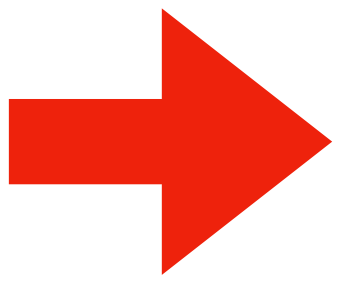
https://jagstalk.page.link/examples

```
fun main() = runBlocking {
    val orders = listOf(...)
    val ordersFlow: Flow<Menu.Cappuccino> = orders.asFlow()
        .produceIn(this)
        .asFlow()

    val espressoMachine = EspressoMachine(this)
    val time = measureTimeMillis {
        flowOf(
            processOrders(ordersFlow, espressoMachine),
            processOrders(ordersFlow, espressoMachine)
        )
            .flattenMerge()
            .collect { cappuccino ->
                log("serve: $cappuccino")
            }
    }
    log("time: $time ms")
    espressoMachine.destroy()
}
```



https://jagstalk.page.link/examples

```
fun main() = runBlocking {
    val orders = listOf(...)
    val ordersFlow: Flow<Menu.Cappuccino> = orders.asFlow()
        .produceIn(this)
        .asFlow()

    val espressoMachine = EspressoMachine(this)
    val time = measureTimeMillis {
        flowOf(
            processOrders(ordersFlow, espressoMachine),
            processOrders(ordersFlow, espressoMachine)
        )
            .flattenMerge()
            .collect { cappuccino ->
                log("serve: $cappuccino")
            }
    }
    log("time: $time ms")
    espressoMachine.destroy()
}
```

https://jagstalk.page.link/examples

```
fun main() = runBlocking {
    val orders = listOf(...)
    val ordersFlow: Flow<Menu.Cappuccino> = orders.asFlow()
        .produceIn(this)
        .asFlow()

    val espressoMachine = EspressoMachine(this)
    val time = measureTimeMillis {
        flowOf(
            processOrders(ordersFlow, espressoMachine),
            processOrders(ordersFlow, espressoMachine)
        )
            .flattenMerge()
            .collect { cappuccino ->
                log("serve: $cappuccino")
            }
    }
    log("time: $time ms")
    espressoMachine.destroy()
}

private fun <T> ReceiveChannel<T>.asFlow(): Flow<T> = flow {
    consumeEach { value ->
        emit(value)
    }
}
```