IntelliJ Dojo Lab Book for O'Reilly Publishing

Daniel Hinojosa

Table of Contents

Dojo: Tool Windows

1. Open the **Project** tool window, Close the **Project** Tool
2. Open the **Run** tool tool window, keep it open. Go to the Editor, remember `ESC`!
3. Maximize the editor (`⌘` + `⇧` + `F12` / `CTRL` + `SHIFT` + `F12`)
4. Run `mvn compile` quickly, both by going to the Maven tool window `⌘` + `E` / `CTRL` + `E` / , select the **Maven** tool, and select **compile**
5. Close all tool windows
6. Close all the tabs through either `Shift` + `Shift` or (`⌘` + `⇧` + `A` / `CTRL` + `SHIFT` + `A`)
7. Run `mvn test` using `Run Anything` using `Ctrl` + `Ctrl`

Dojo: Test Driven Development

1. Along with the instructor, let us work on a Test Driven Development project
2. This will be done with the focus of creating new code with all the wonderful amenities that comes with IntelliJ

Dojo: Editing and Refactoring

1. Find the file `Game.java` file using the techniques that we covered
2. Go to the method `main`. Hint: You can quickly search methods by either (⌘ + F12 / CTRL + F12)
3. Highlight only the text `System.console().readLine()` using increase code selection ⌥ + ↑ / CTRL + W)
4. Now use surround block on the selection (⌘ + ⌥ + T / CTRL + ALT + T), and wrap it in a `try` / `catch` / `finally` block that will catch an `IOError` and print the `message` of the exception
5. Go to `initialDeal` method in `Game.java`
6. Highlight the lines the first round of cards into a method call `dealRound` in the method using the blob (officially increase the selection), refactor it by calling *Refactor Method*
7. Make it so that it is called twice
8. Extract another method in the first part of the main method from `AnsiConsole.systemInstall()` to `Hit Enter to Start` and create a method called `displayWelcomeScreen`

Dojo: Navigating

Usages and Implementations

1. Go to `Deck` and then go to the `size` method using the techniques that you've learned
2. Which other methods are using this method? (⌘ + B, ⌥ + F7 / CTRL + B, CTRL + F7)
3. Jump to one of the methods that uses `size`
4. Go to `Game` and then go the constructor of `Game`, from here go to the implementation of `Deck`

Moving

1. Find the file `Game.java` file using the techniques that we covered
2. Make `main` the last method using (⇧ + ⌘ + ↑↓ / CTRL + SHIFT + UP/DOWN)
3. Rearrange the other methods to your liking

Documentation and Info

1. Go to Maven Tool Window and "Download Sources and/or Documentation", you'll need to use the mouse on this one
2. Dismiss the Maven Tool Window
3. Using "Find Symbol" go to `displayFinalGameState`. What is the type of `dealerHand`?
4. Go to `displayHand` using whatever maneuver you like, put your cursor over `joining`, what is the documentation for this method?
5. What is the type of `Collectors.joining( ansi().cursorUp(6).cursorRight(1).toString())`

Dojo: Multicursors and Templates

Multicursors

1. Find the file `DeckTest.java` file using the techniques that we covered
2. Find the word segment `drawCardFrom` using *Find* (⌘ + F / CTRL + F)
3. Highlight `drawCardFrom` and find all the occurences using (^ + ⌘ + G / ALT + SHIFT + J)

Live Templates

1. Create a Live template for yourself
2. Are there some classes that you create? Perhaps a standard way to do log files? maybe `debug` could expand to your favorite logging tool?
3. How about a fixed thread pool? `ExecutorService ec = Executors.newFixedThreadPool(10);`
4. How about using `???` and create `throw new UnsupportedOperationException("Not Implemented");` for always failing initial tests in TDD?
5. If you like to use AssertJ, perhaps for `assertThat` or `assertThatThrownBy`?

Credit to Ted Young for this wonderful project, https://github.com/tedyoung and https://moretestable.com

Last updated 2024-02-19 11:26:03 -0700