Exercise 1: Karel

September 7, 2021

Goals of this exercise

- setting up and learning IntelliJ
- writing algorithms
- abstraction using functions

Preperations

- If you do not have IntelliJ installed already, please navigate to `https://www.jetbrains.com/idea/download`. Please do not download the Ultimate or Community version, but scroll down and download the Educational version (Edu).
- Download `Karel.zip` from BrightSpace. Put it on a appropriate place on your computer and unpack it (creating the folder Karel).
- From the IntelliJ's welcome screen, select 'Open or Import' and point to the folder called Karel. Some predefined files will appear. For this exercise you only have to modify the files `Border.java`, `FollowLine.java` and `Church.java`. Do not change the other files.

Karel

Karel is a robot who can be controlled using simple commands. Karel lives in a world represented by a grid. He can walk and turn (90 degrees). Karel carries a bag of balls which he can drop or pick up. While programming Karel you should be careful not to let him run into walls. Please only use language constructs that are introduced in the lecture (so for example do not use variables).

Tutorial

Although you should not edit the file, you can find the results of the tutorial we did in the lecture in the `Tutorial.java` file. This can be a helpful resource for your own assignment.

Part 1.1: A Border around the Garden

Charles would like to put a nice hedge around his garden. Write a program that uses the methods `step`, `turnLeft`, `turnRight` en `putBall` to help him do so. To avoid Charles running into a wall, be sure to use the function `inFrontOfWall`.

When Charles is done, the end result should look like this:

```
o o o o o o o o o o o o o o o
o . . . . . . . . . . . . . o
o . . . . . . . . . . . . . o
o . . . . . . . . . . . . . o
o . . . . . . . . . . . . . o
o . . . . . . . . . . . . . o
o . . . . . . . . . . . . . o
o . . . . . . . . . . . . . o
o . . . . . . . . . . . . . o
o . . . . . . . . . . . . . o
o . . . . . . . . . . . . . o
o . . . . . . . . . . . . . o
o o o o o o o o o o o o o o o
```

For this exercise, modify `Border.java`.
Remember that by default, Karel starts in the bottom left corner facing north.

Tip: Walking in a straight line and placing down balls contains a lot of repetition and repeated snippets of code. Have a look at the lecture slides to see how we deal with such problems. Your code should not need to be longer than 30 lines!

Part 1.2: Follow the Yellow Brick Road

In this exercise Charles has to follow a bendy road. Write a program that lets Charles follow the entire path, indicated by the balls. Once Charles is at the end of the road, he can take a rest there. For this exercise, modify `FollowLine.java`. You will see that there is already some code to make Charles step on the start of the road. **Charles does not know how long each section of road is, but he can see in which order he has to turn left or right to get to the end.** As such, you cannot hardcode the amount of steps Charles has to take, but you are allowed to specify which direction he should turn to at each corner.

Tip: Charles is likely to walk off the path at the corners. This is not a problem, but make sure that he gets back on the path when he does! Consider making a function called `stepBack` that helps him do so.

[BONUS] Part 1.3: Around the church

Charles is visiting a famous Church. He want to make a round around the church, staying as close as possible to the walls without bumping into it (to really look at the decorations). Write a program that lets Charles make exactly one round around the church. You cannot hardcode the amount of steps Charles needs to make, **nor** which direction he needs to turn to continue his round. You may assume that there is enough space for Charles to make his round (i.e. there are no other walls than those of the church). Once Charles has made one round, he can stop where he began.

> **Tip:** It might be useful for Charles to mark where he started his round, to avoid going in circles. Of course, he should remove such markers when he is done.

Handing in your assignment

Important rules for handing in your assignment:

- Write both your names and student numbers in the code (comments)
- Only 1 student per group should upload the code
- You can only upload once, so upload when you are ready with all the partial exercises
- Make one project every week (so one project with the partial exercises 1, 2 and optional 3)
- Create a single zip file from your project: lookup your project folder in the explorer (or finder on a Mac) and compress the complete folder into a zip file
- Upload the zip file to BrightSpace