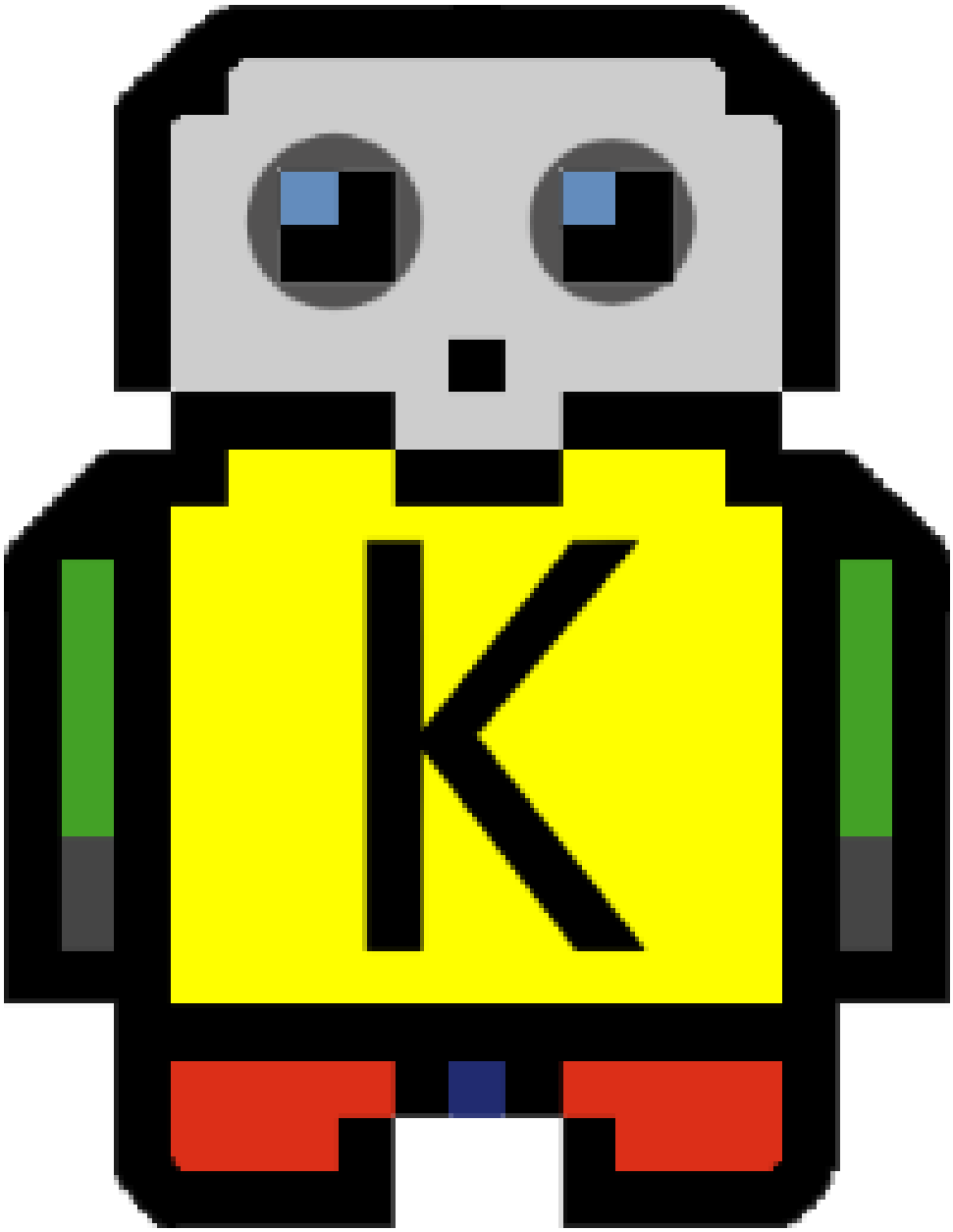


Karel Robot

Table of Contents

1. Introduction 3
2. Code Overview 3
3. Methods 3
   3.1 run() 3
   3.2 putBeepersInColumnsAndRows() 3
   3.3 makeColumn(int columnCount, int rowCount) 5
   3.4 makeRectangle(int mid, int columnCount, int rowCount) 7
   3.5 makeRectangle(int mid, int columnCount, int rowCount) 9
4. Conclusion 10

1. Introduction

The Homework class is a Karel program that divides a given map into four equal-sized chambers, with four additional outer chambers in an L-shaped pattern. This program uses the Stanford Karel library.

2. Code Overview

The Homework class extends the SuperKarel class and inherits its functionality. It adds additional methods to implement the desired map division.

3. Methods

3.1 `run()`

The `run()` method is the entry point of the program. It calls the `putBeepersInColumnsAndRows()` method to divide the map into chambers and then prints the total number of steps taken.

```
public void run() {
    putBeepersInColumnsAndRows(); // Main function to put beepers in columns and rows
    System.out.println(stepsCounter); // Print the total number of steps taken
}
```

3.2 `putBeepersInColumnsAndRows()`

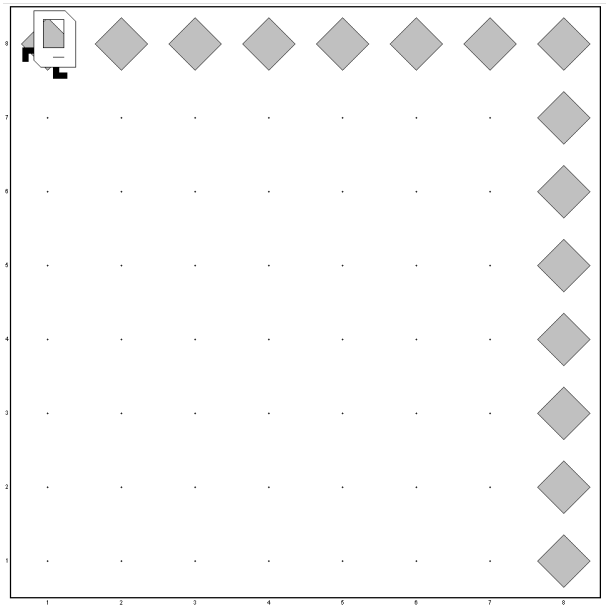
The `putBeepersInColumnsAndRows()` method is responsible for dividing the map into columns and rows and placing beepers accordingly. It initializes the column and row counts, and then traverses the map to determine the size of the grid. It puts beepers in the columns and rows while counting the number of steps taken. Based on the column and row counts, it decides

whether to create an additional beeper and whether the grid is large enough to create inner chambers. It calls the `makeColumn()` and `makeRow()` methods to create the desired chamber structure.

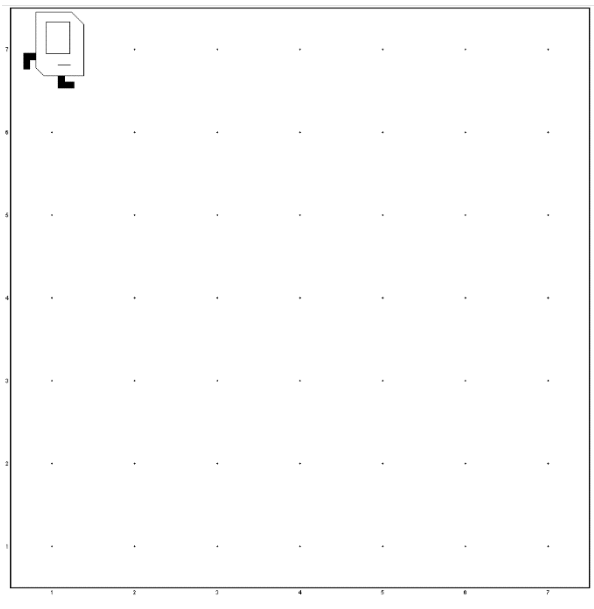
```
public void putBeepersInColumnsAndRows() {
    stepsCounter = 0;
    int columnCount = 1; // Initialize column count to 1
    int rowCount = 1; // Initialize row count to 1

    // Move forward until there is an obstacle in front
    while (frontIsClear()) {
        move();
        stepsCounter++;
        columnCount++; // Increment column count
    }
    turnLeft();

    // Check if the column count is even
    if (columnCount % 2 == 0) {
        // Move forward, put beepers, and increment row count until there is an obstacle in front
        while (frontIsClear()) {
            rowCount++;
            putBeeper();
            move();
            stepsCounter++;
        }
```

```
    putBeeper();
    turnLeft();
} else {
    // Move forward and increment row count until there is an obstacle in front
    while (frontIsClear()) {
        rowCount++;
        move();
        stepsCounter++;
    }
    turnLeft();
}

// Check if the row count is even
if (rowCount % 2 == 0) {
    if (columnCount % 2 != 0) {
        putBeeper();
    }
    // Move forward, put beepers, and increment steps counter until there is an obstacle in front
    while (frontIsClear()) {
        move();
        stepsCounter++;
        putBeeper();
```

```
    }
} else {
    // Move forward and increment steps counter until there is an obstacle in front
    while (frontIsClear()) {
        move();
        stepsCounter++;
    }
}
turnAround();
makeColumn(columnCount, rowCount); // Create columns in the grid
```



8X8



7X7

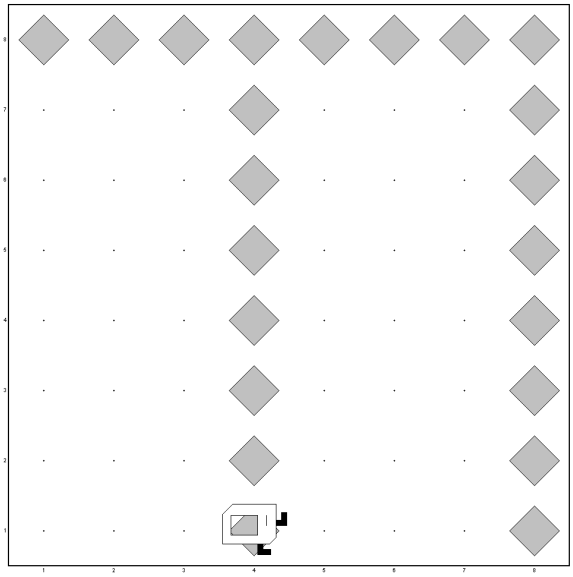
3.3 `makeColumn(int columnCount, int rowCount)`

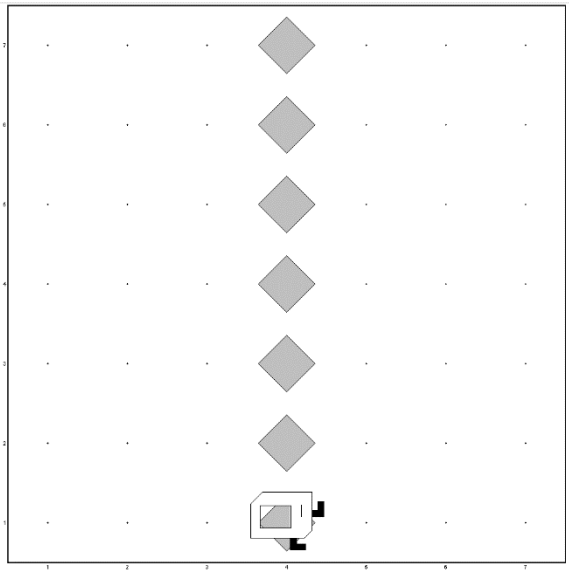
The `makeColumn()` method is responsible for creating columns in the grid. It takes the column and row counts as parameters. It calculates the midpoint of the column and moves Karel to reach the middle position. It puts a beeper if the row count is odd. Then, it moves forward, puts beepers, and increments the steps counter until there is an obstacle in front. It turns around and

adjusts the column and row counts if they are even. If the grid is large enough, it calls the `makeRectangle()` method to create inner chambers. Finally, it calls the `makeRow()` method to create a row in the grid.

```
public void makeColumn(int columnCount, int rowCount) {
    int mid = (int) Math.ceil(columnCount / 2.0);

    // Move forward to reach the middle of the column
    for (int i = 1; i < mid; i++) {
        move();
        stepsCounter++;
    }
    turnRight();

    // Put a beeper if the row count is odd
    if (rowCount % 2 != 0)
        putBeeper();

    // Move forward, put beepers, and increment steps counter until there is an obstacle in front
    while (frontIsClear()) {
        move();
        stepsCounter++;
        putBeeper();
    }
    turnAround();
```

```
    // Adjust column and row counts if they are even
    if (columnCount % 2 == 0)
        columnCount -= 1;
    if (rowCount % 2 == 0)
        rowCount -= 1;

    // Check if the grid is large enough to create inner chambers
    if (rowCount >= 7 && columnCount >= 7) {
        makeRectangle(mid, columnCount, rowCount); // Create inner chambers in the grid
    } else {
        move();
        stepsCounter++;
        System.out.println("The inner chambers can't be done because the map is too small!");
    }
    makeRow(rowCount); // Create a row in the grid
}
```





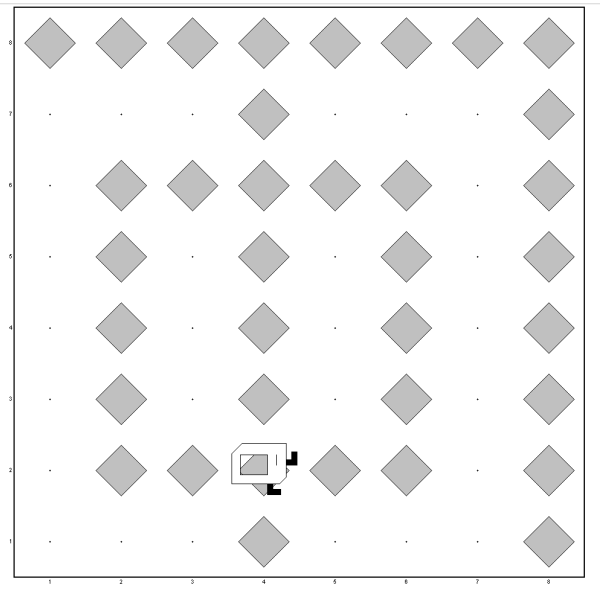
3.4 `makeRectangle(int mid, int columnCount, int rowCount)`

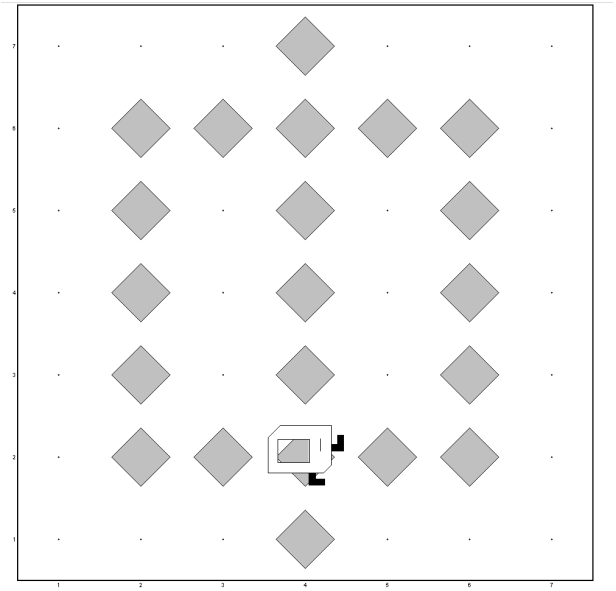
The `makeRectangle()` method is responsible for creating the inner chambers in the grid. It takes the midpoint, column count, and row count as parameters. It moves Karel forward, turns right, and then moves forward while putting beepers until reaching `columnCount - 2`. It turns left and moves forward while putting beepers until reaching `rowCount - 1`. It turns left again and moves backward while putting beepers until reaching 2. It turns left once more and moves forward while putting beepers until reaching `rowCount - 1`. Finally, it turns left and moves forward while putting beepers until reaching `mid - 1`.

```
public void makeRectangle(int mid, int columnCount, int rowCount) {
    move();
    stepsCounter++;
    turnRight();

    // Move forward, put beepers if none present, and increment steps counter until reaching columnCount-2
    for (int i = mid; i <= columnCount - 2; i++) {
        move();
        stepsCounter++;
        if (!beepersPresent())
            putBeeper();
    }
    turnLeft();

    // Move forward, put beepers if none present, and increment steps counter until reaching rowCount-1
    for (int i = 2; i < rowCount - 1; i++) {
        move();
        stepsCounter++;
        if (!beepersPresent())
            putBeeper();
    }
    turnLeft();
```

```
  // Move backward, put beepers if none present, and increment steps counter until reaching 2
  for (int i = columnCount - 1; i > 2; i--) {
      move();
      stepsCounter++;
      if (!beepersPresent())
          putBeeper();
  }
  turnLeft();

  // Move forward, put beepers if none present, and increment steps counter until reaching rowCount-1
  for (int i = 2; i < rowCount - 1; i++) {
      move();
      stepsCounter++;
      if (!beepersPresent())
          putBeeper();
  }
  turnLeft();
```

```
    // Move forward, put beepers if none present, and increment steps counter until reaching mid-1
    for (int i = 2; i < mid - 1; i++) {
        move();
        stepsCounter++;
        if (!beepersPresent())
            putBeeper();
    }
    move();
    stepsCounter++;
    turnLeft();
}
```



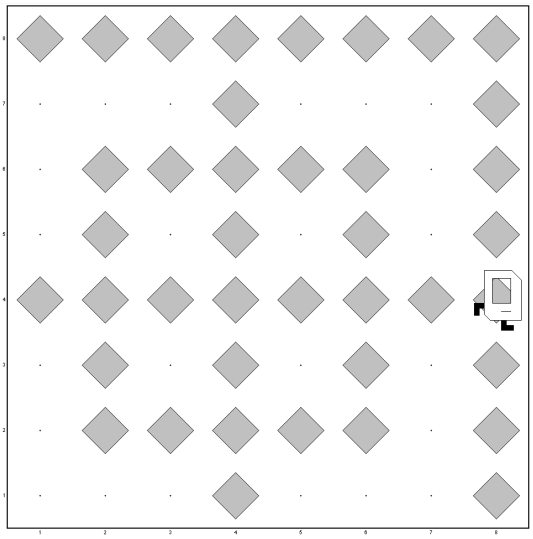


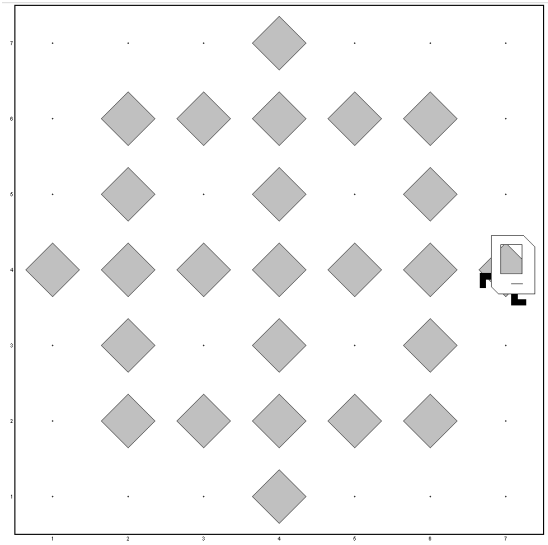
3.5 makeRectangle(int mid, int columnCount, int rowCount)

The `makeRow()` method is responsible for creating a row in the grid. It takes the row count as a parameter. It calculates the midpoint of the row and moves Karel to reach the middle position. It moves forward, puts beepers, and increments the steps counter until there is an obstacle in front. It turns around and moves backward while putting beepers until there is an obstacle behind.

```java
public void makeRow(int rowCount) {
    int mid = rowCount / 2;

    // Move forward to reach the middle of the row
    for (int i = 2; i <= mid; i++) {
        move();
        stepsCounter++;
    }
    turnLeft();

    // Move forward, put beepers if none present, and increment steps counter until there is an obstacle in front
    while (frontIsClear()) {
        move();
        stepsCounter++;
        if (!beepersPresent())
            putBeeper();
    }
    turnAround();
```

```java
    // Move backward, put beepers if none present, and increment steps counter until there is an obstacle behind
    while (frontIsClear()) {
        move();
        stepsCounter++;
        if (!beepersPresent())
            putBeeper();
    }
}
```





4. Conclusion

In conclusion, the Homework class is a program that utilizes the Karel programming language to divide a given map into chambers using beepers. The program is designed to systematically move through the map, placing beepers in specific patterns to create the desired chambers.