SFWRENG 3S03 Software Testing (2025 Winter)
Assignment 3

Administrative information

Weight of the assignment: this assignment is worth 10% of your final grade. It consists of three questions for a total of 70 marks, and a quality multiplier (1.0-0.5) that reflects the overall quality of your report.

How to complete the assignment?

- Answer all questions.
- Submit your solutions on Avenue.
- Submit your report with your answers as a single PDF document.
- In your PDF document, report the key points and the key artifacts. Argue your points when asked to.

Groupwork

You can work in a group of up to three people, although you don't have to. You can also switch groups and work in different groups than in the first two assignments. Everyone in the group needs to submit the very same PDF file.

Due date: April 6, 2025, at 11:59 PM ET.

Assignment: Testing Breakout

The following scenario applies to all three questions below. You are building a 70s-style game like Breakout, as shown in Figure 1.

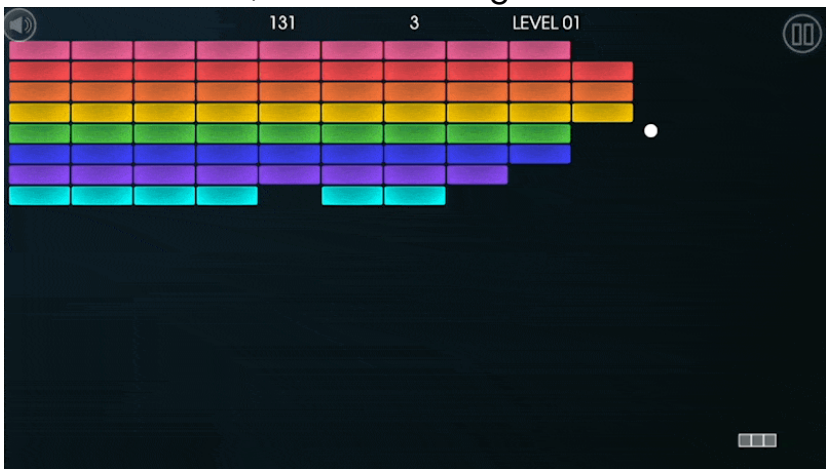

Figure 1 Breakout

The objective of the game is to move a player horizontally at the bottom of the display to hit a puck. The puck will bounce off the player, and the edges of the screen, at a particular velocity. The player should attempt to hit the puck in a way so that it collides with bricks at the top of the screen. There will be multiple rows of bricks. Each collision between puck and bricks will destroy one or more bricks, depending on velocity and position. Once all bricks have been destroyed, the round is successfully finished. If the player misses the puck and it drops off the bottom of the display, a puck is lost. The player has three pucks, and once all three have been lost, the player's game is over.

Consider the class `Velocity`, used in Breakout. Its public interface is listed below. `Velocity` tracks the velocity (speed and direction) of the puck that is currently in play. An instance of `Velocity` represents the movement of a puck on a playfield based on attributes of speed (expressed in playfield units per unit time) and a direction (expressed as an angle in degrees; 0 represents east or right; 90 designates north or up; and so on). The `speed` attribute is broken up into two components: `speedX` (speed in the x direction, i.e., left-right) and `speedY` (speed in the up-down direction). While the speed attribute is always non-negative, the components of a velocity's speed can be negative. The value of `speedX` is negative if a velocity's direction is heading left. The value of `speedY` is negative if the direction is down. `Speed` and `Direction` are abstract types that ultimately are defined as integer values.

```
public class Velocity {
  private Speed speed, speedX, speedY;
  private Direction direction;

  public Velocity(); //constructor
  public Velocity(Speed speed, Direction direction); //constructor

  public Speed getSpeed();
  public Speed getSpeedX(); // get speed in X direction
  public Speed getSpeedY(); // get speed in Y direction
  public Direction getDirection();

  public void setSpeed(Speed speed);
  public void setDirection(Direction direction);
  public void reverse();   // reverse the direction of the puck!
  public void reverseX();  // reverse the direction of the puck in x-axis
  public void reverseY();  // reverse direction of the puck in the y-axis
}
```

1. Using a table, specify a test plan for the `Velocity` class. [15 marks]

Your test plan should state the following:

- Objectives for the class
- Any requirements for inspection/review of the class. Justify why you may need inspection/review.
- Brief statement of objectives for specification-based testing for the class.
- Brief statement of objectives for implementation-based testing for the class.
- Brief statement of objectives for interaction-based testing (i.e., interactions between particular methods of the class).

2. Based on what was discussed in lectures, in terms of contracts and contract-based specification, write preconditions and postconditions for each method of `Velocity` (including the two constructors). Also, write a class invariant: it specifies (as a `boolean` expression) properties that must be true for all instances of class `Velocity`. [35 marks]

Hint: think about the constraints on the attributes of `Velocity` and relationships between these attributes.

Hint: some of the preconditions and postconditions are very simple, others are more complex.

3. Informally define a set of test cases for the following methods:

- `setDirection` – define exactly one test case
- `setSpeed` – define exactly one test case
- `reverse` – define 5 test cases
- `reverseX` – define 5 test cases

[20 marks]

Your tests may be written out in English, pseudocode, or JUnit, whichever you find easiest to use; if you are using English or pseudocode, please be sure that we can understand your answers. Each test case must specify the input to the test case and the expected outputs.

4. Overall quality of the report

Your report should be of reasonable quality. Your intent should be to report your experiences and argue your points, not to copy-paste template text.
To obtain your final mark, the sum mark of Tasks 1-2-3 will be scaled as follows.

Quality	Example characteristics	Multiplier
Good	Concisely and precisely reports artifacts, results, and if the apply, assumptions and limitations. Clearly argues points. Figures and images are of proper resolution. Uses proper English.	1.0
Medium	Reports artifacts and results at an acceptable level. Makes points without solid arguments. Awkward and hard-to-follow figures and images. Typos in text.	0.75
Poor	Clearly lacks effort at producing a quality report. Only reports data; interpretations and arguments are completely missing. Figures and images are of poor resolution. Poor English.	0.5