In-Class Exercise 6: Moving Ball and Inner Wall Interactions

Instructors: Mackale Joyner, Stephen Wong

Course Website: `https://www.clear.rice.edu/comp504`

Goals for this exercise

- Changing object behaviors after object collisions
- Identifying and fixing unintended consequences that arise due to object interactions

Important tips and links

NOTE: The instructions below are written for Mac OS and Linux computers, but should be easily adaptable to Windows with minor changes. For example, you may need to use \ instead of / in some commands.

1 Checkout Github Classroom Exercise 6 Repo

First, you'll need to accept the ex 6 repository by visiting GitHub ex6. Check out your github classroom exercise 6 starter code from the remote github repo in IntelliJ from either the "Check out from Version Control" option on the welcome screen or the "Checkout from Version Control" option accessible under the top menu's VCS tab. Use the plus button in the pop-up window to add your GitHub ex6 repo. You should see a ex6 directory created in your working directory with the source code for this exercise.

2 Exercise

For this exercise, there are stationary inner walls and moving balls. A ball will eat an inner wall when the ball collides with it. The walls must fit within the ball's stomach after the ball eats the inner wall. The ball's stomach is the invisible line that runs vertically from the ball center to the bottom of the ball. The stomach length is the ball radius. When the ball eats an inner wall larger than the ball's radius, the inner wall will be resized to the ball's radius so that it fits in the ball's stomach. The inner wall is then moved to the ball's stomach. When the ball eats an inner wall that fits in the ball's stomach, the inner wall will be moved to the ball's stomach with the same wall size. The wall should remain in the ball's stomach.

3 Model

The inner walls are stationary by default. The balls have a random velocity and travel straight horizontally. Both balls and inner walls implement the `PropertyChangeListener` interface. When a property change event occurs, all property change listeners should execute the command passed to them.

You'll need to modify the `execute` method in the `UpdateStateCmd`. This method will determine if a moving ball has collided with an inner wall. If so, the command should override the normal behavior of the inner wall. The ball eats inner walls when colliding with them as described in the previous section. Balls and inner walls should behave normally when there is no collision. You cannot use any control logic in either the `Ball` or `Wall` `propertyChange` method or in the strategy. The logic should be in the `UpdateStateCmd`.

You'll also need to modify the `updateBallWorld` and `loadPaintObj` methods in the BallWorldStore. There's a hint in `updateBallWorld` to pass the `UpdateStateCmd` as an argument to `firePropertyChange`. The `Ball` and `Wall` concrete classes will just execute the command when there's a property change.

4 Demonstrating and submitting the exercise

You should be able to start up a local server and draw multiple moving balls and stationary inner walls. The balls will eat inner walls when they collide with them. You'll next need to host your app on heroku. The app name should be [netid]-ex6-ball-wall. Place the app name and your name in the README.md file.

Please don't forget to commit and push your work to your github classroom repository. To perform a git commit, select VCS on the menu and click on "Commit...". Add a commit message and click on the `Commit` button. To push the local changes, select VCS on the menu, highlight the Git option and select "Push...".