3S03 Assignment 3

Ethan Walsh - 400383566 - walshe11
Ahsan Muzammil - 400455916 - muzammia

March 27, 2025

Question 1

Class Objectives	Ensure that all methods and constructors meet expected behaviour. This includes usual operations such as getting and setting the direction, speed, and its components (speedX and speedY). Boundary values and unexpected inputs should be tested for a complete and robust test suite.
Inspection/Review Requirements	May require inspection in order to understand the interface implementations - for example, how the components (speedX and speedY) are calculated. This will give the test developers a better idea of what to target, taking advantage of any vulnerabilities that can be seen during white box testing.
Specification-Based Testing Objectives	Assuming this is more aligned with the game's function (reversing direction when at a boundary, speedY remains the same and speedX is multiplied by -1 when colliding with side walls).
Implementation-Based Testing Objectives	May be helpful for components calculations (cosine etc.)
Interaction-Based Testing Objectives	interaction between reverse, as well as reverseX and reverseY may be worth testing (as it's not immmediately clear why all 3 are needed, except maybe for corner hits when both components are reversed).

General planning ideas:
- For speed, input partitioning (negative max, negative integer, zero, positive integer, positive max) - For direction, testing overflow (going from 359 back to zero) and standard angles

Question 2

Contract - weakest precondition, strongest postcondition for all methods.

Class invariant:

- $\text{speed} \in \mathbb{Z}$ (speed is always non-negative)
- $\text{direction} \in [0, 360) \land \text{direction} \in \mathbb{Z}$ (direction is always a valid angle)
- $\text{speedX} = \text{speed} \cdot \cos(\text{direction})$
- $\text{speedY} = \text{speed} \cdot \sin(\text{direction})$

`public Velocity()`

Pre-condition:

- No input constraints (default constructor).

Post-condition:

- speed = 0, speedX = 0, speedY = 0.
- direction is initialized to a default value (e.g., 0).

`public Velocity(Speed speed, Direction direction)`

Pre-condition:

- speed $\in \mathbb{Z}$
- direction $\in \mathbb{Z} \land$ direction $\in [0, 360)$

Post-condition:

- this.speed = speed
- this.direction = direction
- this.speedX = speed $\cdot \cos($direction$)$
- this.speedY = speed $\cdot \sin($direction$)$

`public Speed getSpeed()`

Pre-condition:

- speed $\in \mathbb{Z}$.

Post-condition:

- Returns speed.
- this.speed $= \frac{\text{speedX}}{\cos(\text{direction})}$.
- this.speed $= \frac{\text{speedY}}{\sin(\text{direction})}$.

`public Speed getSpeedX()`

Pre-condition:

- speedX $\in \mathbb{Z}$.

Post-condition:

- Returns speedX and this.speedX = speed $\cdot \cos($direction$)$.

`public Speed getSpeedY()`

Pre-condition:

- speedY $\in \mathbb{Z}$.

Post-condition:

- Returns speedY and this.speedY = speed $\cdot \sin($direction$)$.

`public Direction getDirection()`

Pre-condition:

- direction $\in \mathbb{Z} \land$ direction $\in [0, 360)$

Post-condition:

- Returns direction, this.direction = direction, and direction $\in [0, 360) \land$ direction $\in \mathbb{Z}$.

`public void setSpeed(Speed speed)`

Pre-condition:

- speed $\in \mathbb{Z}$

Post-condition:

- this.speed = speed
- this.speedX = speed $\cdot \cos$(direction)
- this.speedY = speed $\cdot \sin$(direction)

`public void setDirection(Direction direction)`

Pre-condition:

- direction $\in \mathbb{Z} \land$ direction $\in [0, 360)$

Post-condition:

- this.direction = direction
- this.speedX = speed $\cdot \cos$(direction)
- this.speedY = speed $\cdot \sin$(direction)

`public void reverse()`

Pre-condition:

- speedX $\in \mathbb{Z}$.
- speedY $\in \mathbb{Z}$.
- direction $\in \mathbb{Z} \land$ direction $\in [0, 360)$

Post-condition:

- this.direction = (direction + 180)%360.
- this.speedX = speed $\cdot \cos$(direction)
- this.speedY = speed $\cdot \sin$(direction)
- direction $\in [0, 360) \land$ direction $\in \mathbb{Z}$

`public void reverseX()`

Pre-condition:

- speedX $\in \mathbb{Z}$.
- direction $\in \mathbb{Z} \land$ direction $\in [0, 360)$

Post-condition:

- this.speedX = $-$speedX.
- this.direction = $(180 - \text{direction}) \ \% \ 360$.

`public void reverseY()`

Pre-condition:

- speedY $\in \mathbb{Z}$.
- direction $\in \mathbb{Z} \wedge$ direction $\in [0, 360)$

Post-condition:

- this.speedY $= -$speedY.
- this.direction $= (-\text{direction}) \ \% \ 360$.

Question 3

Informally defined set of test cases for:
setDirection (1), setSpeed (1), reverse (5), reverseX (5)

```
1   public class TestVelocity
2   {
3       @Test
4       public void testSetDirection()
5       {
6           Velocity puck = new Velocity(5, 90);
7           puck.setDirection(180);
8           assertEquals(180, puck.getDirection());
9       }
10
11      @Test
12      public void testSetSpeed()
13      {
14          Velocity puck = new Velocity(5, 90);
15          puck.setSpeed(10);
16          assertEquals(10, puck.getSpeed());
17      }
18
19      @Test
20      public void testReverseWithCurrentDirectionOnXAxisOnly()
21      {
22          Velocity puck = new Velocity(5, 0);
23          puck.reverse();
24          assertEquals(180, puck.getDirection());
25          assertEquals((int) (puck.getSpeed()*Math.cos(180)), puck.getSpeedX());
26          assertEquals((int) (puck.getSpeed()*Math.sin(180)), puck.getSpeedY());
27      }
28
29      @Test
30      public void testReverseWithCurrentDirectionOnYAxisOnly()
31      {
32          Velocity puck = new Velocity(5, 90);
33          puck.reverse();
34          assertEquals(270, puck.getDirection());
35          assertEquals((int) (puck.getSpeed()*Math.cos(270)), puck.getSpeedX());
36          assertEquals((int) (puck.getSpeed()*Math.sin(270)), puck.getSpeedY());
37      }
38
39      @Test
40      public void testReverseWithCurrentDirectionOnDiagonalLessThan180()
41      {
42          Velocity puck = new Velocity(5, 45);
43          puck.reverse();
```

```
44              assertEquals(225, puck.getDirection());
45              assertEquals((int) (puck.getSpeed()*Math.cos(225)), puck.getSpeedX());
46              assertEquals((int) (puck.getSpeed()*Math.sin(225)), puck.getSpeedY());
47          }
48
49          @Test
50          public void testReverseWithCurrentDirectionOnDiagonalGreaterThan180()
51          {
52              Velocity puck = new Velocity(5, 190);
53              puck.reverse();
54              assertEquals(10, puck.getDirection());
55              assertEquals((int) (puck.getSpeed()*Math.cos(10)), puck.getSpeedX());
56              assertEquals((int) (puck.getSpeed()*Math.sin(10)), puck.getSpeedY());
57          }
58
59          @Test
60          public void testReverseWithCurrentDirectionEqualTo359()
61          {
62              Velocity puck = new Velocity(5, 359);
63              puck.reverse();
64              assertEquals(179, puck.getDirection());
65              assertEquals((int) (puck.getSpeed()*Math.cos(179)), puck.getSpeedX());
66              assertEquals((int) (puck.getSpeed()*Math.sin(179)), puck.getSpeedY());
67          }
68
69          @Test
70          public void testReverseXWithCurrentDirectionOnXAxisOnly()
71          {
72              Velocity puck = new Velocity(5, 0);
73              puck.reverseX();
74              assertEquals(180, puck.getDirection());
75              assertEquals(180, puck.getSpeedX());
76          }
77
78          @Test
79          public void testReverseXWithCurrentDirectionOnYAxisOnly()
80          {
81              Velocity puck = new Velocity(5, 90);
82              puck.reverseX();
83              assertEquals(0, puck.getDirection());
84              assertEquals((int) (puck.getSpeed()*Math.cos(0)), puck.getSpeedX());
85          }
86
87          @Test
88          public void testReverseXWithCurrentDirectionLessThan180()
89          {
90              Velocity puck = new Velocity(5, 179);
91              puck.reverseX();
92              assertEquals(1, puck.getDirection());
93              assertEquals((int) (puck.getSpeed()*Math.cos(1)), puck.getSpeedX());
94          }
95
96          @Test
97          public void testReverseXWithCurrentDirectionGreaterThan180()
98          {
99              Velocity puck = new Velocity(5, 225);
100             puck.reverseX();
101             assertEquals(315, puck.getDirection());
102             assertEquals((int) (puck.getSpeed()*Math.cos(315)), puck.getSpeedX());
103         }
104
105         @Test
```

```
106         public void testReverseXWithCurrentDirectionEqualTo359()
107         {
108             Velocity puck = new Velocity(5, 359);
109             puck.reverseX();
110             assertEquals(181, puck.getDirection());
111             assertEquals((int) (puck.getSpeed()*Math.cos(181)), puck.getSpeedX());
112         }
113     }
```