In-Class Exercise 5: Moving Lines with the Command Design Pattern

Instructor: Mackale Joyner

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

Goals for this in-class exercise

- Get familiar with the Command Design Pattern

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 In-Class Exercise 5 Repo

First, you'll need to accept the in-class exercise 5 repository by visiting GitHub ex5. Check out your github classroom exercise 5 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 ex5 repo. You should see a ex5 directory created in your working directory with the source code for this lab.

2 Exercise 5

For this exercise, you will use the Command design pattern to draw lines that move across the canvas. Recall that the Command design pattern separates the invoker from the receiver. The command `execute` method will determine which receiver is affected by the command. In this exercise, the model will determine where to initially draw the moving lines and how fast each line should move across the canvas. For a moving line, the view will update the line position every .2 seconds.

The controller will process REST method requests from the view and call the appropriate `DispatchAdapter` function in the model to update the line positions. The controller may also signal the adapter to switch strategies for both lines in the `LineWorld`. The Command design pattern should be used to implement this functionality. The next sections detail what you'll need to implement in the model, view, and controller.

2.1 Model

The line will begin moving once it is drawn on the canvas. The initial strategy is chosen randomly. If the strategy is the horizontal strategy, the horizontal strategy will move the line horizontally to the right. There is no collision detection with the right wall of the canvas. As a result, the line will keep moving right until the line moves off the canvas.

Each line should switch strategies when the user clicks on the `switch strategy` button. The horizontal strategy will switch to the vertical strategy. The vertical strategy moves the line vertically down. A line will keep this strategy until the `switch strategy` button is clicked again. The vertical strategy will switch to the composite strategy. The composite strategy has the horizontal and vertical strategies as children. Recall, the only job the composite strategy has is to call each child's update operation. All lines with the composite strategy WILL NOT MOVE. You cannot use any control logic in the `CommCmdAdapter sendRecvCmd` method. The only thing you can do is call the command's `execute` method for the short and long line. The moving lines will keep using the composite strategy until the `switch strategy` button is clicked again. The composite strategy switches back to the horizontal strategy. The only `TODO` in `CommCmdAdapter` is to implement the `sendRecvCmd` method.

The `DispatchAdapter` sits in between the client (user) and the model. The `DispatchAdapter` will communicate with the controller to determine what type of command should be sent to both lines in the `LineWorld`. Each `DispatchAdapter` method used to communicate with the controller and the model needs to be implemented. The singleton design pattern should be used where appropriate. Again, there should be no control logic in the `CommCmdAdapter sendRecvCmd` method.

2.2 Controller

You'll need to finish implementing the endpoints in the `LineDrawController` class. The controller interacts with the model by using the `DispatchAdapter` to update the `LineWorld`.

2.3 View

You'll need to implement the `update` endpoint in `view.js`. The view ensures that the lines start moving immediately after they are drawn on the canvas. The view updates the line position every .2 seconds. After implementing the view endpoints, you should be able to start up a local server, draw short and long moving lines, and switch strategies for both lines every time the user clicks the `switch strategy` button. A line with the `CompositeStrategy` should not move. The line should start moving again once it has switched strategies.

3 Demonstrating and submitting lab work

You'll need to host your app on heroku. The app name should be [netid]-ex5-move-line. Place the app name and your name in the README 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...".