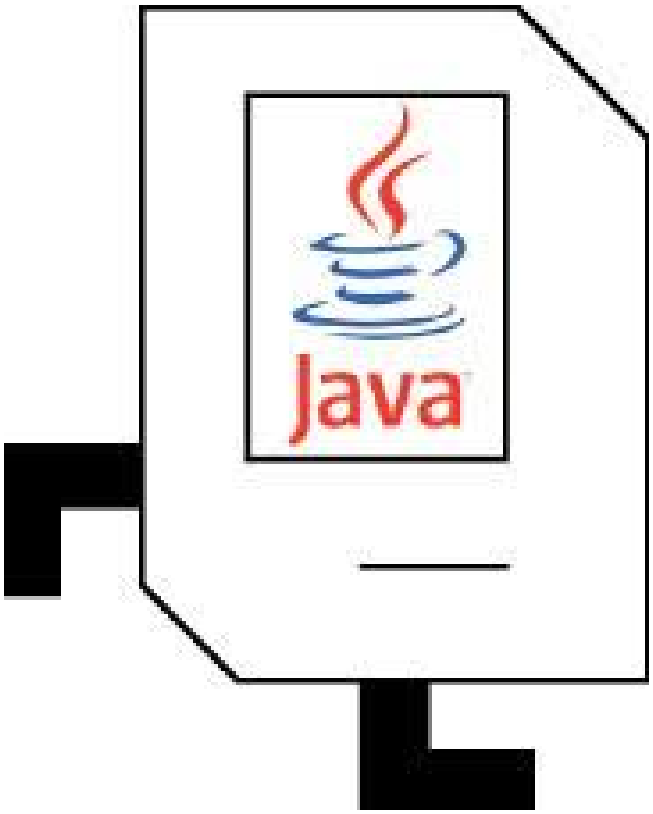


KAREL THE ROBOT

First project

Bahaeddin Ahmad Ibrahim Hammad

INTRODUCTION

There are many ways to divide the world in Karel the robot into four equal chambers, I tried many ways and I noticed that it's better to divide the work according to the length of the world.

The divisions are as follows:

- Odd horizontally and odd vertically
- Odd horizontally and even vertically
- Even horizontally and odd vertically
- Even horizontally and even vertically

Except for special cases

EXPLAINING THE VARIABLES

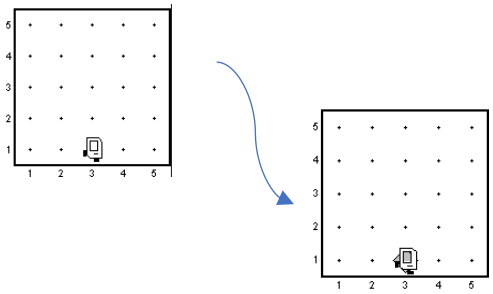
```
int numberOfSteps = 0,horizontalSteps = 0,verticalSteps = 0;
5 usages
boolean flag = true;
7 usages
```

- numberOfSteps: To count all the steps that karel takes
- horizontalSteps: To count the horizontal length
- verticalSteps: To count the vertical length
- flag: I used it in the special case to see if the length was one or two. I will explain more about it later

GENERAL METHODS

when we have a repeated code in a project, we compose this code into methods so that it can be easy to be used in different locations in the project and it will be more readable and understandable, so that's what I did in some codes in my project and these codes can be shown as:

```
7 usages
private void checkAndPutBeeper() {
    if(noBeepersPresent())
        putBeeper();
}
```

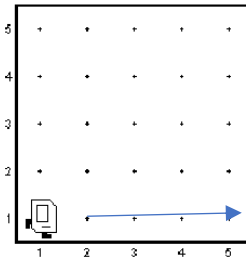


If there is no beeper, Karel will place a beeper in his parking spot

```
27 usages
private void moving() {
    move();
    numberOfSteps++;
}
```

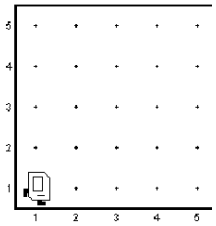
numberOfSteps will be increased by one with every move

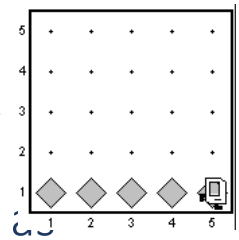
```
3 usages
private void movingAtAllPointsOfTheLine() {
    while(frontIsClear())
        moving();
}
```



Karel will move to the end of the line

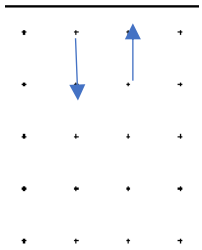
```
6 usages
private void beepersAtAllPointsOfTheLine() {
        while(frontIsClear()) {
            checkAndPutBeeper();
            moving();
        }
        checkAndPutBeeper();
}
```





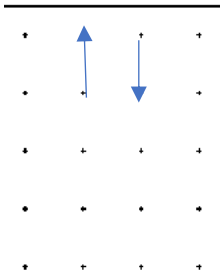
Karel will still put the beeper and move to the front, as long as there is no obstacle in front of it

```
3 usages
private void turnAroundInTheLeftLine() {
    turnLeft();
    moving();
    turnLeft();
}
2 usages
```



Karel will go to the left side and turn around

```
2 usages
private void turnAroundInTheRightLine() {
    turnRight();
    moving();
    turnRight();
}
1 usage
```



Karel will go to the right side and return

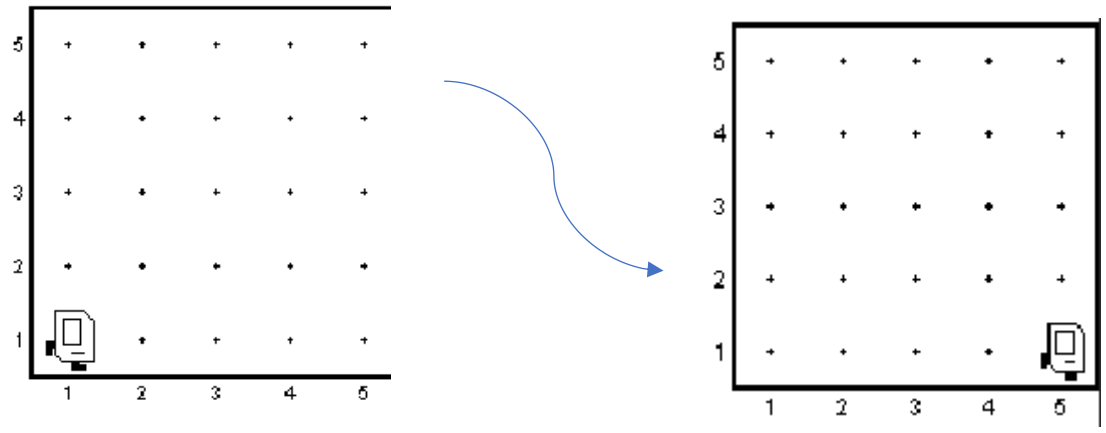
WAYS OF DIVISION

I divided my work into four cases according to horizontal and vertical dimensions, as mentioned in the introduction

Now I will show the first four steps that are shared in all the cases :

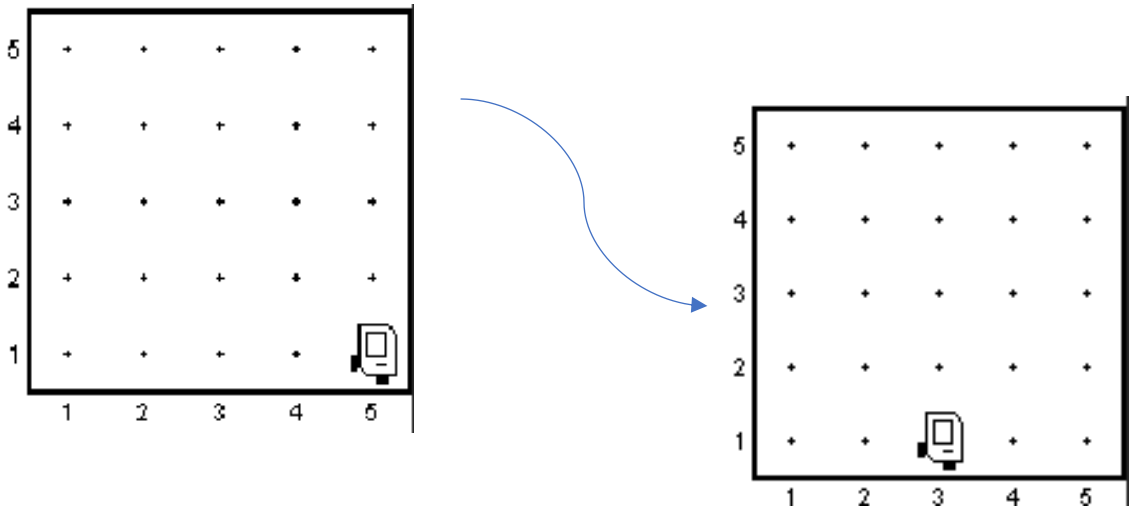
```
1 usage
public void lengthOfHorizontalLine() {
    while(frontIsClear()) {
        moving();
        horizontalSteps++;
    }
    backToMiddleHorizontalLine();
}
```

In this method, karel will move from the beginning of the line to its end to measure the horizontal length



```
1 usage
public void backToMiddleHorizontalLine() {
    turnAround();
    for(int i = 0;i< horizontalSteps/2;++i)
        moving();
    putBeeperVertically();
}
```

the Karel will return to the middle point, which Karel wants to cut through vertically, so I will call the putBeeperVertically() method

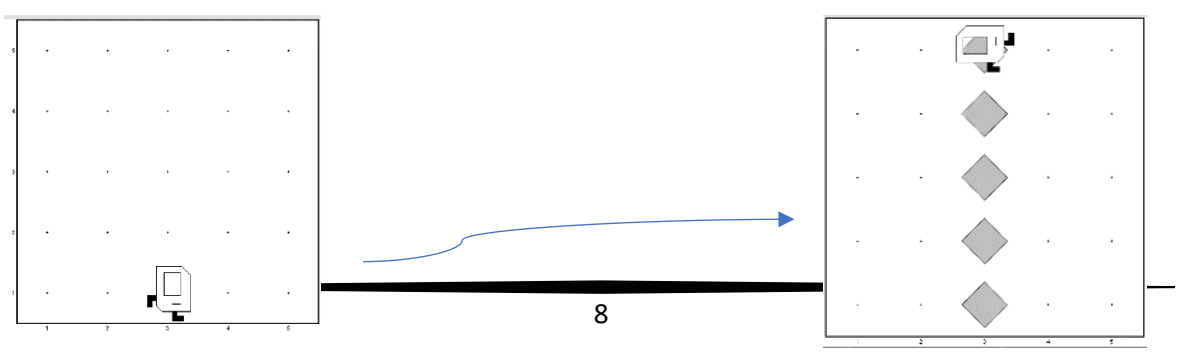


```
1 usage
private void putBeeperVertically() {
    turnRight();
    while (frontIsClear()) {
        if (horizontalSteps > 1)
            checkAndPutBeeper();
        moving();
        verticalSteps++;
    }
    if (horizontalSteps > 1 && verticalSteps > 1)
        checkAndPutBeeper();
    whereShouldIGo();
}
```

After Karel reached the middle point it will put a beeper on the middle vertical line,

Note* When Karel moved from the bottom to the top the vertical length is measured

Note* the if statement (horizontalSteps>1) to rule out less than one length which will be explained in the special case



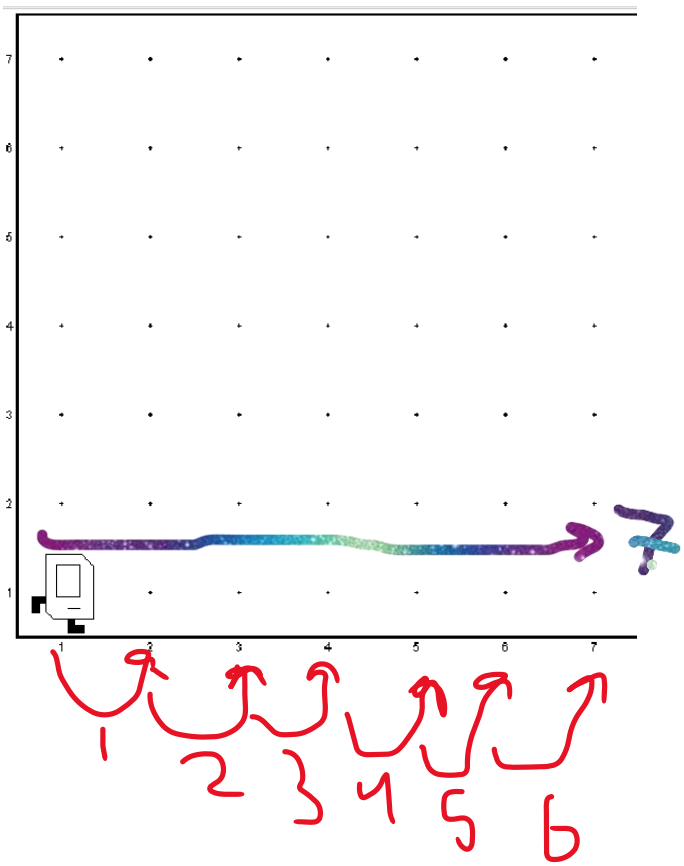
Now after the first shared three steps had been completed, the method whereShouldIGo()

Will lead the path specifically through one of the four ways based on horizontal and vertical dimensions

```
1 usage
private void whereShouldIGo() {
    if(verticalSteps <= 1 || horizontalSteps <= 1)
        specialCase();
    else if(horizontalSteps % 2 == 0)
        goToMiddleVerticalLine();
    else {
        putBeeperToUpperHalfVerticalLine();
        putBeeperToLowerHalfVerticalLine();
    }
}
```

- Odd horizontally and odd vertically

Note*In this case, the horizontalSteps and verticalSteps are even in number although the length is odd

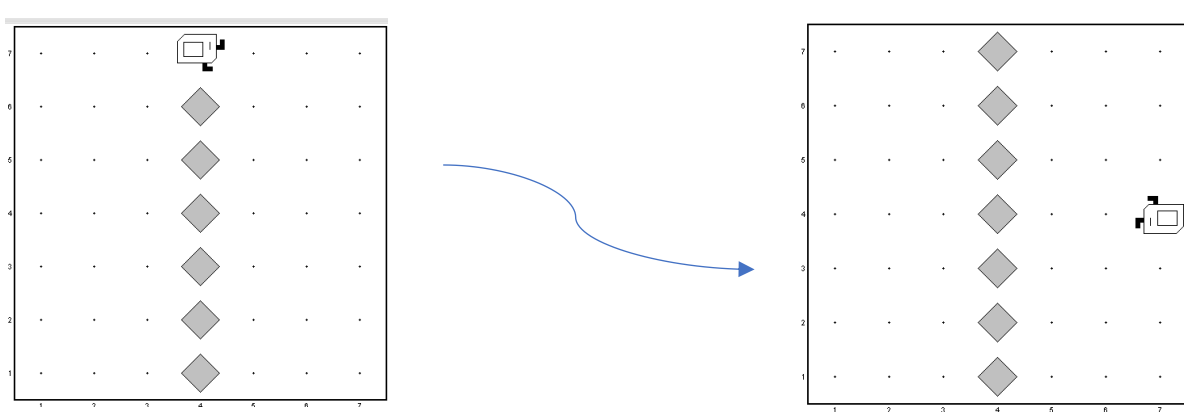


```
1 usage
private void whereShouldIGo() {
    if(verticalSteps <= 1 || horizontalSteps <= 1)
        specialCase();
    else if(horizontalSteps % 2 == 0)
        goToMiddleVerticalLine();
    else {
        putBeeperToUpperHalfVerticalLine();
        putBeeperToLowerHalfVerticalLine();
    }
}
```

Based on what was explained earlier, mod horizontal steps by two equal to zero, so we will use (goToMiddleVerticalLine(), to put a beeper on it horizontally

```java
private void goToMiddleVerticalLine() {
    if(verticalSteps % 2 == 0)
        goToMiddleVerticalOddLine();
    else
        goToMiddleVerticalEvenLine();
    putBeeperHorizontally();
}
1 usage
private void goToMiddleVerticalOddLine() {
    turnRight();
    movingAtAllPointsOfTheLine();
    turnRight();
    for(int i = 0;i < verticalSteps/2;++i)
        moving();
}
```

As shown in the figure, goToTheMiddleVerticalLine() has two options so we go through (goToTheMiddleOddVerticalLine()), Karel reached the middle vertical line

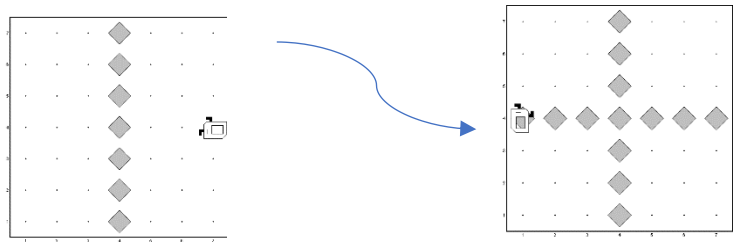


after reaching the middle vertical line, so now horizontal beeper can be added by putBeeperHorizontally() method

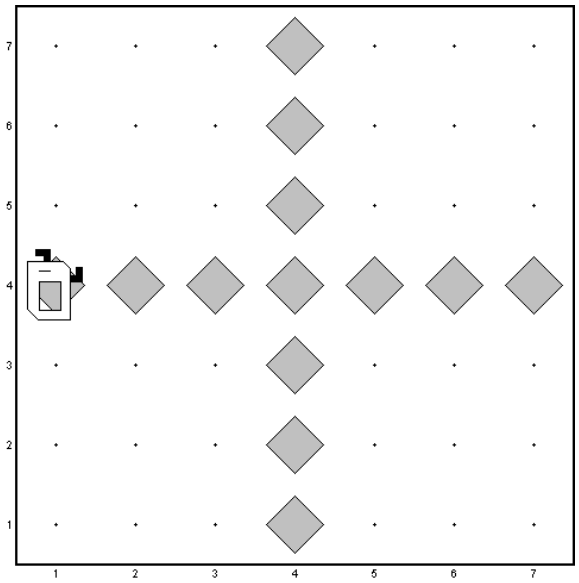
```
}
2 usages
private void putBeeperHorizontally() {
    if(horizontalSteps % 2 == 0) {
        if(verticalSteps % 2 == 0)
            putBeeperHorizontallyInOddLines();
        else
            putBeeperHorizontallyInOddHorizontalEvenVerticalLine();
    }
    else {
        turnRight();
        beepersAtAllPointsOfTheLine();
        if(verticalSteps % 2 == 0)
            putBeeperHorizontallyInEvenHorizontalAndOddVerticalLines();
        else
            putBeeperHorizontallyInEvenHorizontalAndEvenVerticalLines();
    }
}
```

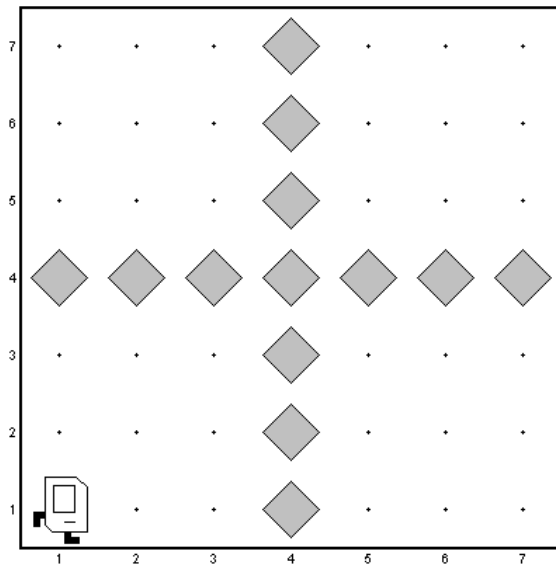
```
1 usage
private void putBeeperHorizontallyInOddLines(){
    turnRight();
    beepersAtAllPointsOfTheLine();
    KarelFinishedCutting();
}
```

putBeeperHorizontally() method leads us to more than one option so we go through the option that's highlighted in the figure above and illustrated in the second figure



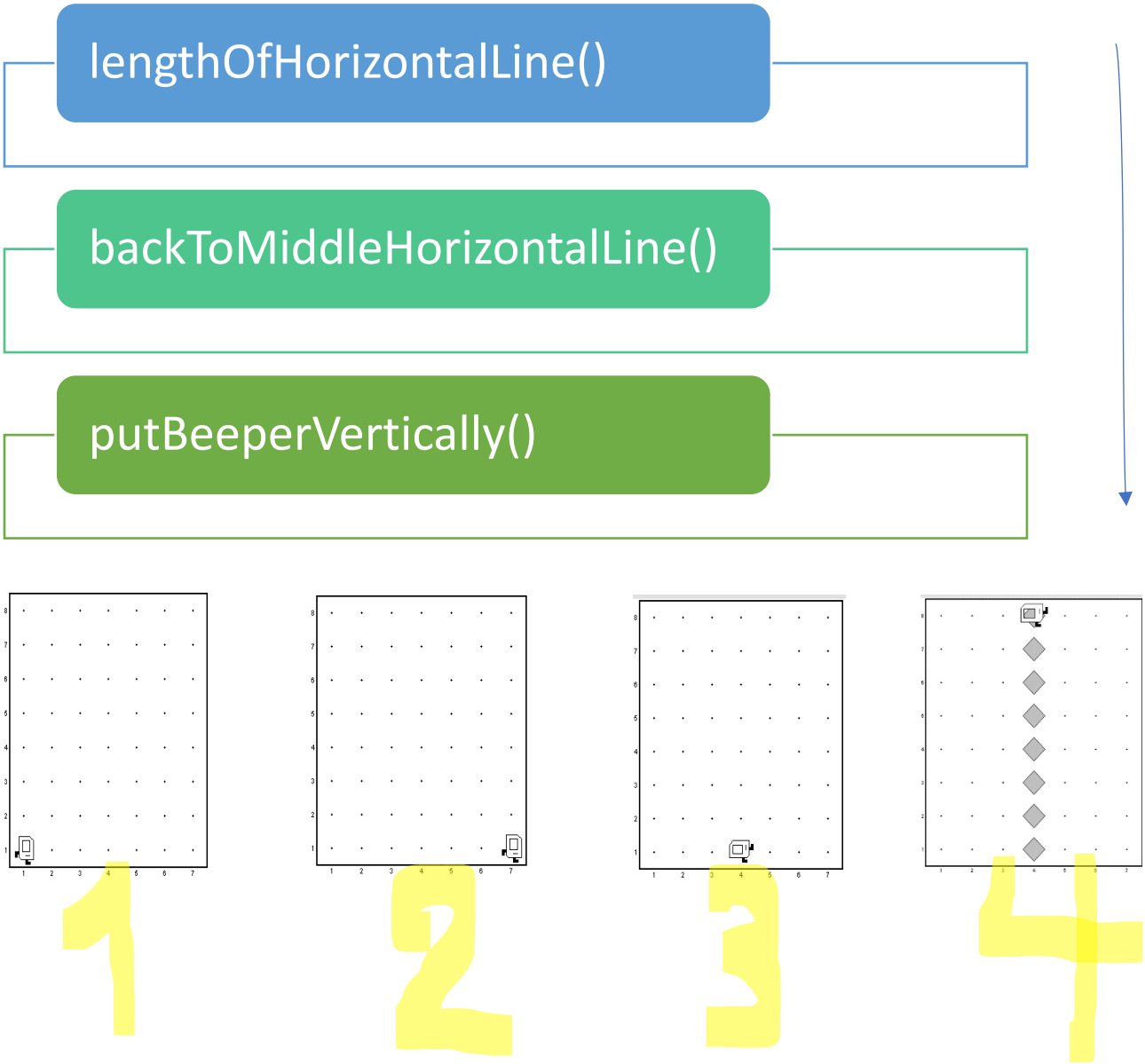
In the end, Karel will reach the beginning point again





- Odd horizontally and even vertically

After the four shared steps, which are summarized in the following diagram:



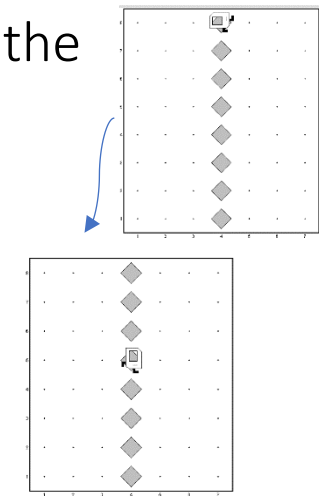
(whereShouldIGo()) method leads us to the next step (goToMiddleVerticalLine()) method

```
1 usage
private void whereShouldIGo() {
    if(verticalSteps <= 1 || horizontalSteps <= 1)
        specialCase();
    else if(horizontalSteps % 2 == 0)
        goToMiddleVerticalLine();
    else {
        putBeeperToUpperHalfVerticalLine();
        putBeeperToLowerHalfVerticalLine();
    }
}
```

```
private void goToMiddleVerticalLine() {
    if(verticalSteps % 2 == 0)
        goToMiddleVerticalOddLine();
    else
        goToMiddleVerticalEvenLine();
    putBeeperHorizontally();
}
```

goToMiddleVerticalLine() method leads us to goToMiddleVerticalEvenLine()

Note* this step differentiates this way from the previous one (even length vs odd length)



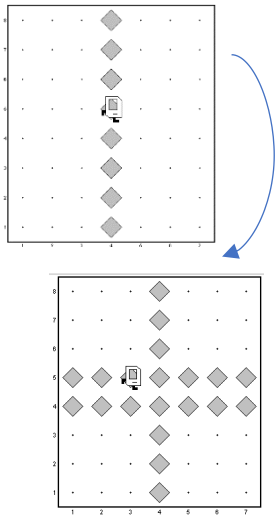
```
1 usage
private void goToMiddleVerticalEvenLine() {
    turnAround();
    for(int i = 0;i < verticalSteps/2;++i)
        moving();
```

After goToMiddleVerticalEvenLine() finished it's work, GoToMiddleVerticalLine() will call putBeeperHorizontally()

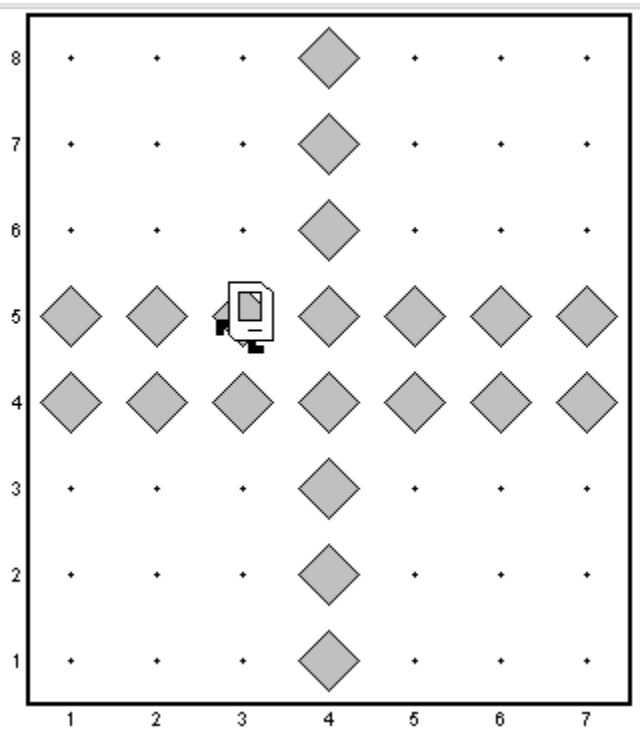
```
111     private void putBeeperHorizontally() {
112         if(horizontalSteps % 2 == 0) {
113             if(verticalSteps % 2 == 0)
114                 putBeeperHorizontallyInOddLines();
115             else
116                 putBeeperHorizontallyInOddHorizontalEvenVerticalLine();
117         }
118         else {
119             turnRight();
120             beepersAtAllPointsOfTheLine();
121             if(verticalSteps % 2 == 0)
122                 putBeeperHorizontallyInEvenHorizontalAndOddVerticalLines();
123             else
124                 putBeeperHorizontallyInEvenHorizontalAndEvenVerticalLines();
125         }
126     }
```

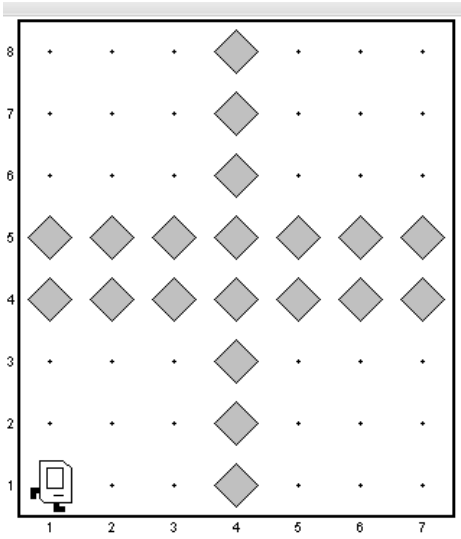
Because the horizontalSteps are even in number (that means the horizontal length is odd) and verticalSteps are odd in number (that means the vertical length is even), it will call the highlighted method

```
1 usage
private void putBeeperHorizontallyInOddHorizontalEvenVerticalLine() {
    turnLeft();
    beepersAtAllPointsOfTheLine();
    turnAroundInTheRightLine();
    beepersAtAllPointsOfTheLine();
    turnAroundInTheRightLine();
    for(int i =0 ;i < horizontalSteps/2-1;++i) {
        checkAndPutBeeper();
        moving();
    }
    checkAndPutBeeper();
    KarelFinishedCutting();
}
```



In the end, Karel will reach the beginning point again





In this case, notice

That 2 horizontal

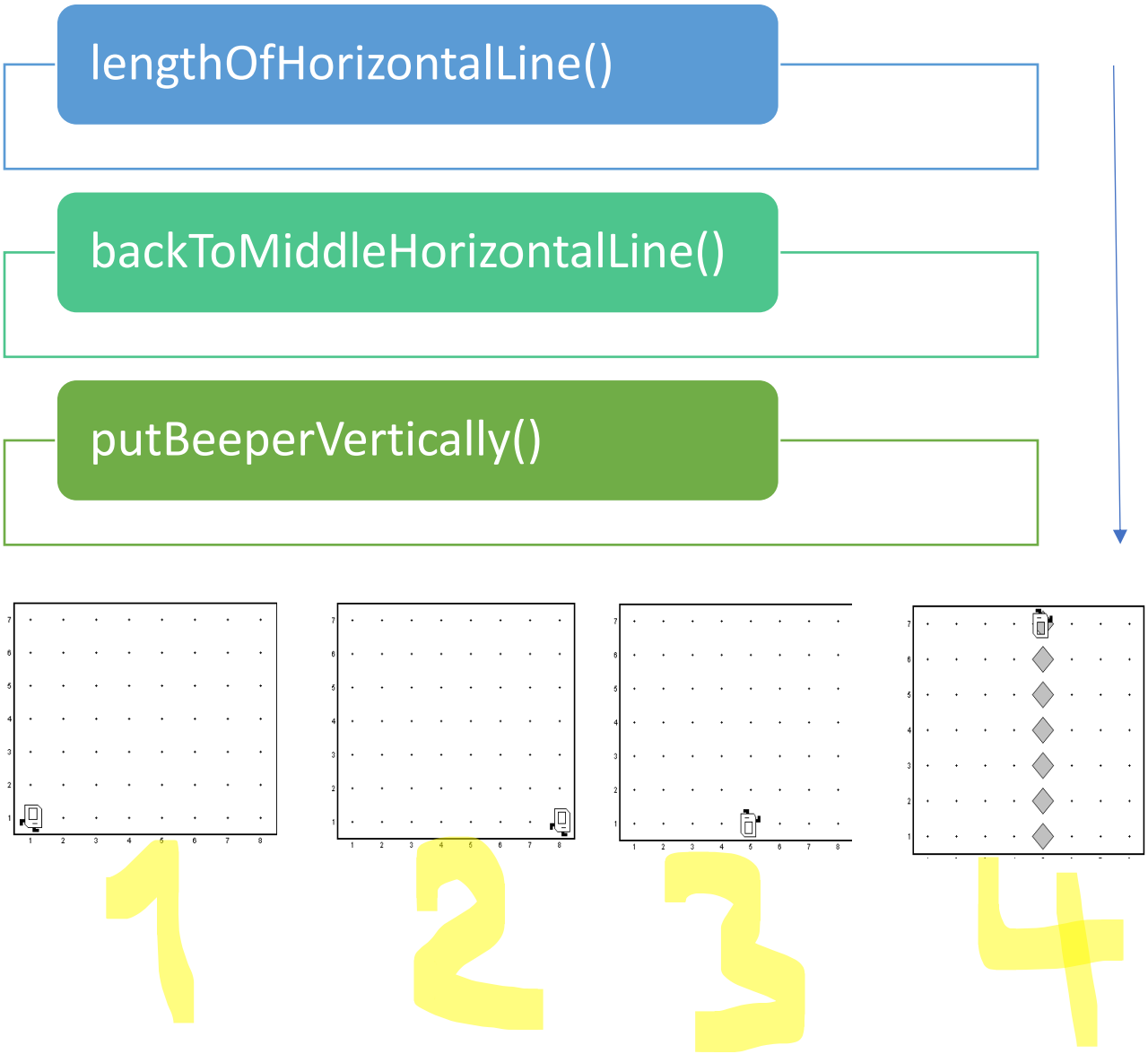
Lines are created

As a rectangle in the

World Because we have an even length

- Even horizontally and odd vertically

After the four shared steps, which are summarized in the following diagram:



(whereShouldIGo()) the method leads us to the last else in this method because HorizontalSteps%2!=0 and all the steps are more than one

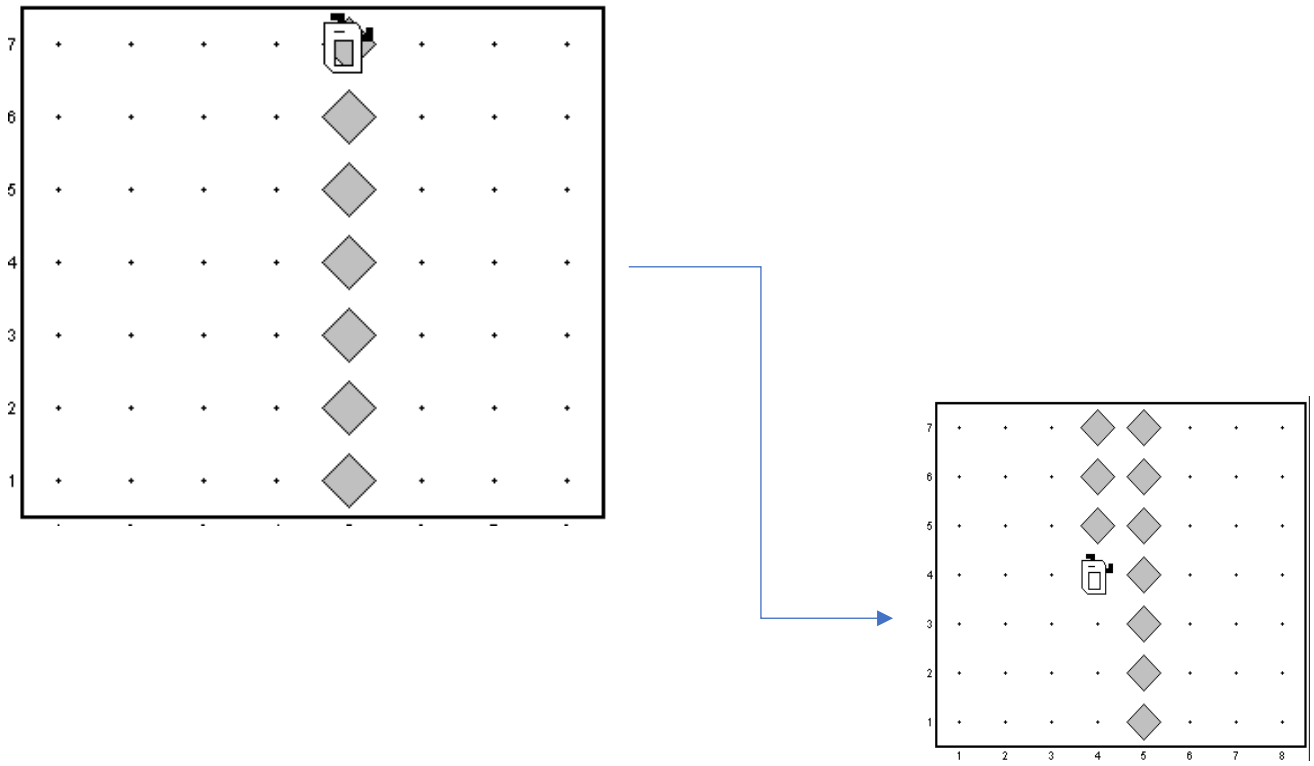
```
1 usage
private void whereShouldIGo() {
    if(verticalSteps <= 1 || horizontalSteps <= 1)
        specialCase();
    else if(horizontalSteps % 2 == 0)
        goToMiddleVerticalLine();
    else {
        putBeeperToUpperHalfVerticalLine();
        putBeeperToLowerHalfVerticalLine();
    }
}
```

Because the horizontal length is even, we are going to create two vertical lines

```
1 usage
private void putBeeperToUpperHalfVerticalLine() {
    turnAroundInTheLeftLine();
    for(int i = 0;i < verticalSteps/2;++i) {
        putBeeper();
        moving();
    }
    putBeeperHorizontally();
}
```

We use putBeeperToUpperHalf VerticalLine()

So, half a vertical line is created and now we are at the midpoint

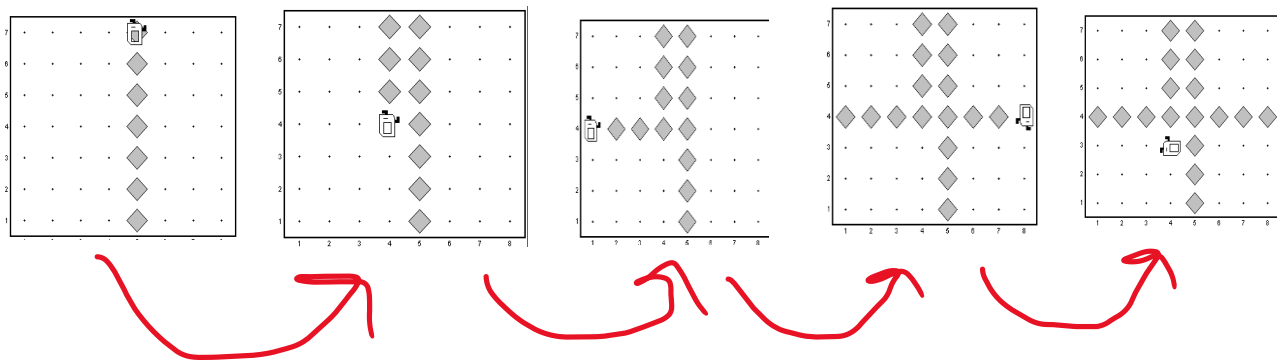


Now, this method will call putBeeperHorizontally()

Because we are at the middle horizontal axis

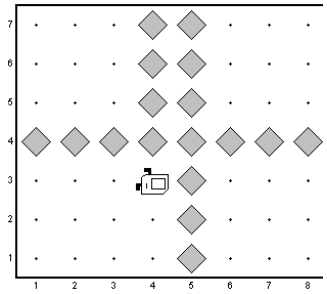
```
2 usages
private void putBeeperHorizontally() {
    if(horizontalSteps % 2 == 0) {
        if(verticalSteps % 2 == 0)
            putBeeperHorizontallyInOddLines();
        else
            putBeeperHorizontallyInOddHorizontalEvenVerticalLine();
    }
    else {
        turnRight();
        beepersAtAllPointsOfTheLine();
        if(verticalSteps % 2 == 0)
            putBeeperHorizontallyInEvenHorizontalAndOddVerticalLines();
        else
            putBeeperHorizontallyInEvenHorizontalAndEvenVerticalLines();
    }
}
```

```
private  void putBeeperHorizontallyInEvenHorizontalAndOddVerticalLines() {
    turnAround();
    beepersAtAllPointsOfTheLine();
    turnAround();
    for(int i = 0;i < horizontalSteps/2;++i)
        moving();
}
```

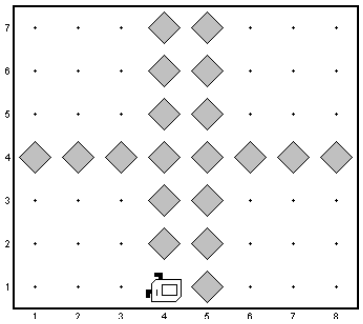


Now, we go back to whereShouldIgo() method and continue the work of this method throw putBeeperToLowerHalfVerticalLine()

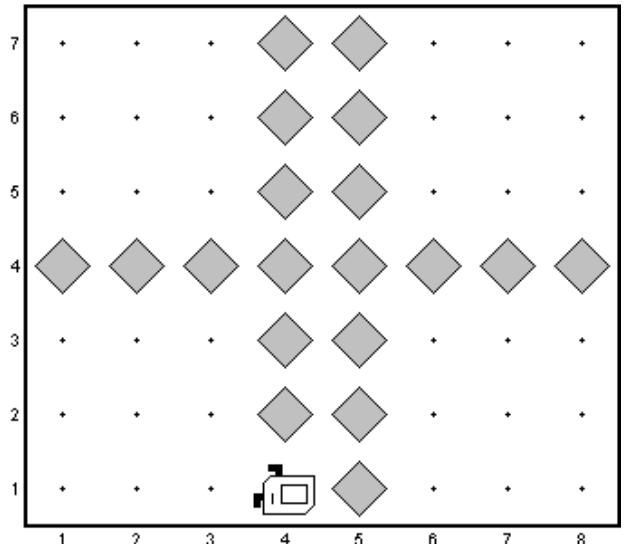
```
1 usage
private void putBeeperToLowerHalfVerticalLine() {
    moving();
    turnLeft();
    while(frontIsClear()) {
        moving();
        checkAndPutBeeper();
    }
    KarelFinishedCutting();
}
```



After the horizontal line was created as shown in the figure, I call the putBeeperToLowerHalfVerticalLine() to complete the second vertical line



In the end, Karel will reach the beginning point again



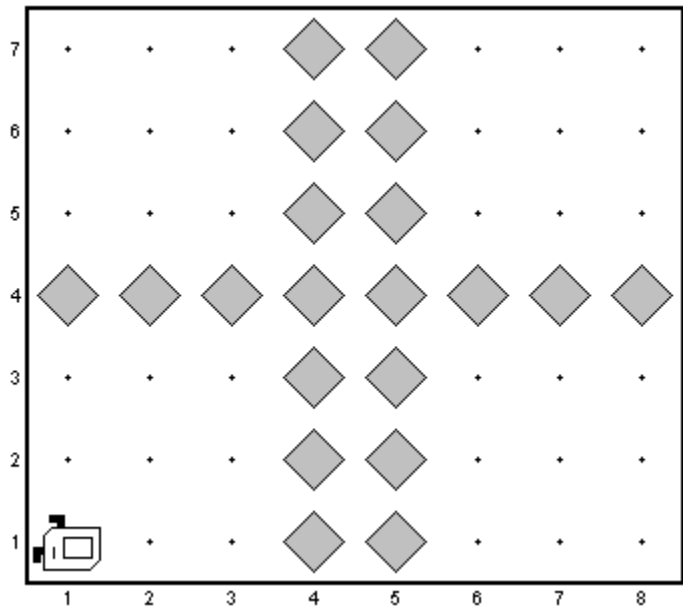
Note that two

Vertical lines are

Created because

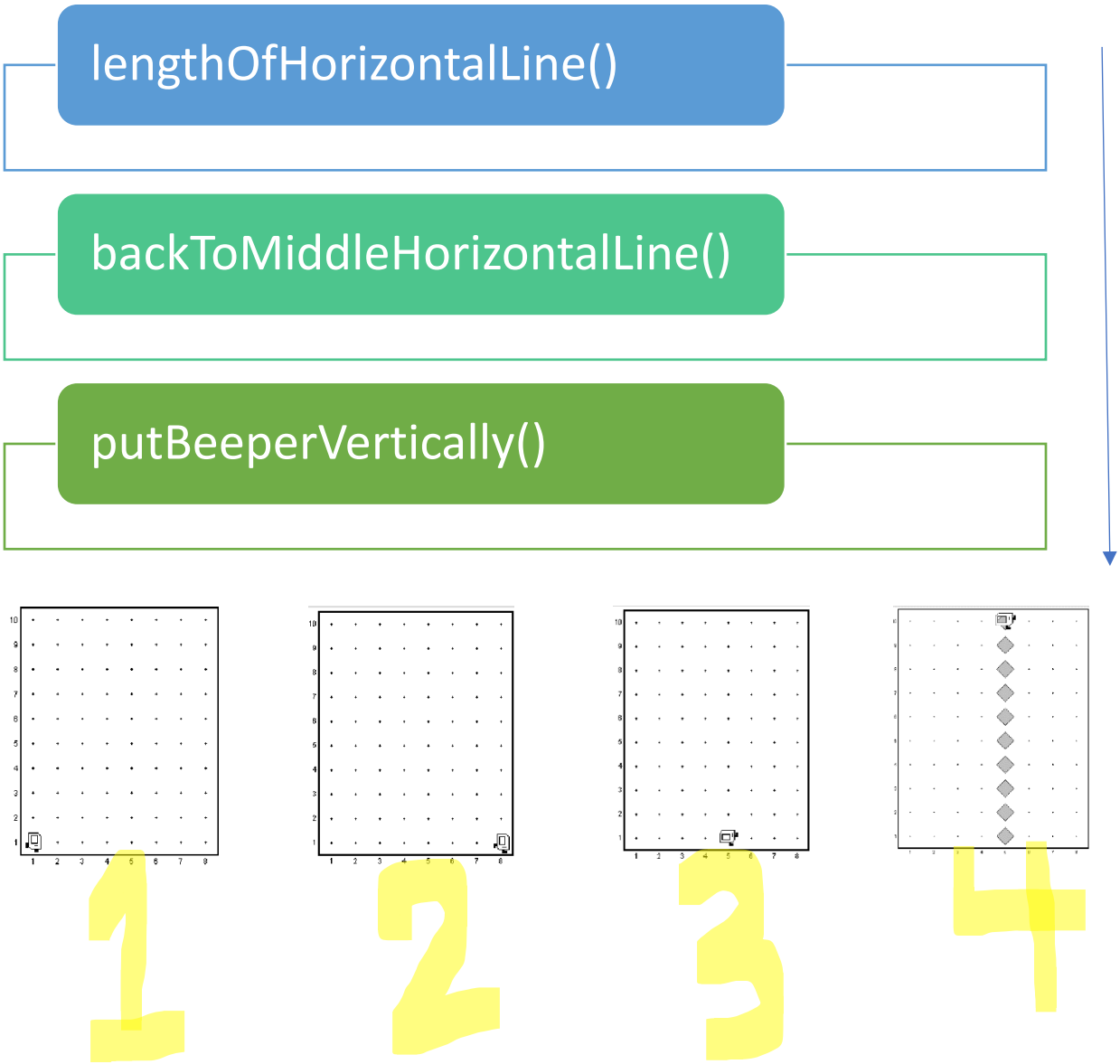
The length horizontal

Length is even



- Even horizontally and even vertically

After the four shared steps, which are summarized in the following diagram:



(whereShouldIGo()) the method leads us to the last else in this method because HorizontalSteps%2!=0 and all the steps are more than one

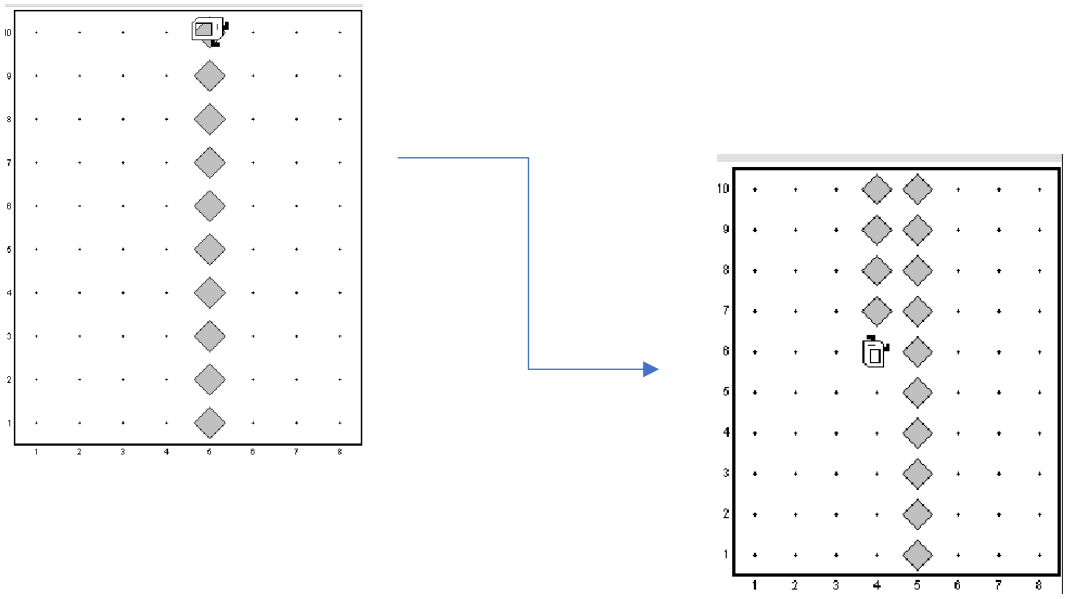
```
1 usage
private void whereShouldIGo() {
    if(verticalSteps <= 1 || horizontalSteps <= 1)
        specialCase();
    else if(horizontalSteps % 2 == 0)
        goToMiddleVerticalLine();
    else {
        putBeeperToUpperHalfVerticalLine();
        putBeeperToLowerHalfVerticalLine();
    }
}
```

Because the horizontal length is even, we are going to create two vertical lines

```
1 usage
private void putBeeperToUpperHalfVerticalLine() {
    turnAroundInTheLeftLine();
    for(int i = 0;i < verticalSteps/2;++i) {
        putBeeper();
        moving();
    }
    putBeeperHorizontally();
}
```

We use putBeeperToUpperHalf VerticalLine()

So, half a vertical line is created and now we are at the midpoint

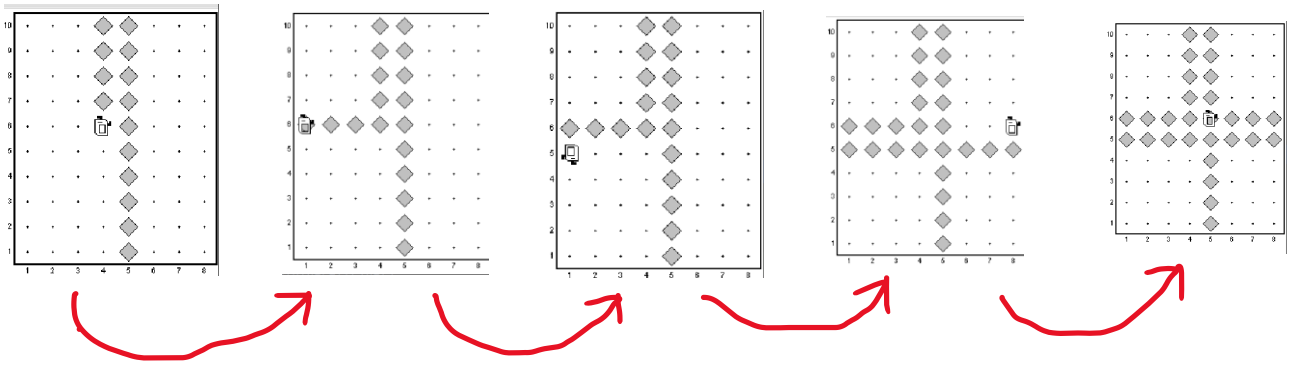


Now, this method will call putBeeperHorizontally()

Because we are at the middle horizontal axis

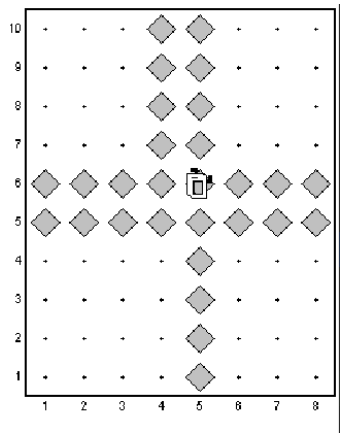
```
2 usages
private void putBeeperHorizontally() {
    if(horizontalSteps % 2 == 0) {
        if(verticalSteps % 2 == 0)
            putBeeperHorizontallyInOddLines();
        else
            putBeeperHorizontallyInOddHorizontalEvenVerticalLine();
    }
    else {
        turnRight();
        beepersAtAllPointsOfTheLine();
        if(verticalSteps % 2 == 0)
            putBeeperHorizontallyInEvenHorizontalAndOddVerticalLines();
        else
            putBeeperHorizontallyInEvenHorizontalAndEvenVerticalLines();
    }
}
```

```
1 usage
private  void putBeeperHorizontallyInEvenHorizontalAndEvenVerticalLines() {
    turnAroundInTheLeftLine();
    beepersAtAllPointsOfTheLine();
    turnAroundInTheLeftLine();
    while(noBeepersPresent()) {
        putBeeper();
        moving();
    }
}
```



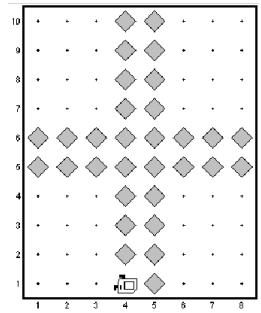
Now, we go back to whereShouldIgo() method and continue the work of this method throw putBeeperToLowerHalfVerticalLine()

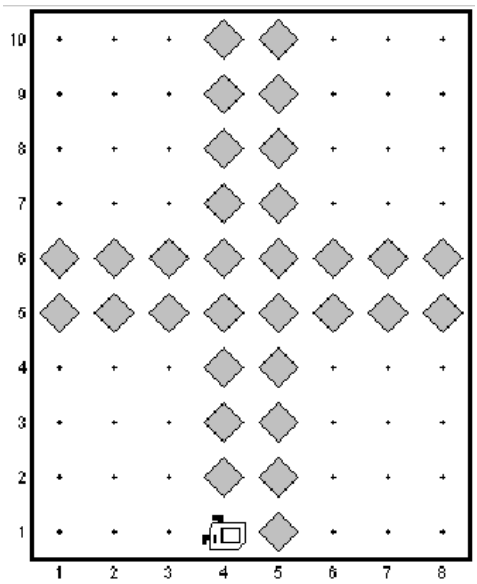
```
1 usage
private void putBeeperToLowerHalfVerticalLine() {
    moving();
    turnLeft();
    while(frontIsClear()) {
        moving();
        checkAndPutBeeper();
    }
    KarelFinishedCutting();
}
```

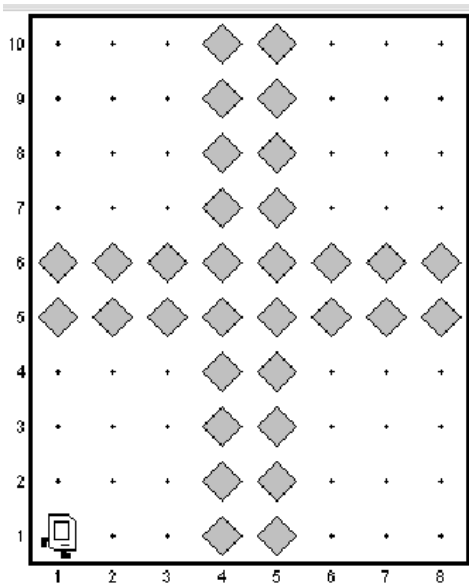


After the horizontal line was created as shown in the figure, I call the putBeeperToLowerHalfVerticalLine() To complete the second vertical line.

Two horizontal lines are created because the vertical line are even







Note that two

Vertical lines are

Created because

The length horizontal

Length is even

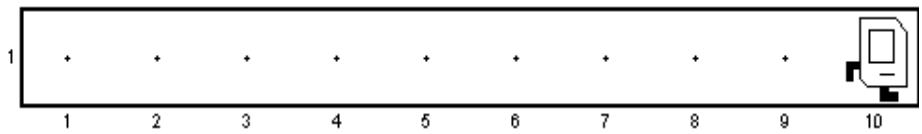
- Special case

The special case is applied if the horizontal or vertical length is 2 or 1

First case in special case:

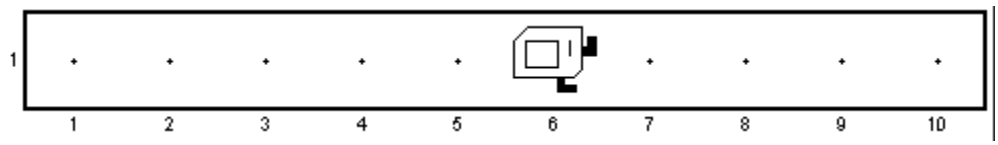
If vertical length equals one and horizontal length is more than two

```
1 usage
public void lengthOfHorizontalLine() {
    while(frontIsClear()) {
        moving();
        horizontalSteps++;
    }
    backToMiddleHorizontalLine();
}
```



Note that we measure the horizontal length but we don't know that the vertical length is one, so so Karel will back to mid horizontal point as usual in the same function we talked about it before

```
1 usage
public void backToMiddleHorizontalLine() {
    turnAround();
    for(int i = 0;i< horizontalSteps/2;++i)
        moving();
    putBeeperVertically();
}
```



Now backToMiddleHorizontalLine() will call putBeeperVertically()

```
1 usage
private void putBeeperVertically() {
    turnRight();
    while (frontIsClear()) {
        if (horizontalSteps > 1)
            checkAndPutBeeper();
        moving();
        verticalSteps++;
    }
    if (horizontalSteps > 1 && verticalSteps > 1)
        checkAndPutBeeper();
    whereShouldIGo();
}
```

But here because front is not clear, Karel will do anything here, in the last if Karel can’t do anything because verticalSteps less than one

Now putBeeperVertically() method Will call whereShouldIgo() method

```java
private void whereShouldIGo() {
    if(verticalSteps <= 1 || horizontalSteps <= 1)
        specialCase();
    else if(horizontalSteps % 2 == 0)
        goToMiddleVerticalLine();
    else {
        putBeeperToUpperHalfVerticalLine();
        putBeeperToLowerHalfVerticalLine();
    }
}
```

Because verticalSteps less than one, whereShouldIGo() method will lead us to specialCase() method

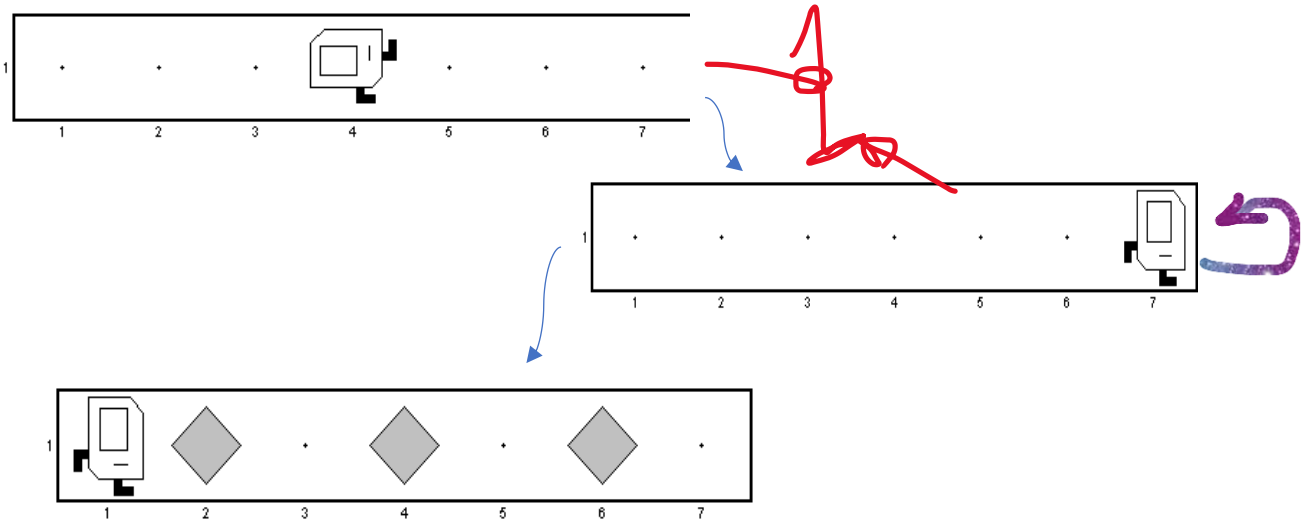
```java
1 usage
private void specialCase() {
    if(verticalSteps == 1) {
            turnAround();
            moving();
            if(beepersPresent())
                pickBeeper();
            turnAround();
    }
    if((verticalSteps <= 1 && horizontalSteps >= 7 ) || (verticalSteps >= 7 && horizontalSteps <= 1))
        putBeeperInSpecialCaseIfLengthGreaterThanSeven();
    else if((horizontalSteps < 7 && horizontalSteps > 1 && verticalSteps <= 1 ) ||
            (horizontalSteps <= 1 && verticalSteps <7 && verticalSteps > 1))
        putBeeperInSpecialCaseIfLengthGreaterThanTwoAndLessThanEight();
    KarelFinishedCutting();
}
```

whether the horizontal length more than two and less than eight or more than eight

we will start by more than 2 and less than 8

now specialCase() method will call

```
1 usage
private void putBeeperInSpecialCaseIfLengthGreaterThanTwoAndLessThanEight() {
    if (verticalSteps <= 1) {
        turnRight();
        movingAtAllPointsOfTheLine();
    }
    turnAround();
    while(frontIsClear()) {
        moving();
        putBeeper();
        if(verticalSteps == 1 || horizontalSteps == 1)
            putBeeperOnTheOppositeSide();
        if(frontIsClear())
            moving();
    }
}
```



As illustrated in the figures above, Karel will move from the mid horizontal point to the end point and start putting beepers following (leave one put one principle)

Until reaching the starting point

An important thing in this case is:

some worlds can’t be divided into 4

identical champers so it would be divided to the most possible number of identical champers

second case in special case:

If horizontal length equals one and vertical length is more than two

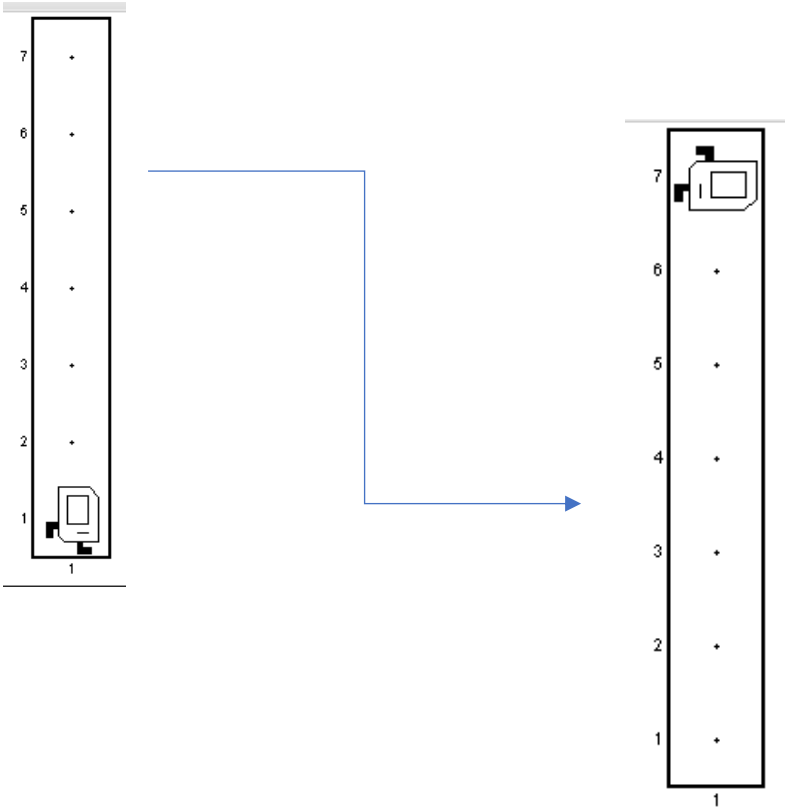
```
1 usage
public void lengthOfHorizontalLine() {
    while(frontIsClear()) {
        moving();
        horizontalSteps++;
    }
    backToMiddleHorizontalLine();
}
```

This method in this case will not do any thing because horizontal length is 1

We will go to backToTheMiddleHorizontalLine()

Will not do any thing because Karel already at the middle point, now in putBeeperVertically() method Karel will go to the top of the world and calculate the vertical length and will not put beeper in it's road because there is an if statement for this case

```
1 usage
private void putBeeperVertically() {
    turnRight();
    while (frontIsClear()) {
        if (horizontalSteps > 1)
            checkAndPutBeeper();
        moving();
        verticalSteps++;
    }
    if (horizontalSteps > 1 && verticalSteps > 1)
        checkAndPutBeeper();
    whereShouldIGo();
}
```

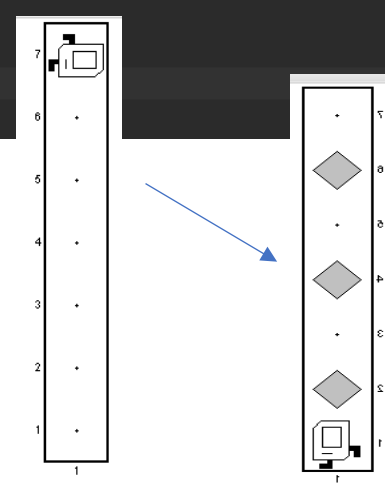


Now whereShouldIGo() method leads us to SpecialCase() again which has two choices

Now I'm going to show a case of horizontal length equals one and vertical length more than two and less than eight

now specialCase() will lead us to this method

```
1 usage
private void putBeeperInSpecialCaseIfLengthGreaterThanTwoAndLessThanEight() {
    if (verticalSteps <= 1) {
        turnRight();
        movingAtAllPointsOfTheLine();
    }
    turnAround();
    while(frontIsClear()) {
        moving();
        putBeeper();
        if(verticalSteps == 1 || horizontalSteps == 1)
            putBeeperOnTheOppositeSide();
        if(frontIsClear())
            moving();
    }
}
```



Third case in special case:

If vertical length equals one and horizontal length is eight or more

We mentioned earlier that special case has two options in this case we will go through the second option

```
private void putBeeperInSpecialCaseIfLengthGreaterThanSeven() {
    incrementVerticalAndHorizontalStepsByOne();
    int k = horizontalSteps * verticalSteps;
    int steps = k / 4;
    int neglectedArea = k % 4;
    if (verticalSteps <= 2)
        goToBeginningOfTheHorizontalLineInSpecialCase();
    turnAround();
    if (neglectedArea == 3)
        divideWorldUsingThreeBeepers(steps);
    else
        divideWorldUsingAppropriateBeepers(steps,neglectedArea);
}
```

In this case it’s possible to divide any world into 4 identical champers

And we depend mainly on the mod and division.

The mod enables us to know the number of unwanted points

The division enables us to know the number of points in each world

This is appeared in red brackets in the figure above

```
1 usage
private void incrementVerticalAndHorizontalStepsByOne() {
    if (verticalSteps != 1 && horizontalSteps != 1) {
        verticalSteps++;
        horizontalSteps++;
    } else {
        flag=false;
        if (verticalSteps != 1)
            verticalSteps++;
        else
            horizontalSteps++;
    }
}
```

Note that in this case we depend on the length mainly to make the calculations correctly .

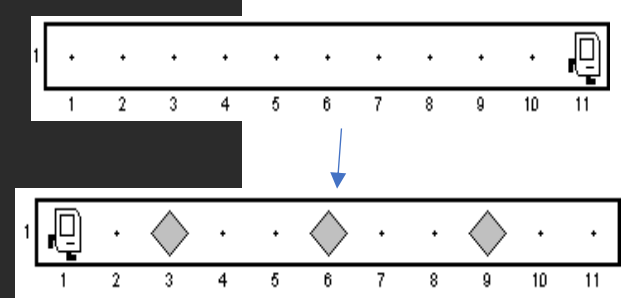
Let’s back to the main function in this case, here there are two ways for putting a beeper.

One way is when the mod equals three, the second is when the mod equals one or two.

**note: we can use one way in putting a beeper, but in case of mod equals 3 there will be excess beepers, so we separate it as a single case.

Now we are going to explain the case of mod equals 3 :

```
private void divideWorldUsingThreeBeepers(int steps) {
    for(int i = 0;i < steps;++i)
        moving();
    if (flag == false)
        putBeeperOnTheOppositeSide();
    putBeeper();
    for(int i = 0;i < 2;++i) {
        for(int y=0;y<=steps;++y)
            moving();
        if (flag == false)
            putBeeperOnTheOppositeSide();
        putBeeper();
    }
    for(int i=0;i<steps;++i)
        moving();
}
```



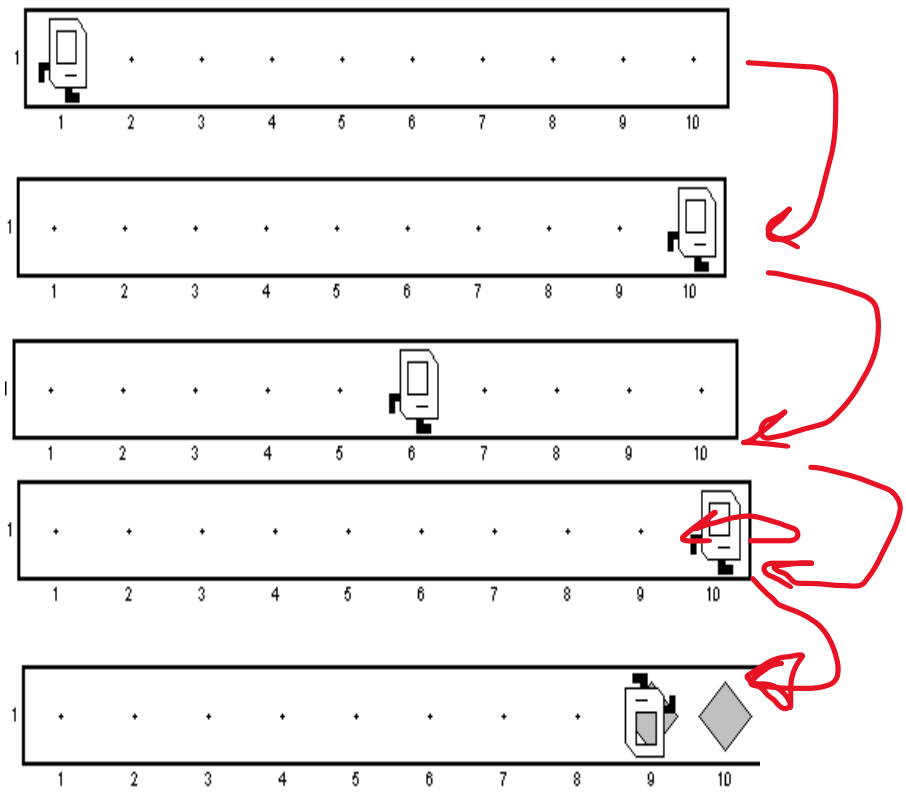
the world would be divided into four identical champers without excluding any points.

But if the mod equals one or two:

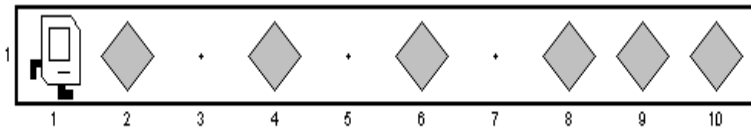
At the beginning, we will exclude all unwanted points by putting Beepers over them.

After that we will divide the world into four identical champers.

```java
private void divideWorldUsingAppropriateBeepers(int steps,int neglectedArea) {
    for (int i = 0; i < neglectedArea; ++i) {
        putBeeper();
        if (flag == false)
            putBeeperOnTheOppositeSide();
        moving();
    }
    while (frontIsClear()) {
        putBeeper();
        if (flag == false)
            putBeeperOnTheOppositeSide();
        for (int i = 0; i < steps; ++i) {
            if (frontIsClear())
                moving();
        }
    }
}
```



note that two bepeers are added because the mod is two.



Note that there is two steps between each beeper and the following one because the division is two.

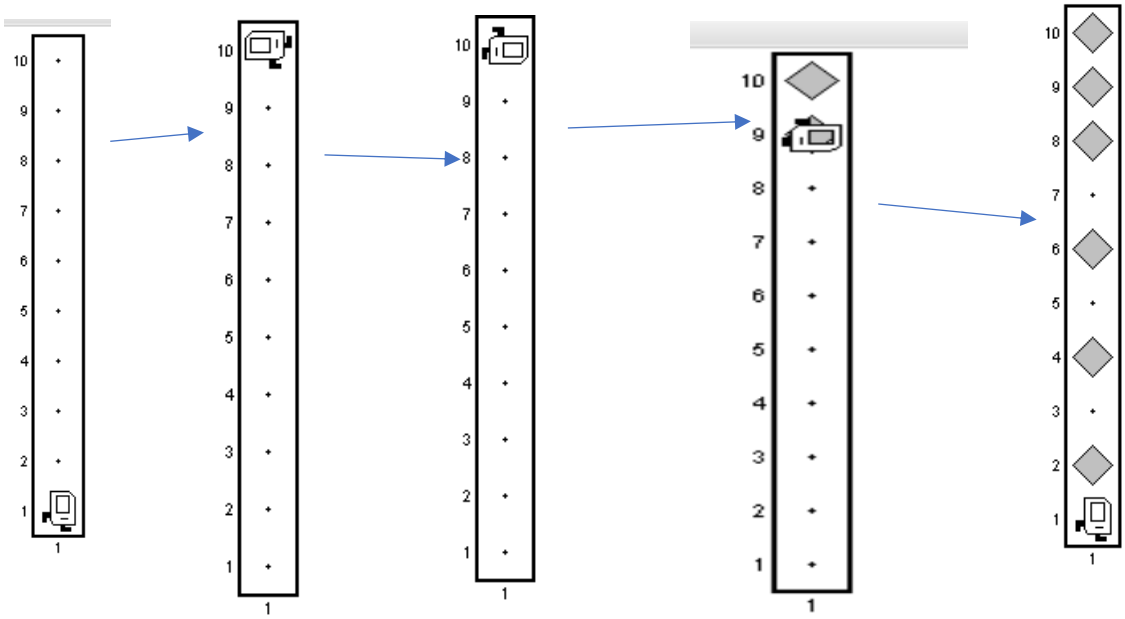
Fourth case in special case :

If horizontal length equals one and vertical length is eight or more

This case is very similar to the previous case, but karel will go vertically to the end point according to PutBeeperVertically() method but will not put a Beeper while it is going up , in this case it will measure the vertical length

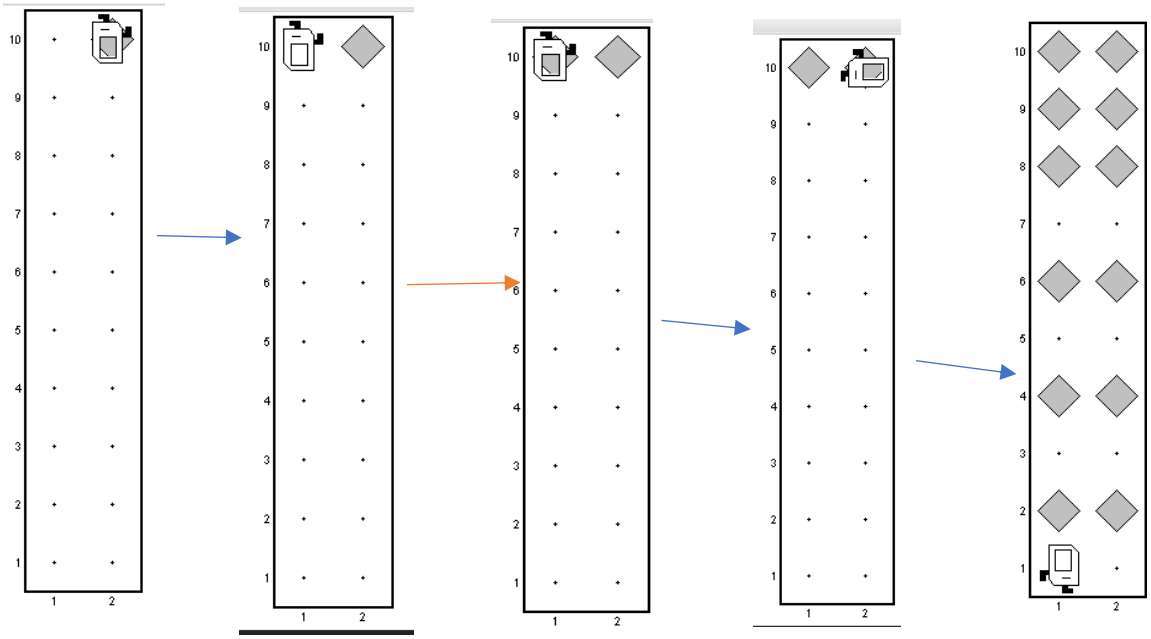
PutBeeperVertically() calls whereShouldIGo() that leads us to the special case which leads to PutBeeperInSpecailCaseIfLengthGreaterThanSeven()

Karel will turn around and do then do the same thing as the previous case



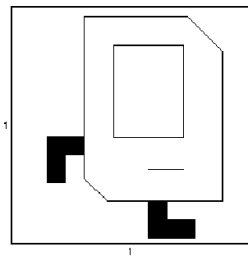
I will discuss a shared function for all special cases : it is when the length is two , we gonna use method to put beepers on the second line

```
5 usages
private void putBeeperOnTheOppositeSide() {
        turnRight();
        moving();
        putBeeper();
        turnAround();
        moving();
        turnRight();
}
4 usages
```

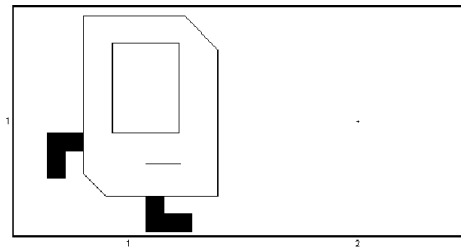


- Special cases

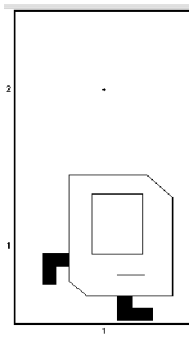
In any one of the following cases, there is no way to divide them anymore.



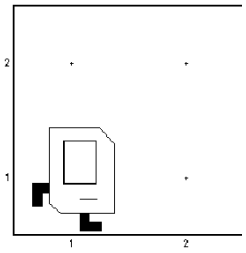
(1,1)



(2,1)



(1,2)



(2,2)