Project 2: DragonBar (Chapters 11 & 12)

Due: Friday, May 10, 2019

You will refactor `DragonLand`'s code into `DragonBar`, a version of the game played via a GUI. Code needed for this project is posted along with this write-up. Download the zipped folder: `Project 2 (DragonBar).zip`, containing an ECLIPSE project.

Binary File & Exceptions

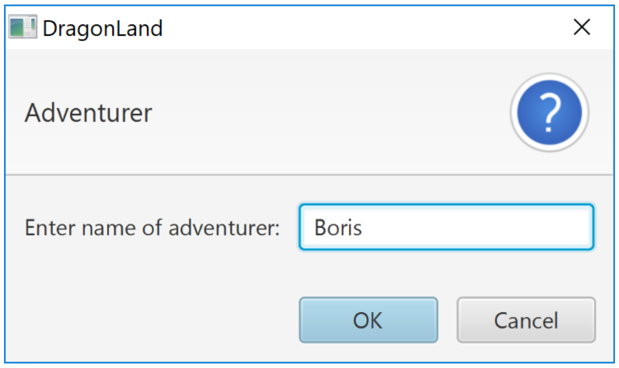
When the program starts, create a *binary* log file (`actions.log`). Record each action of the user in that file. Action records should be a sequence # (1, 2, …) and a number representing the action: 1=Work, 2=Eat, 3=Nap, 4=Quit. *Close the file when the user quits the application.* **Catch any exceptions (display a message on the console in such cases).**

GUI

The game will now be played via a GUI rather than the console.

Dialogs

- When the program starts, ask for the adventurer's name using a dialog box (use the JAVAFX class `TextInputDialog`). E.g.,

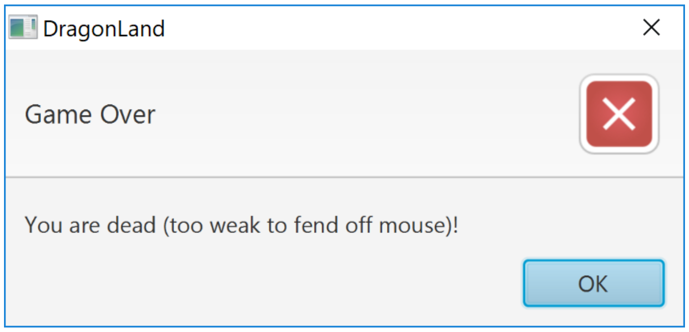


DragonLand

Adventurer

Enter name of adventurer: Boris

- Display messages when the game ends via dialog boxes. E.g.,



DragonLand

Game Over

You are dead (too weak to fend off mouse)!

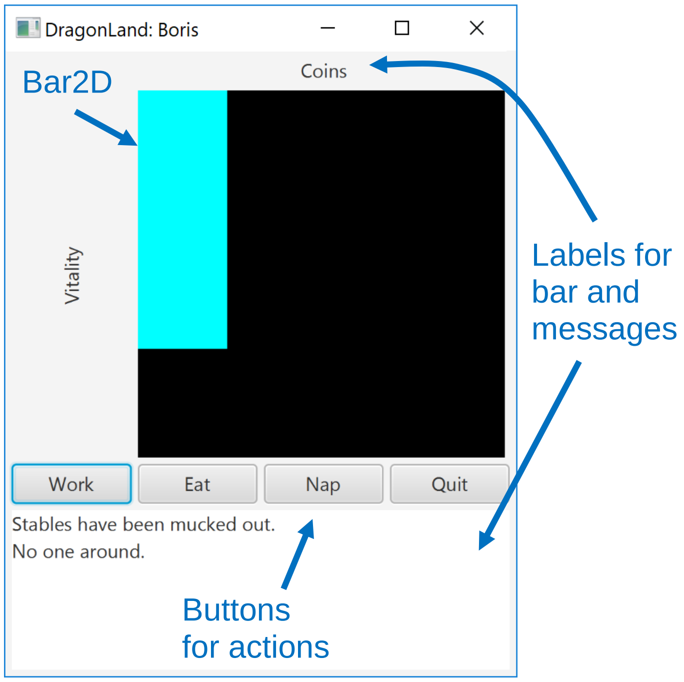
Information about the adventurer's status as well as any confrontation will be displayed in components (see next page); and thus, does not require dialogs.

Canvas

Create a custom component (class name `Bar2D`) that shows 2 fractions as a 2D bar by adapting class `EggPanel` from lecture code (provided). Here, the component should display the values (e.g., coins/vitality) using a bar drawn from the top, left. Recall coins go from 0 to the winning amount of 30 and vitality from 0.0 to 1.0. See example below.

Custom Window

After the user enters the adventurer's name, create a window that replicates the layout on the right, with a grid to display: the adventurer's *coins* and *vitality*, a label for textual *messages*, and buttons to perform *actions*.



Stables have been mucked out.
No one around.

- *Coins* and *vitality* are shown using a 2D bar.
    - Width of the bar represents fraction of coins toward the goal (on right, coins is 7/30).
    - Height of the bar represents fraction of maximum vitality (on right, vitality is 0.7).

  Be sure to draw a few pixels even if the fraction is zero (else no bar will be visible).

- Messages go in the bottom label using **black text on a white background.** Make it tall enough for multiple lines of text.

- Rotate a component using: `component.setRotate(angle);`

- Control the alignment of text using the grid pane in which it is placed:

  ```
  pane.setValignment(component, VPos.CENTER);
  pane.setHalignment(component, HPos.LEFT);
  ```

- Use `setMinWidth` and/or `setMaxWidth` to control the size of components. **Also note that some components should span multiple positions in the grid.** This can be accomplished via an overloaded version of `add`:

  ```
  pane.add(component, col, row, colSpan, rowSpan);
  ```

Buttons

Include 4 buttons for actions. Respond to user clicks with `EventHandlers`. **Write them as *nested classes*.** Because actions are now controlled via buttons, there will no longer be a loop in `main()`. Each button should perform the requested action plus any confrontation, update the window's data, and check whether the player won/died.

Titlebar

Show "DragonLand" in the titlebar of all windows/dialogs. *The custom window's titlebar should also include the adventurer's name.*

Tips

- Import all the JAVAFX classes you will need.
- Make class `DragonLand` inherit from `Application`, override `start`, and call `launch` from `main`.
- Write a method to update the 2D bar and check to see if the player won/lost. Call it after each confrontation.
- Write a method to record the user's action in the log file and deal with any exceptions.
- Constants/variables may be moved to other methods (or to the class level) if needed. The adventurer and villains have already been placed at the class level.

Standards

Adhere to our standards for naming variables/constants, aligning code, and writing explanatory comments when needed. **Include the file name, your name, project name, course name, date, and a description at the top of each source file you edit.**

Your canvas drawing class (`Bar2D`) should not be hardcoded for coins and vitality, it should be usable with other values.

Test your program by playing the game multiple times.

Submission

Complete the Java application in ECLIPSE (downloadable from eclipse.org/downloads/).

You also need the Java Standard Edition Development Kit (JDK) on your machine (available at `oracle.com/technetwork/java/javase/downloads/`).

Name your top-level folder `Project 2 (DragonBar)` *`Your Name`*. Compress that folder in `ZIP` format and upload it to Canvas. **Delete all the `.class` files under the `bin` folder before compressing, but leave the `.java` source code files under `src`.**

Bonuses

You may attempt these bonuses once you complete the program above.

a) Record an action of 5=Close when the user clicks the Close button (X). Also close the log file in that case.

b) Draw the bar from the bottom, left upwards instead of from the top, left downwards.

Mark what bonuses were attempted in a comment at the top of `DragonLand.java`.