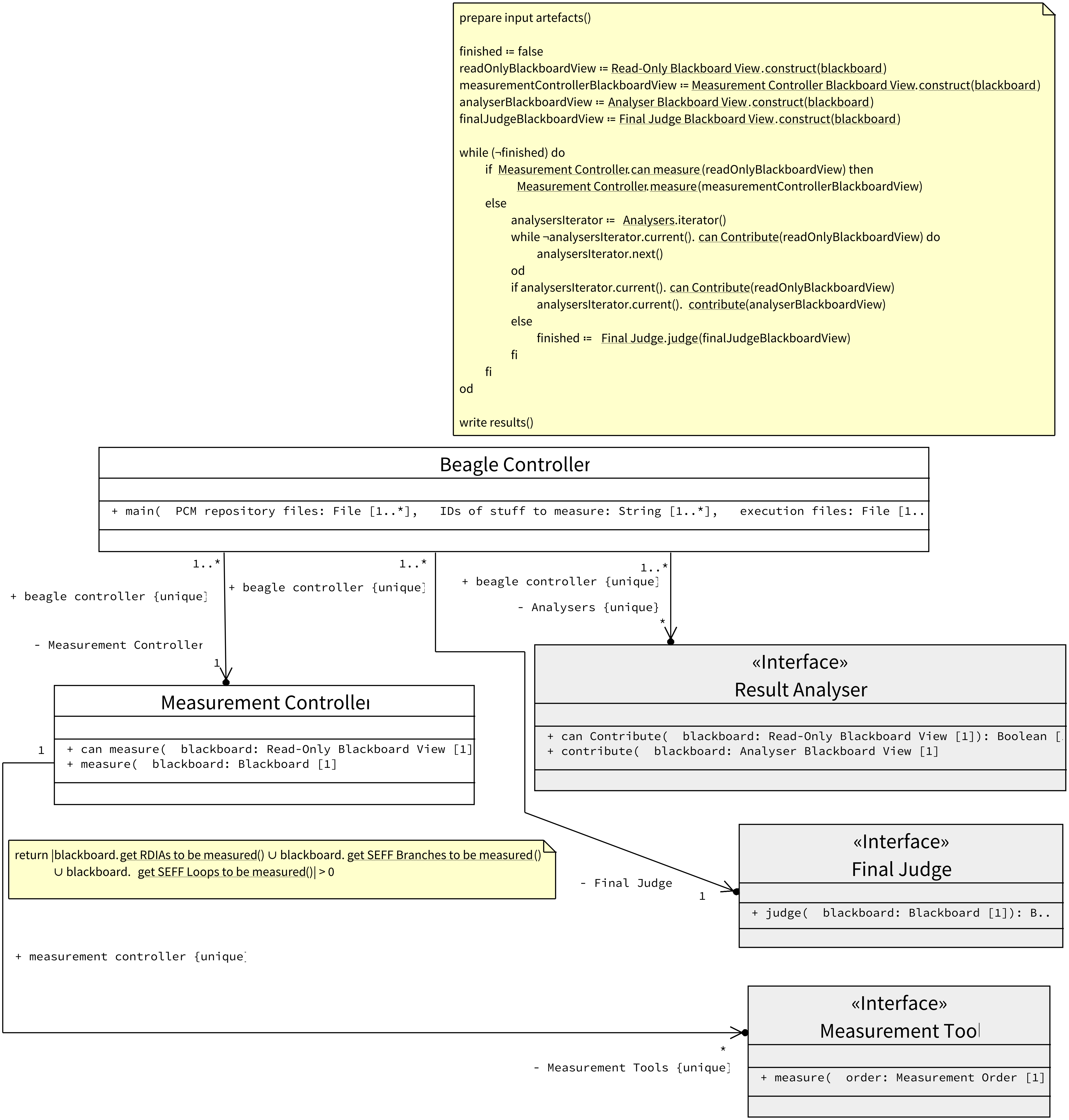

```
prepare input artefacts()

finished ≔ false
readOnlyBlackboardView ≔ Read-Only Blackboard View.construct(blackboard)
measurementControllerBlackboardView ≔ Measurement Controller Blackboard View.construct(blackboard)
analyserBlackboardView ≔ Analyser Blackboard View.construct(blackboard)
finalJudgeBlackboardView ≔ Final Judge Blackboard View.construct(blackboard)

while (¬finished) do
    if Measurement Controller.can measure(readOnlyBlackboardView) then
        Measurement Controller.measure(measurementControllerBlackboardView)
    else
        analysersIterator ≔ Analysers.iterator()
        while ¬analysersIterator.current(). can Contribute(readOnlyBlackboardView) do
            analysersIterator.next()
        od
        if analysersIterator.current(). can Contribute(readOnlyBlackboardView)
            analysersIterator.current(). contribute(analyserBlackboardView)
        else
            finished ≔ Final Judge.judge(finalJudgeBlackboardView)
        fi
    fi
od

write results()
```

```
return |blackboard.get RDIAs to be measured() ∪ blackboard. get SEFF Branches to be measured()
    ∪ blackboard. get SEFF Loops to be measured()| > 0
```

Beagle Controller
+ main(PCM repository files: File [1..*], IDs of stuff to measure: String [1..*], execution files: File [1..
1..*
1..*
1..*
+ beagle controller {unique}
+ beagle controller {unique}
+ beagle controller {unique}
- Measurement Controller
1
- Analysers {unique}
*
«Interface»
Result Analyser
+ can Contribute(blackboard: Read-Only Blackboard View [1]): Boolean [
+ contribute(blackboard: Analyser Blackboard View [1]
Measurement Controller
+ can measure(blackboard: Read-Only Blackboard View [1]
+ measure(blackboard: Blackboard [1]
1
«Interface»
Final Judge
+ judge(blackboard: Blackboard [1]): B..
- Final Judge
1
+ measurement controller {unique}
«Interface»
Measurement Tool
+ measure(order: Measurement Order [1]
*
- Measurement Tools {unique}