3-basic-multi-chan-example - multi-chan-core-async

Benefits

In this exercise you'll gain understanding of the following:

- `alts!` functions and how it enables reading multiple queues

Assumptions

- You have Leiningen installed.
- You have an internet connection (if you don't have this – then we can copy the maven archive across)
- You have worked through the previous exercise

Code to Read

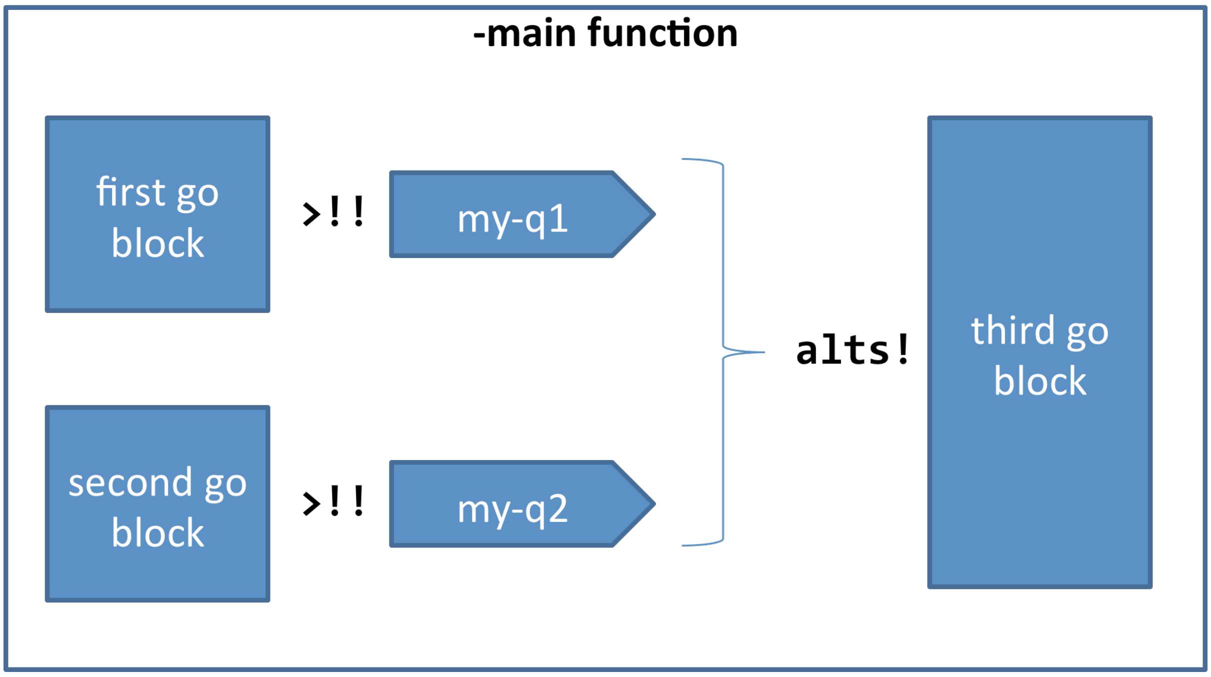
- `lambdajam-2014-core.async-workshop\3-basic-multi-chan-example\multi-chan-core-async\src\multi_chan_core_async\core.clj`

Things to Note In the Code

1. the *three* `go` blocks
2. the *two* queues created with the `chan` function
3. the sequence of queue items in the `my-seq` symbol
4. items being added to the *first* queue in the *first* `go` block
5. items being added to the *second* queue in the *second* `go` block
6. items being removed from *both queues* in the *third* `go` block using the `alts!` function
7. the `timeout` in the *first* and *second* `go` blocks
8. the absence of a `timeout` in the *third* `go` block
9. That the `timeout` in the second `go` block has a *random interval* with `rand-int`
10. That the third `go` block selects a path based on arbitrary queue choice

Code Model

This is a quick way to understand what is going on in the code:



Activities

1. Run the code with `lein run`
2. Change the input to the `rand-int` function for the `timeout` duration in the second `go` block and run it again
3. Remove the `rand-int` function and make the input to the second `timeout` the same as the first and run it again

Questions for Reflection

1. Assume you're writing Java code. How would you write code to wait to get the first result from the first of two queues?