https://jagstalk.page.link/examples

```
class EspressoMachine {
    private val portafilterOne: SendChannel<CoffeeBean.GroundBeans> = actor {
        consumeEach { groundBeans ->
            val espresso = processEspressoShot(groundBeans)
            TODO("deliver espresso shot to sender")
        }
    }

    private val portafilterTwo: SendChannel<CoffeeBean.GroundBeans> = actor {
        consumeEach { groundBeans ->
            val espresso = processEspressoShot(groundBeans)
            TODO("deliver espresso shot to sender")
        }
    }

    suspend fun pullEspressoShot(groundBeans: CoffeeBean.GroundBeans): Espresso {
        TODO("pull espresso shot not implemented")
    }

    suspend fun steamMilk(milk: Milk): Milk.SteamedMilk {
        TODO("steam milk not implemented")
    }

    private suspend fun processEspressoShot(groundBeans: CoffeeBean.GroundBeans): Espresso {...}

    private suspend fun processSteamMilk(milk: Milk): Milk.SteamedMilk {...}
}
```

https://jagstalk.page.link/examples

```
class EspressoMachine {
    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")
            }
        }
    }
    ...
}
```