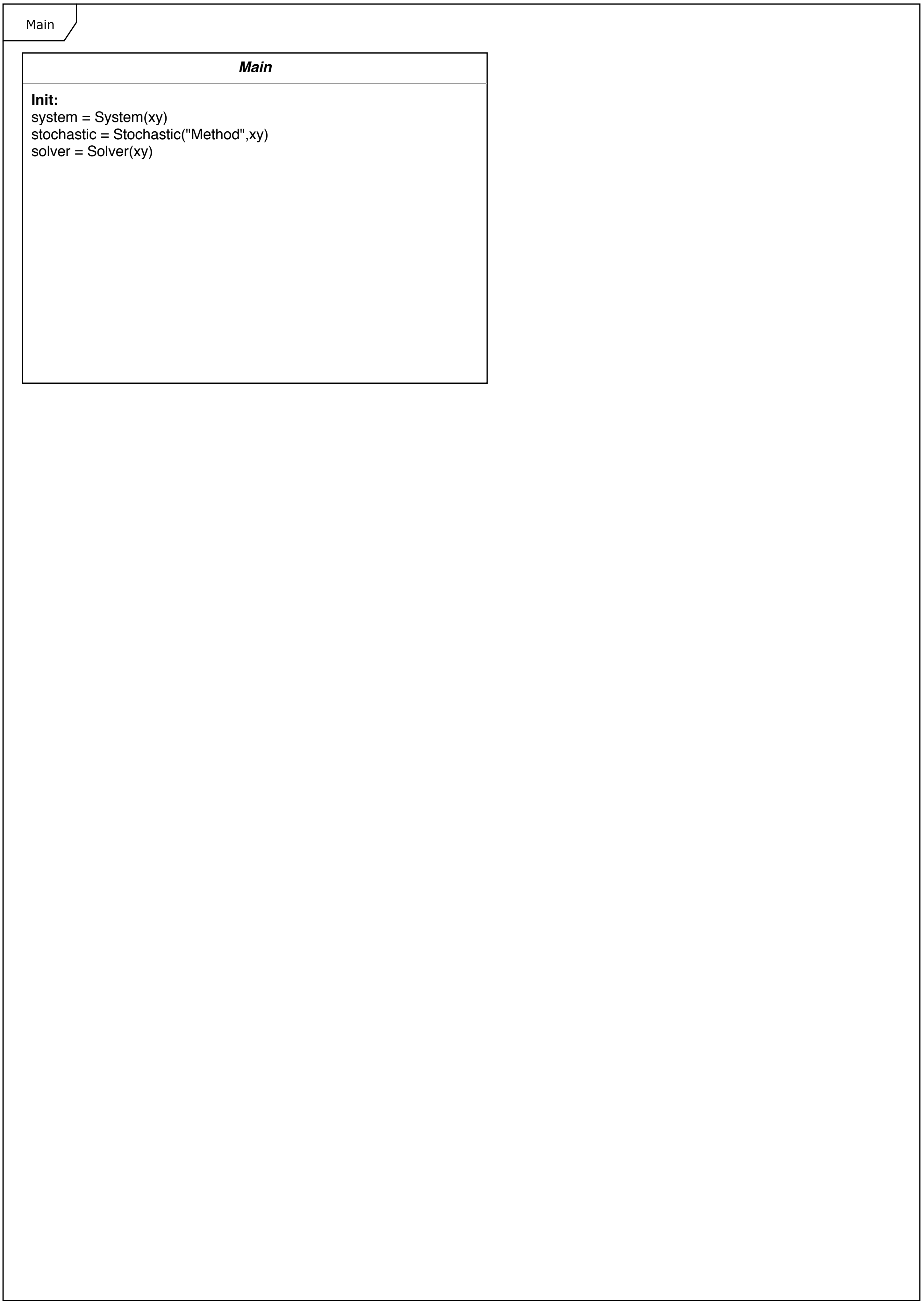
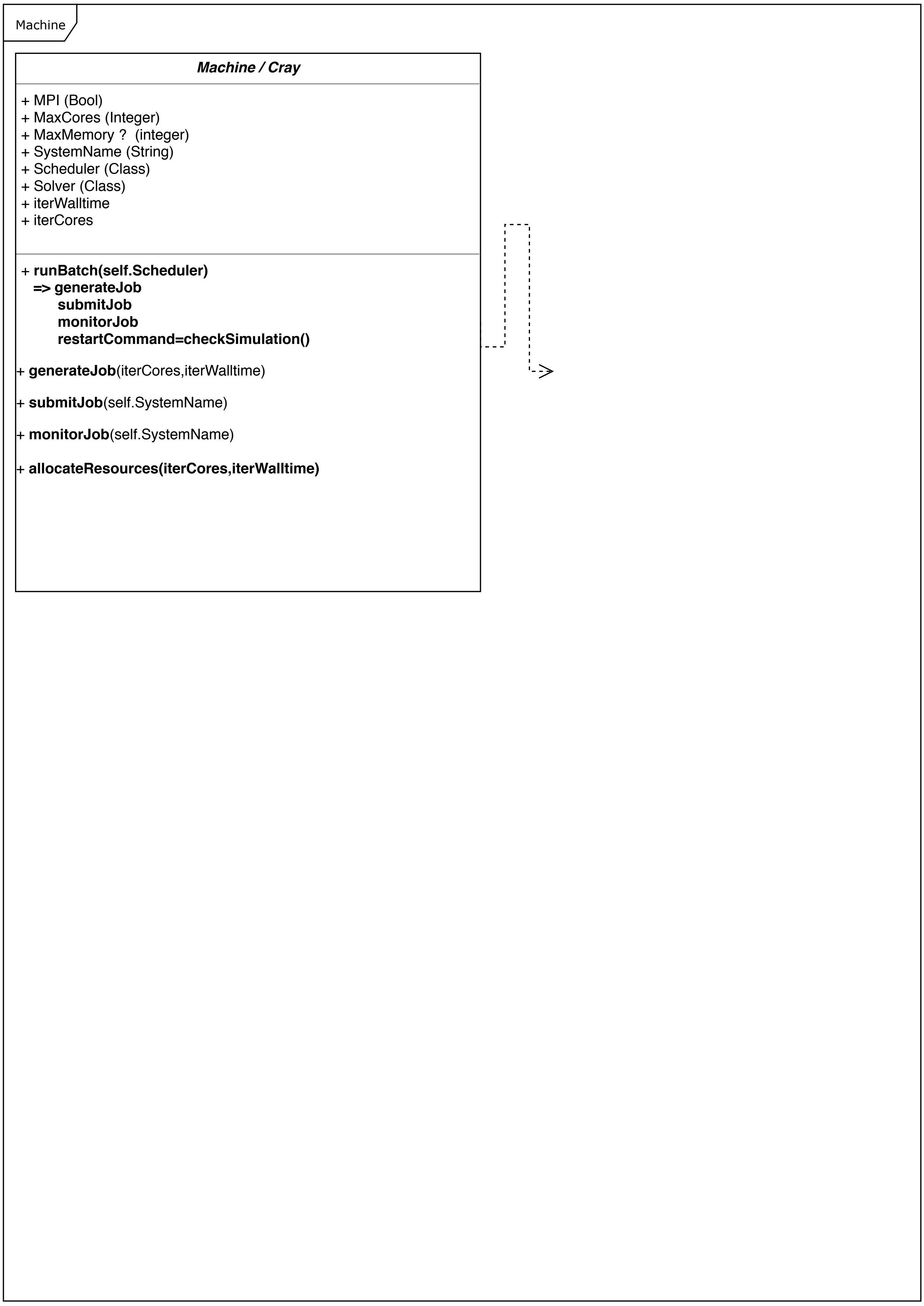
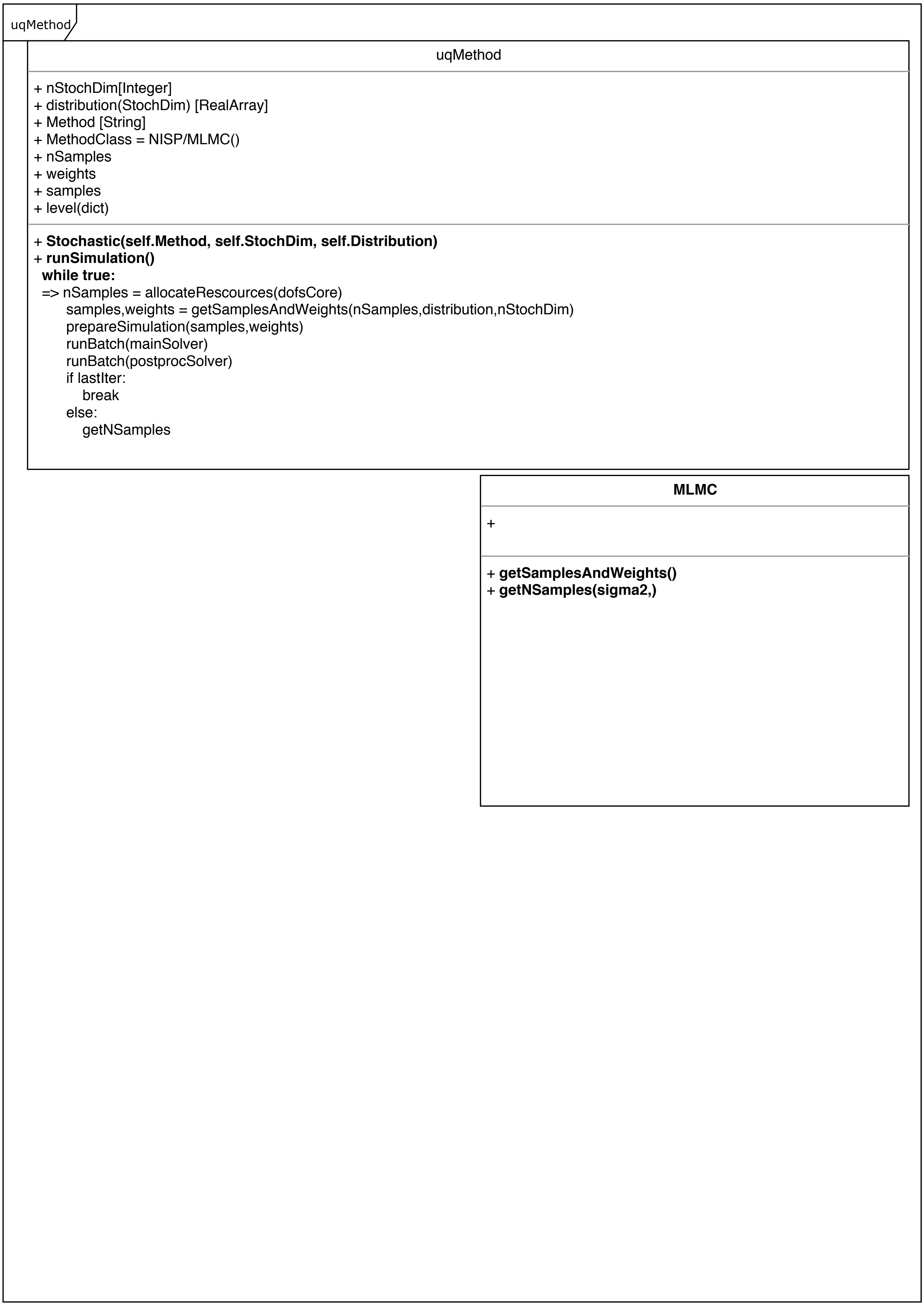


Init:

```
system = System(xy)
stochastic = Stochastic("Method",xy)
solver = Solver(xy)
```


Machine
Machine / Cray
+ MPI (Bool)
+ MaxCores (Integer)
+ MaxMemory ? (integer)
+ SystemName (String)
+ Scheduler (Class)
+ Solver (Class)
+ iterWalltime
+ iterCores
+ runBatch(self.Scheduler)
=> generateJob
submitJob
monitorJob
restartCommand=checkSimulation()
+ generateJob(iterCores,iterWalltime)
+ submitJob(self.SystemName)
+ monitorJob(self.SystemName)
+ allocateResources(iterCores,iterWalltime)



uqMethod

+ nStochDim[Integer]
+ distribution(StochDim) [RealArray]
+ Method [String]
+ MethodClass = NISP/MLMC()
+ nSamples
+ weights
+ samples
+ level(dict)

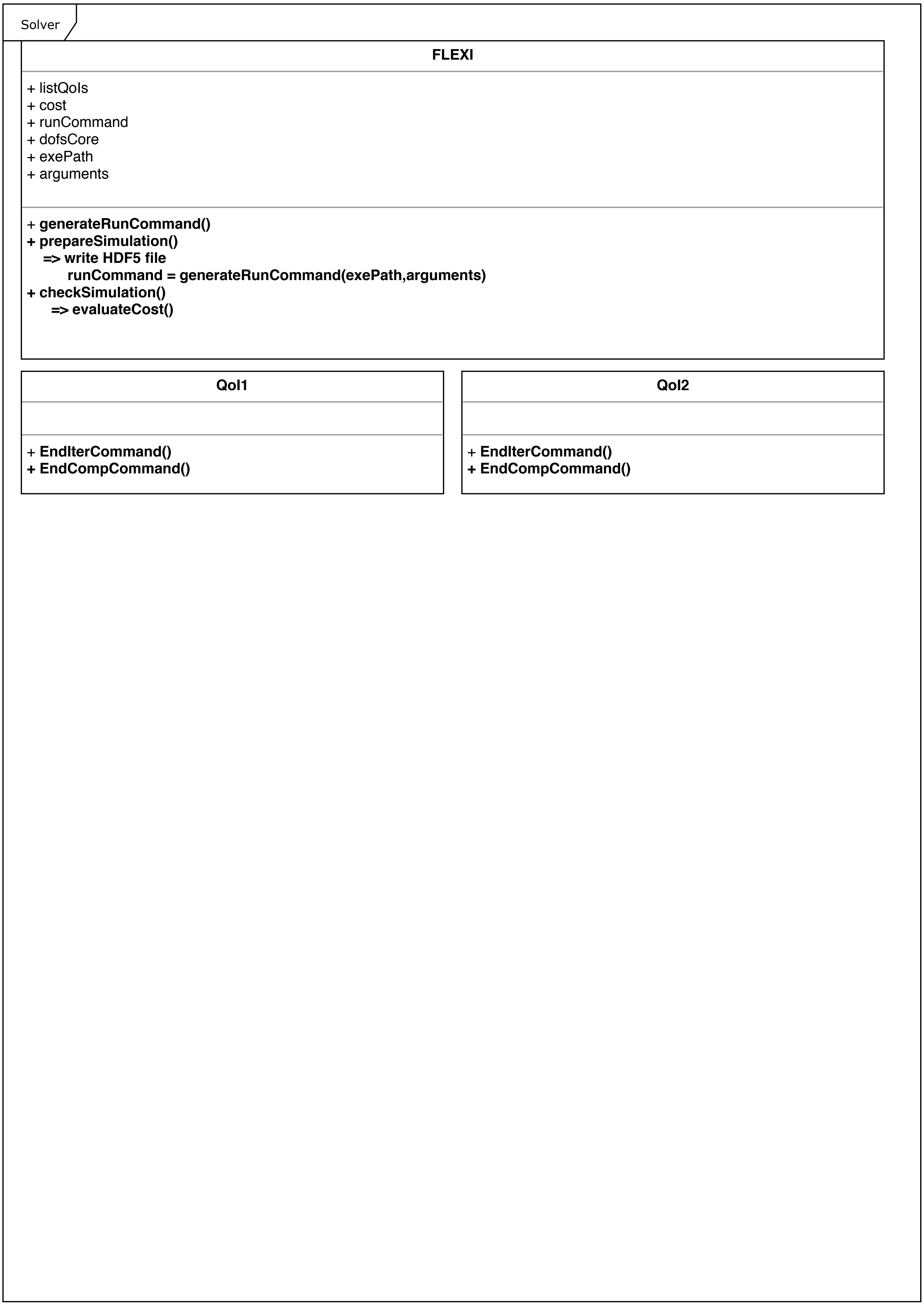
+ Stochastic(self.Method, self.StochDim, self.Distribution)
+ runSimulation()

```
 while true:
 => nSamples = allocateRescources(dofsCore)
      samples,weights = getSamplesAndWeights(nSamples,distribution,nStochDim)
      prepareSimulation(samples,weights)
      runBatch(mainSolver)
      runBatch(postprocSolver)
      if lastIter:
         break
      else:
         getNSamples
```

MLMC

+

+ getSamplesAndWeights()
+ getNSamples(sigma2,)


Solver
FLEXI
+ listQoIs
+ cost
+ runCommand
+ dofsCore
+ exePath
+ arguments
+ generateRunCommand()
+ prepareSimulation()
=> write HDF5 file
runCommand = generateRunCommand(exePath,arguments)
+ checkSimulation()
=> evaluateCost()
QoI1
+ EndIterCommand()
+ EndCompCommand()
QoI2
+ EndIterCommand()
+ EndCompCommand()