Lab2 Document

Han, Yizhan

March 29, 2023

Overview

This code is a simple implementation of Karel the Robot where Karel can move in a 2D plane and pick up rocks. The program performs input and output via the terminal and ends when receiving the input "Q" or user having collected all the rocks.

System Design

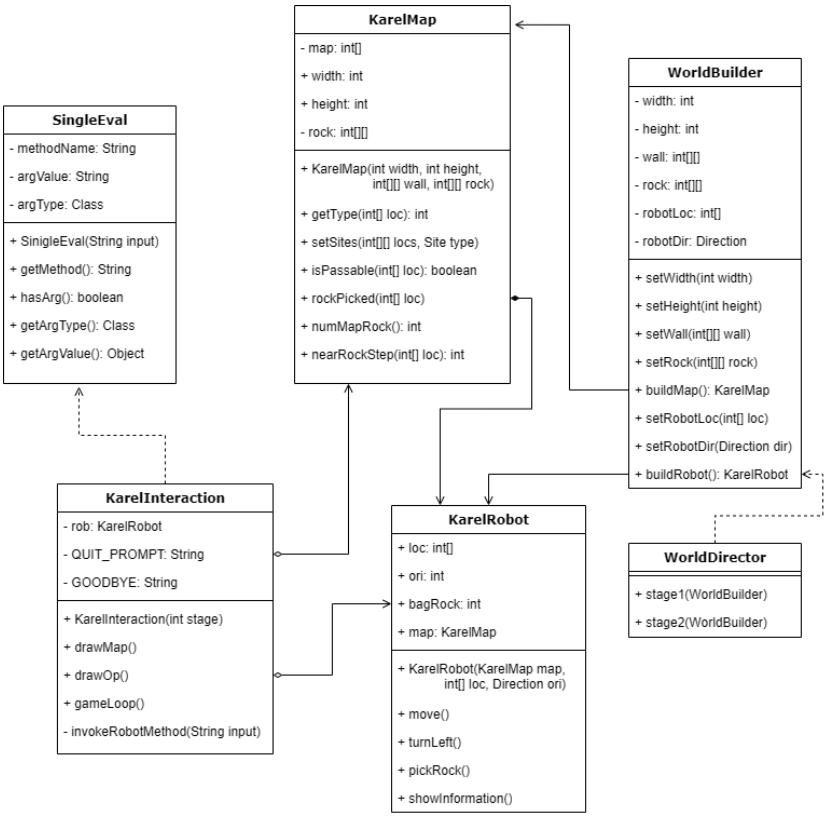

Figure 1: UML class diagram

The program consists of 6 classes: WorldBuilder, WorldDirector, KarelMap, KarelRobot, KarelInteraction and SingleEval. WorldBuilder and WorldDirector implement a Builder pattern to build the world differing on the user input. KarelMap uses an array that stores the information of the world. KarelRobot is the robot that can move

and pick up rocks. KarelInteraction is the main class that interacts with the user and SingleEval is an auxiliary class that evaluates the user input during the game and returns the corresponding infomation needed to invoke methods of the robot.

Classes and Methods

WorldBuilder and WorldDirector

WorldBuilder and *WorldDirector* implement a Builder pattern allowing the program to build different KarelMap and KarelRobot step by step with similar codes. For this stage of program development everything is finished with existent code in the Director class.

KarelMap

KarelMap represents a 2D plane where Karel can move. It uses a 1D array to store the information of each site in the plane, which can be either GROUND, KAREL, WALL or ROCK. The main methods include:

- *KarelMap*(*int width*, *int height*, *int*[][] *wall*, *int*[][] *rock*): Constructor. Initialize the map with the given width and height and then set the given sites to corresponding types. Often invoked by the builder.
- *int getType*(*int*[] *loc*): return the value of *loc*.
- *void setSites*(*int*[][] *locs*, *Site type*): set the type of *locs* to *type*.
- *boolen isPassable*(*int*[] *loc*): check if the specified site is passable namely not out of the map or not the site that cannot pass.
- *void rockPicked*(*int*[] *loc*): change the state when the rock at the given site is picked.
- *int numMapRock*(): return the number of rocks in the map.
- *int nearRockStep*(): return the steps to the nearest rock.

KarelRobot

KarelRobot represents Karel the robot and contains the information of the robot's location, orientation and the number of rocks collected. When the robot performs a certain action, methods of KarelMap will be invoked to update the state. The main methods include:

- *KarelRobot*(*KarelMap map*, *int*[] *loc*, *int ori*): Constructor. Initialize the robot with the given map, location and orientation. Often invoked by the builder.
- *void move*([*int n*]): Robot moves one step forward and an exception will be thrown if the next site is not passable. If given an integer as the argument, robot can move *n* steps forward.
- *void turnLeft*(): Robot turns to the left.
- *void pickRock*(): Robot picks the stone in front of it.
- *void showInformation*(): print the information of the Karel world, including the number of rocks on the map, the number of rocks in the bag, and the distance to the nearest rock

KarelInteraction

KarelInteraction is the entrance of program and responsible for the interaction with the user. It uses a scanner and while(true) loop to receive user's input and invoke methods of KarelRobot to implement the user's action. Some static outputs, such as welcome message and exit command, are also stored in this class. The main methods include:

- *KarelInteraction*(*int stage*): Constructor. Initialize the map and robot with the given *stage* value.
- *drawMap*(): Invoke *drawSite*() and *drawRow*() to output the map via the terminal.
- *drawOp*(): Print the welcome message.
- *gameLoop*(): Main loop of the game. Receive user input and invoke methods of KarelRobot to implement the user's action requirement.
- *invokeRobotMethod*(*String input*): Invoke methods of KarelRobot according to the user input. First parse the *input* to get the method name and the argument, then exploit the reflect mechanism. Throw exceptions if the methods are non-existent or throwing exceptions.

SingleEval

SingleEval is a helper class that evaluates the user input during the game and parse the input to invoke methods of the robot. The main methods include:

- *SingleEval*(*String input*): Constructor. Initialize a evaluated input object with the given *input*, dividing the *input* to the method name part and the argument part.
- *String getMethod*(): return the method name.
- *boolean hasArg*(): check if the input is with arguments.
- *Class getArgType*([*String type*]): get the type of the argument, judging from the format of input or explicitly the argument type.
- *Object getArgValue*(): return the parsed value of argument, invoking *parseArgValue*().
- *Object parseArgValue*(*Class argType*, *String argValue*): parse the argument value *argValue* to the corresponding type *argType*.

User Manual

Initialization and Quit

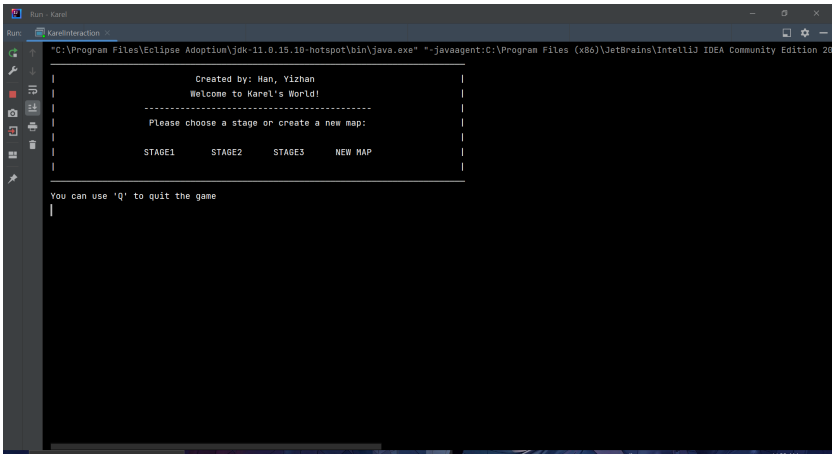

Figure 2: Welcome message

Once the program is executed, the welcome message will be printed. The user can input the stage number to choose the stage. If the input is not in the range of STAGE1 to STAGE3 or NEW MAP (also due to the limits of development when receiving inputs other than STAGE1 or STAGE2, the program will print "Not implemented yet. Choose another stage please."), the program will request the user to input again. The user can input Q to exit the program at any time.

Map and the Karel

After the stage choosing, the map and the karel will be printed. The icons are shown in the table1.

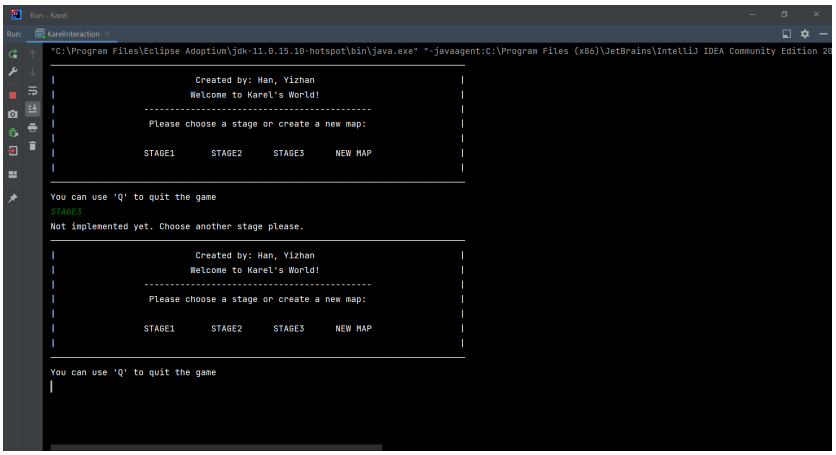

Figure 3: Not implemented yet

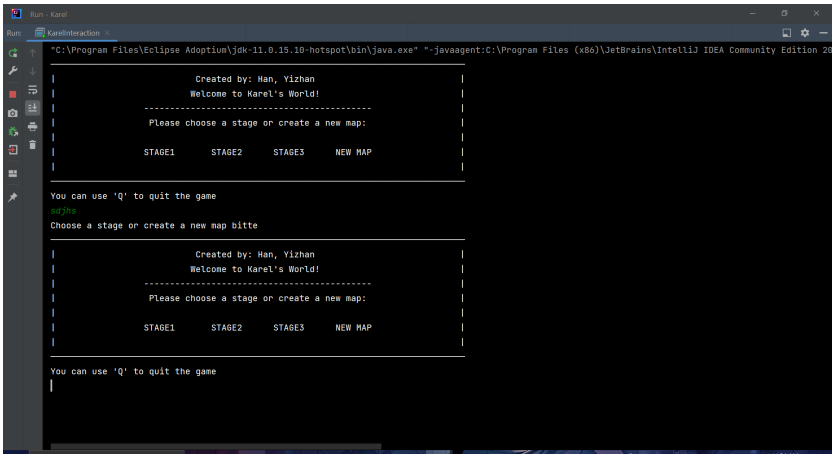

Figure 4: Please choose another stage

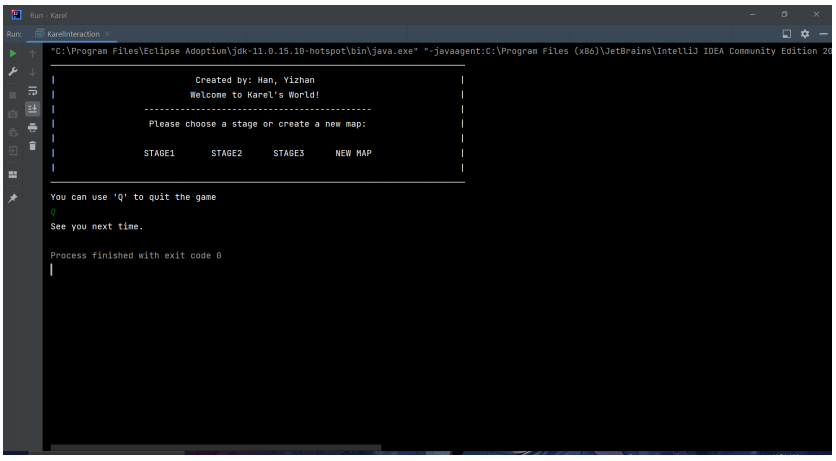

Figure 5: Exit message

Mark	Meaning
▲	Karel facing upwards
▼	Karel facing down
◀	Karel facing left
▶	Karel facing right
·	Ground
■	Wall
●	Rock

Table 1: marks in the map

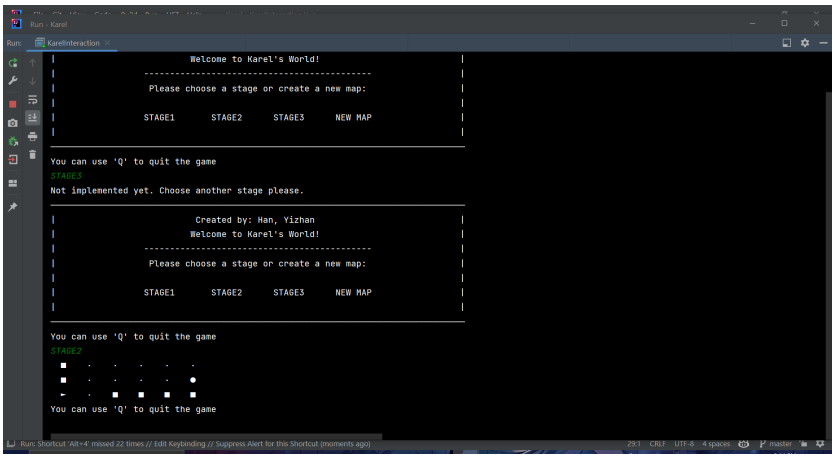

Figure 6: Map and the Karel

Actions

In the game, Karel can perform different types of actions corresponding to the given input. With undefined input there will be an exception "Not supported the command". The actions are shown in the table2.

When Karel is heading to an illegal site (e.g. a wall or a site out of the map), the program will throw an exception "The robot cannot move towards this direction anymore!" and the Karel will not move anymore. Also when Karel is instructed to pick a rock in front which is not existent the program will print out "There is no rock ahead! Please enter again."

Instruction	Description
move()	Karel moves one step forward
move(int x)	Karel moves x stpes forward
turnLeft()	Karel turns left
pickRock()	Karel picks the rock in front
showInformation()	Print the current state of the game

Table 2: instructions and actions

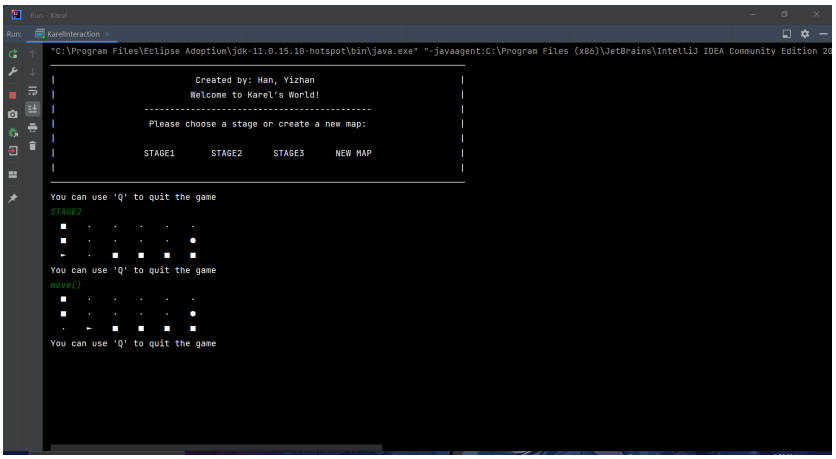

Figure 7: Karel moves one step forward

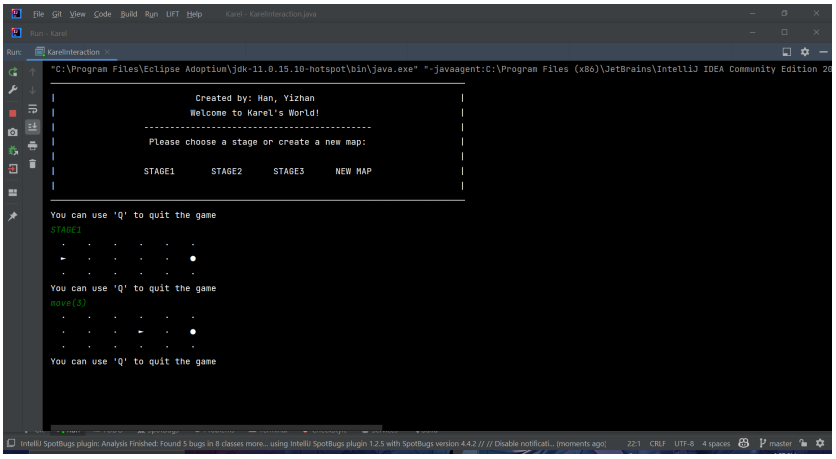

Figure 8: Karel moves 3 step forward

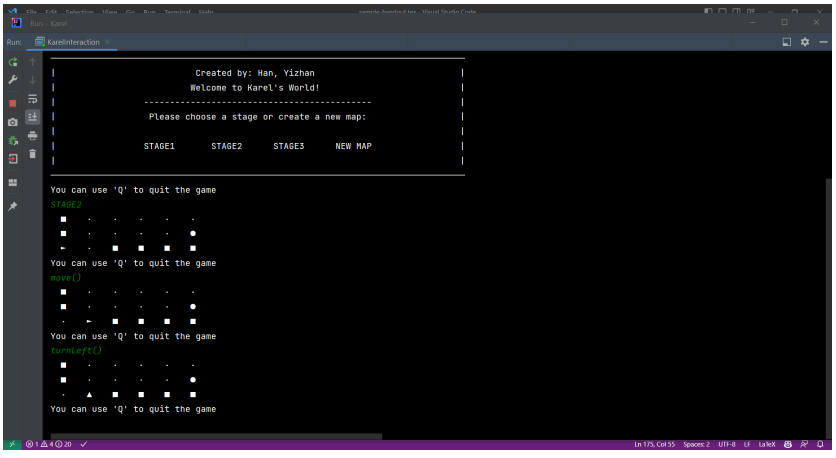

Figure 9: Karel turns left

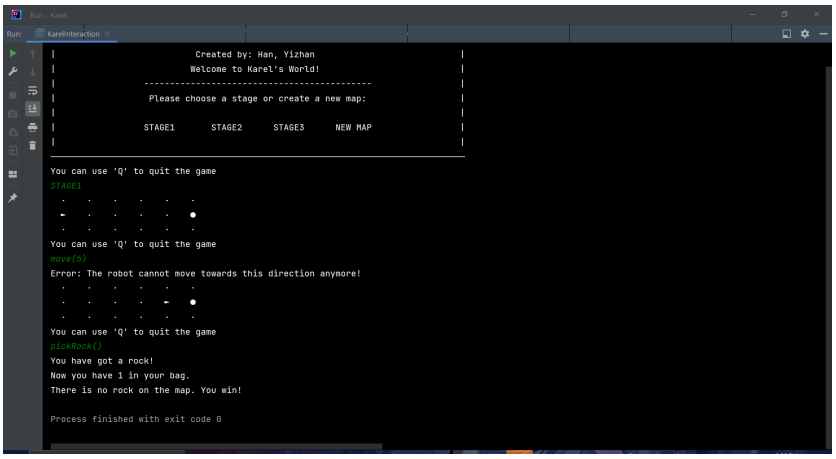

Figure 10: Karel picks one rock in front

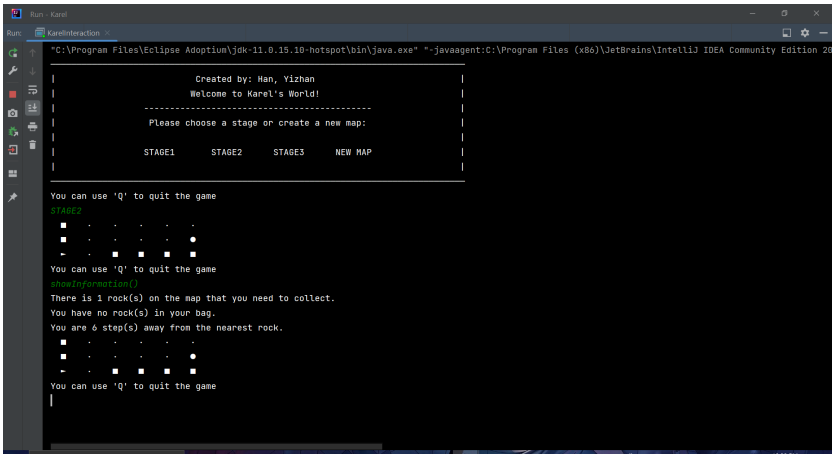

Figure 11: Print out the current state

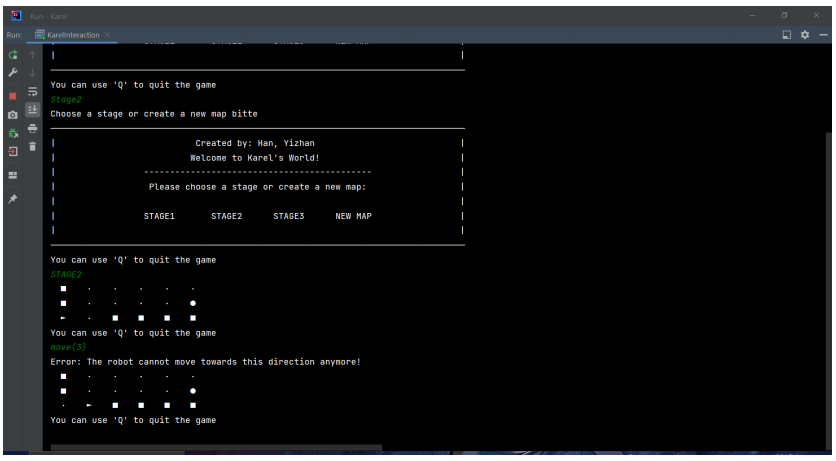

Figure 12: Karel cannot perform illegal move

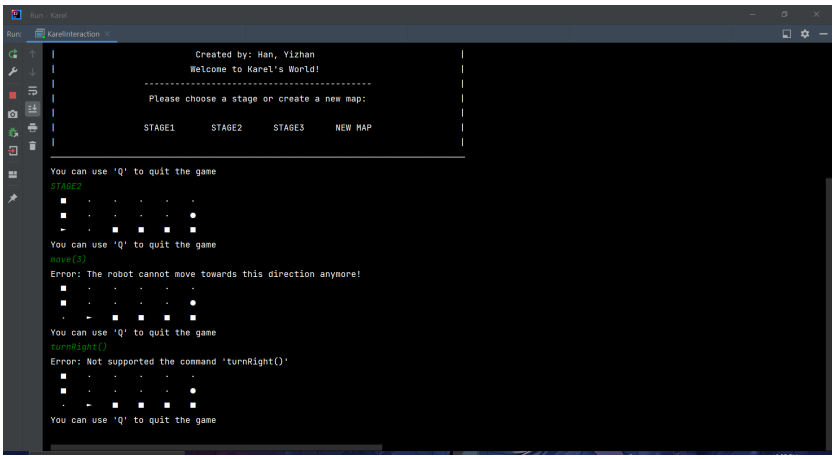

Figure 13: Karel with undefined Instructions

Win and Exit

When Karel has collected all the rocks in the map, the program will print out "There is no rock on the map. You win!" and exit.

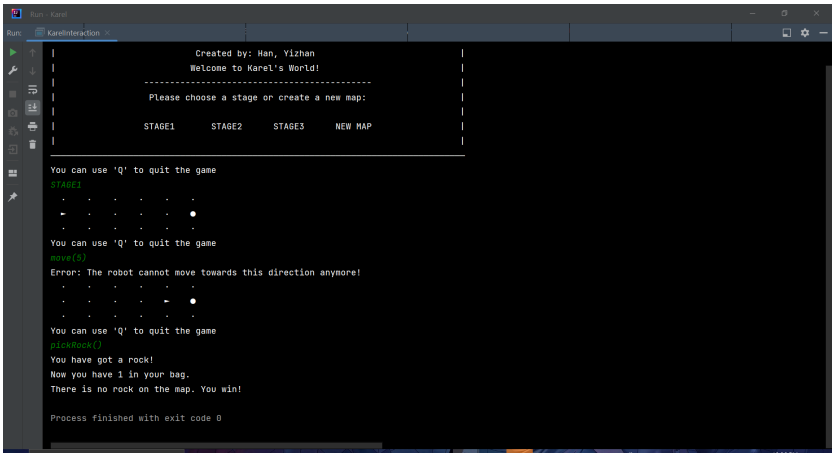

Figure 14: Winning Message