CODING TEST ANSWER

2019.07.28 YE JIAYI

1. REQUIREMENTS ANALYSIS

(* This is not a formal requirement analysis, just my understanding on the testing requirements.)

A. The car would be capable of moving forward into different direction, in my opinion, it can only move forward or backward, at the same time its orientation remain unchanged.

B. The car would be able to turn clockwise, at the same time its position remains unchanged.

C. The car park would be rectangular in shape with configurable dimensions. I.e. It should require configuration of the shape of the car park at the first place.

D. The car should remember its position and orientation, thus variables related to position and orientation should be kept private.

E. An exception will be thrown when the car moves outside the car park.

F. Implement the Car based on a provided interface, and implement as a Maven module, use library only those available in Maven Central.

G. Object-Orientation modelling for the application, keep minimal and viable for the application. The application should be easy, automatic, repeatable ran and built for maintainability purpose.

Unclear: By automatic, dose it means the application randomly generate it movement? I do not think it is reasonable, so I design the app to ask user to input next command after every single movement.

2. ENVIRONMENT

Java version: 1.8.0

Maven version: Apache Maven 3.6.1

IDE: Intellij Idea Ultimate 2019.2

3. CLASS DESIGN

For the convenience of extension and maintainable, I put the class in another file(CarType1.java) other than interface(Car.jave), and I also design create a carpark class for the parking lot.

The following are the source code for the CarType1 class implements based on the provided interface.

- Feel free to check the remaining code in my project !

`/test/src/main/java/com/leo/app/cars/CarType1.java`

```
package com.leo.app.cars;

import com.leo.app.carpark.CarPark;


public class CarType1 implements Car{

    private int positionX; //car's X_coordinate
    private int positionY;  //car's Y_coordinate
    private char orientation;  //car's orientation

    // an array store for calculating position after moving
    int direction[][]={{0,1}, {1,0}, {0,-1}, {-1,0}};
    // a String use for get the orientation after turning clockwise
    String dir = "NESW";

    /**
     * Three instructions can be executed on the car, i.e.moving FORWARD,
BACKWARD and Tuning clockwise.
     *
     * @param  command   the command(an upper-letter: F for moving FORWARD,
B for moving BACKWARD, T for Tuning clockwise. )
     */
    public void move(char command) {

        int now = dir.indexOf(getOrientation());  //get current orientation
        //get current position
        int x = getPositionX();
        int y = getPositionY();

        try {
            // Three types of command
            switch (command) {
                case 'F':
                    System.out.println("Moving Forward!");
                    x = direction[now][0] + getPositionX();
                    y = direction[now][1] + getPositionY();
                    break;
                case 'B':
                    System.out.println("Moving Backward!\n");
                    x = direction[now][0] * (-1) + getPositionX();
                    y = direction[now][1] * (-1) + getPositionY();
                    break;
                case 'T':
                    System.out.println("Turning clockwise\n");
                    if (now == 3) {
                        orientation = 'N';
                    } else {
                        orientation = dir.charAt(now + 1);
                    }
                    break;
                default:
```

```
                    System.out.println("---------Invalid input!---------\n"
+
                            "Please input F(moving FORWARD), B(moving
BACKWARD) or T(Tuning clockwise).");
            }
            // check if this command can be safely executed.
            isvalidMove(x, y);
        }catch (Exception e){
            e.printStackTrace();
        }

    }

    /**
     * Check if the car is outside the boundaries.
     *
     * @param  x   the command(an upper-letter: F for moving FORWARD, B for
moving BACKWARD, T for Tuning clockwise. )
     */
    public void isvalidMove(int x, int y) throws Exception{
        if(x > CarPark.PARK_LENGTH-1 || y > CarPark.PARK_WIDTH-1 || x < 0 ||
y <0){
            // array out of index, throw exception
            throw new Exception("Car is outside the parking lot!");
        }else{
            // valid operation, thus set car's new position
            setPositionX(x);
            setPositionY(y);
        }
  }

    /**
     * current positionX of the car.
     *
     * @return       current positionX of the car
     */
    public int getPositionX() {
        return this.positionX;
    }

    /**
     * current positionY of the car.
     *
     * @return       current positionY of the car
     */
    public int getPositionY() {
        return this.positionY;
    }

    /**
     * Increment a value by delta and return the new value.
     *
     * @return       current orientation of the car
     */
    public char getOrientation() {
        return this.orientation;
    }

    /**
     * Set car's X_coordinate.
     *
     * @param  x   new positionX for the car
     */
    public void setPositionX(int x) {
        this.positionX = x;
    }

    /**
```

```
     * Set car's Y_coordinate.
     *
     * @param  y   new positionY for the car
     */
    public void setPositionY(int y) {
        this.positionY = y;
    }

    /**
     * Set car's orientation.
     *
     * @param  orientation   the amount the value should be incremented by
     */
    public void setOrientation(char orientation) {
        this.orientation = orientation;
    }

}
```

4. APPLICATION TESTING

(*Following are some of my test results. The application meets the requirement according to preliminary test.)

Note: I use an array stands for the car park, of which the coordinate starts from zero, so the valid coordinate of the car should be [0~length-1, 0~width-1], an exception would be thrown if outside this range.

I use "+" to stand for a single point, so if the size of the car park is 4*4. Then it will be like as follow.

++++

++++

++++

++++

The bottom leftmost point stands for coordinate (0, 0).

And I use letter to represent the orientation of the car, 'N' stands for North, 'S' for South, 'E' for East, 'W' for West. Current state will be printed after any movement.

First, ask for parking lot size configuration, if enter zero, then use the default setting, i.e. both length and width are equal to 4.

```
Please enter the length of the parking lot:
0
Please enter the width of the parking lot:
0
The length and width cannot be zero.
Use default setting: 4*4
```

If enter negative, then it will be automatically converted to its absolute.

```
Please enter the length of the parking lot:
-5
Please enter the width of the parking lot:
-5
```

The input above will be interpreted as both side equal to five.

If input is not integer, then it will demand to input integer.

```
Please enter the length of the parking lot:
k
PLEASE CHECK YOUR INPUT! INTEGER ONLY!
Please enter the length of the parking lot:
```

Then, ask the initial coordinate of the car and its orientation, it will also do the checking.

```
Please enter the initial coordinate 'X' of the car:
2
Please enter the initial coordinate 'Y' of the car:
2
Please enter the initial orientation of the car:
(It can only have for direction: N(North), E(East), S(South) and W(West)
E
++++
++E+
++++
++++
Current position: (2, 2)
Orientation: E
```

Then, ask the movement, it will also check the input.

```
Please input movement:(F: move forward, B: move backward, T: turn clockwise)
t
Turning clockwise

++++
++S+
++++
++++
Current position: (2, 2)
Orientation: S

Please input movement:(F: move forward, B: move backward, T: turn clockwise)
t
Turning clockwise

++++
++W+
++++
++++
Current position: (2, 2)
Orientation: W
```

```
Please input movement:(F: move forward, B: move backward, T: turn clockwise)
f
Moving Forward!
++++
+W++
++++
++++
Current position: (1, 2)
Orientation: W

Please input movement:(F: move forward, B: move backward, T: turn clockwise)
f
Moving Forward!
++++
W+++
++++
++++
Current position: (0, 2)
Orientation: W
```

Now, the car is at (0, 2), the orientation is WEST, if it moves forward, i.e. move west for even a unit, it will be outside the parking lot, thus throw an exception.

```
Please input movement:(F: move forward, B: move backward, T: turn clockwise)
f
Moving Forward!
java.lang.Exception: Car is outside the parking lot!
    at com.leo.app.cars.CarType1.isvalidMove(CarType1.java:70)
    at com.leo.app.cars.CarType1.move(CarType1.java:55)
    at com.leo.app.appmain.main.main(main.java:124)
++++
W+++
++++
++++
Current position: (0, 2)
Orientation: W
```

Moving backward.

```
Please input movement:(F: move forward, B: move backward, T: turn clockwise)
b
Moving Backward!

++++
+W++
++++
++++
Current position: (1, 2)
Orientation: W
```

Enter ‘q’ or ‘Q’ to quit.

```
Please input movement:(F: move forward, B: move backward, T: turn clockwise)
q
quit!
```

5. USAGE

I directly export the jar using Intellij Idea. I have already declare the main class, so you can enter the following command on the console to access my little application.

java -jar test-driveless-car.jar