map in Karle world into 4 + 4 using the SuperKarel class. The problem-solving steps involve dividing and drawing chambers within a map based on certain conditions, such as:

1. The inner chambers should be the biggest possible equal's squares.
2. The outer chambers should be equal in size, and they should be L-shaped (they should not be rectangles or squares).
3. You are allowed to use double lines of beepers if you need to, however, you need to observe that beepers use should be optimized.

Understanding the Problem

The initial step in problem-solving is to thoroughly understand the problem statement and requirements, at first I read the Karel pdf that was given in the assignment, and understood what the Karel functionalities and how he works, then I start to think of the problem and used paper and pen to draw the map to came up with the right logic to divide the map and draw the peepers.

1. Analyzing the Code

After understanding the problem, I analyze the problem, I start by thinking of the methods that I will need to implement, and then I started to think about how to solve the problem using these methods and override what is needed of these method, I needed a formula and a logic to work based on it. After many hours of analyzing I discovered the formulas and the logic needed.

2. Defining the Problem

Based on the assignment given, I can define the problem as follows:

The problem is to divide the outer chambers of a map based on the number of streets and avenues. The division should consider various conditions, such as even/odd number of streets or avenues, and whether they are equal or not. After division, the inner chambers should be drawn in a specific pattern using beepers.

Problem-Solving Approach

1. Initializing Variables and helper methods

I initialized the necessary variables, such as:

1) beepers
2) movesCounter
3) streets
4) avenues
5) beepersUsed
6) stepsCounter
7) streetInitialPoint
8) avenuesInitialPoint
9) initialPoints
10) streetMidPoint
11) avenuesMidPoint

Then I overrided these methods:

1) move() method to count the moves
2) putBeeper() method to decrement the beepers in the bag

Then I initializes the necessary helper methods, such as:

1) countStreet()
2) countAvenues()
3) returnHome()
4) checkIfBoundLengthIsEven(int bound)
5) checkIfBoundsIsLargeEnough()
6) checkBoundsIfEqual()
7) drawLineToDivideTheMap()
8) setTheValuesToDrawInnerChampersIfBothOddOrEvenAndAvenuesBiggerThanStreets()
9) setTheValuesToDrawInnerChampersIfBothOddOrEvenAndStreetsBiggerThanAvenues()
10) drawInnerChampersIfBothOddOrEven()
11) drawInnerChampersIfAvenuesOddAndStreetsEvenOrAvenuesEvenAndStreetsOddWitherStreetsLongerOrAvenues(int lengthOfTheSquareSideForAvenues,int lengthOfTheSquareSideForStreets)

2. Counting Streets and Avenues

Then counts the Length of streets and avenues present in the map by moving Karel forward until no wall then turn left and move until no wall. This step helps determine the size of the map.

3. Dividing the Map

Based on the conditions, such as even/odd streets or avenues, the code divides the outer chambers of the map. It achieves this by moving Karel to specific positions and placing beepers at each step where there is no beeper presents.

4. Drawing Inner Chambers

After dividing the outer chambers, the code proceeds to draw the inner chambers within the map. The drawing process follows a specific pattern and considers various cases, such as even/odd streets or avenues and their equality.

5. Reporting Results

Finally, the code reports the results of the problem-solving process. It prints the values of movesCounter, beepers, avenues, and streets to provide information on the execution of the program.

Conclusion

The "Homework" code effectively solves the problem of dividing and drawing chambers within a map using the SuperKarel class. It follows a systematic approach, considering various conditions, to achieve the desired outcome. By analyzing and understanding the problem-solving steps outlined in this report, you can gain a comprehensive understanding of the code's functionality and its solution to the problem at hand.