```
class EspressoMachine {
    private data class EspressoShotRequest(
        val deferredEspressoShot: CompletableDeferred<Espresso>,
        val groundBeans: CoffeeBean.GroundBeans
    )

    private val portafilterOne: SendChannel<CoffeeBean.GroundBeans> = actor {...}

    private val portafilterTwo: SendChannel<CoffeeBean.GroundBeans> = actor {...}

    suspend fun pullEspressoShot(groundBeans: CoffeeBean.GroundBeans): Espresso {
        return select {
            portafilterOne.onSend(groundBeans) {
                TODO("sent to portafilter one - wait for result")
            }
            portafilterTwo.onSend(groundBeans) {
                TODO("sent to portafilter two - wait for result")
            }
        }
    }
    ...
}
```

https://jagstalk.page.link/examples

https://jagstalk.page.link/examples

```
class EspressoMachine {
    private data class EspressoShotRequest(
        val deferredEspressoShot: CompletableDeferred<Espresso>,
        val groundBeans: CoffeeBean.GroundBeans
    )

    private val portafilterOne: SendChannel<CoffeeBean.GroundBeans> = actor {...}

    private val portafilterTwo: SendChannel<CoffeeBean.GroundBeans> = actor {...}

    suspend fun pullEspressoShot(groundBeans: CoffeeBean.GroundBeans): Espresso {
        return select {
            portafilterOne.onSend(groundBeans) {
                TODO("sent to portafilter one - wait for result")
            }
            portafilterTwo.onSend(groundBeans) {
                TODO("sent to portafilter two - wait for result")
            }
        }
    }
    ...
}
```

https://jagstalk.page.link/examples

```
class EspressoMachine {
    private data class EspressoShotRequest(
        val deferredEspressoShot: CompletableDeferred<Espresso>,
        val groundBeans: CoffeeBean.GroundBeans
    )

    private val portafilterOne: SendChannel<EspressoShotRequest> = actor {
        consumeEach { request ->
            val espresso = processEspressoShot(request.groundBeans)
            request.deferredEspressoShot.complete(espresso)
        }
    }

    private val portafilterTwo: SendChannel<EspressoShotRequest> = actor {...}

    suspend fun pullEspressoShot(groundBeans: CoffeeBean.GroundBeans): Espresso {
        return select {
            portafilterOne.onSend(groundBeans) {
                TODO("sent to portafilter one - wait for result")
            }
            portafilterTwo.onSend(request) {
                TODO("sent to portafilter two - wait for result")
            }
        }
    }
    ...
}
```