TYPES II – FIELDS AND METHODS

JAVA FUNDAMENTALS

Lab 9, Types II – Fields and Methods

Objective

You will further your understanding of object oriented programming.
In this lab by practising creating classes and overloading methods.

Part 1 – The Map Room.

Consider a large map as used in the Battle of Britain (see above). Various tokens are placed on the map at specified coordinates. As new information comes in, the positions of the tokens are updated by pushing them with rakes to new coordinates.

1. Back in the labs project which you created in *Lab1*, add a new package called lab09.

   Please refer to lab1's instructions if you need help.

2. Create a class called Program in this package. Give it a main() method.
3. Create a class called Map. This requires instance variables xSize and ySize, to be set by the constructor. Add getters to read xSize and ySizez.
4. Create a class called Token. This requires instance variables for x and y coordinates, and getters and setters. A third instance variable, map, is of type Map.
5. Add a constructor to Token to set the initial values of x and y, and an object of type Map to be stored in map.

6. Add a method to Token called move(). Its parameters are direction (a String) and distance (an int). The possible values for direction are “north”, “south”, “east” and “west”. The return type of move() is boolean. For now, have it return true.

7. With the map coordinates starting at the bottom left, add code to move() so that if a token has the coordinates (20, 30), a call of move(“east”, 40) will change the coordinates to (60, 30), i.e. a move east will increase the x coordinate but leave the y coordinate alone. Ensure that the token remains within the bounds of the map, e.g. use the Map object’s xGetter() to check that an x coordinate of 60 is possible.

   For the sake of simplicity, assume that distances are in miles, and movement of one mile corresponds to a change of coordinate of one.

8. In main(), create a single instance of Map and store its reference in a variable called map. Call the constructor with the arguments 500 and 400.

9. Also in main(), create three instances of Token – t1, t2 and t3 – with suitable x and y values, making use of the same Map object, e.g.

   ```
   Token t1 = new Token(70, 30, map);
   ```

   In real terms, you have produced a single map, and you have placed three tokens on it.

10. Still in main(), print the coordinates for all three tokens using the getters. Then move each token using the move() method, e.g.

    ```
    t1.move(“east”, 100);
    ```

    Print the new coordinates.

11. Once you have this working, change the code in move() so that If the new coordinates are “impossible”, e.g. x is less than zero or greater than 500, have the move() method print, “The token has fallen off the map,” and return false. Have main() read the return values of the calls to move(). If move() returns false, delete the reference. e.g.

    ```
    if(t3.move(“north”, 200) == false) t3 = null;
    ```

12. Add code to the move() method so that it is not case sensitive, i.e. it will accept “south”, “SOUTH”, “South” etc. Enable it to also accept the initial letter – “n”, “s”, “e”, “w”.

13. Slight stretch: Overload move() so that it will accept an int instead of a String for direction and treat the value as degrees clockwise from north, i.e. 0 is north, 90 east, 180 south and 270 is west.

14. Bigger stretch: If you are mathematically inclined, use Java’s Math library to allow direction values other than the cardinal ones (i.e. divisible by 90). For example,

    ```
    t1.move(60, 50):
    ```

will move the token 50 miles, with an x coordinate change of 40 and a y coordinate change of 30.

Hint: the x coordinate changes by cosine(direction) * distance, and the y coordinate by sine(direction) * distance. Don’t forget that Java treats angles as radians by default rather than degrees so you may need to look up the toRadians() method in Math.

Part 2 – Speed Limit

In this part you'll make use of static keyword.

If the map is updated hourly, then the distances moved by each token must be limited to its maximum speed, e.g. a Jeep in the 1940s could do 65mph. If the tokens are all of the same type then they will have the same maximum speed.

(In a later exercise we will see how to create tokens of different types.)

If a token is moved an "impossible" distance, the operator must be alerted. (It may be that the previous position was incorrect, but the new position is correct.)

1. Create a static int field in Token called maxSpeed and a suitable value such as 65.

2. Change the move() method so that it compares distance with maxSpeed. If distance is in excess of maxSpeed, print the message, "Please check data."

** End **