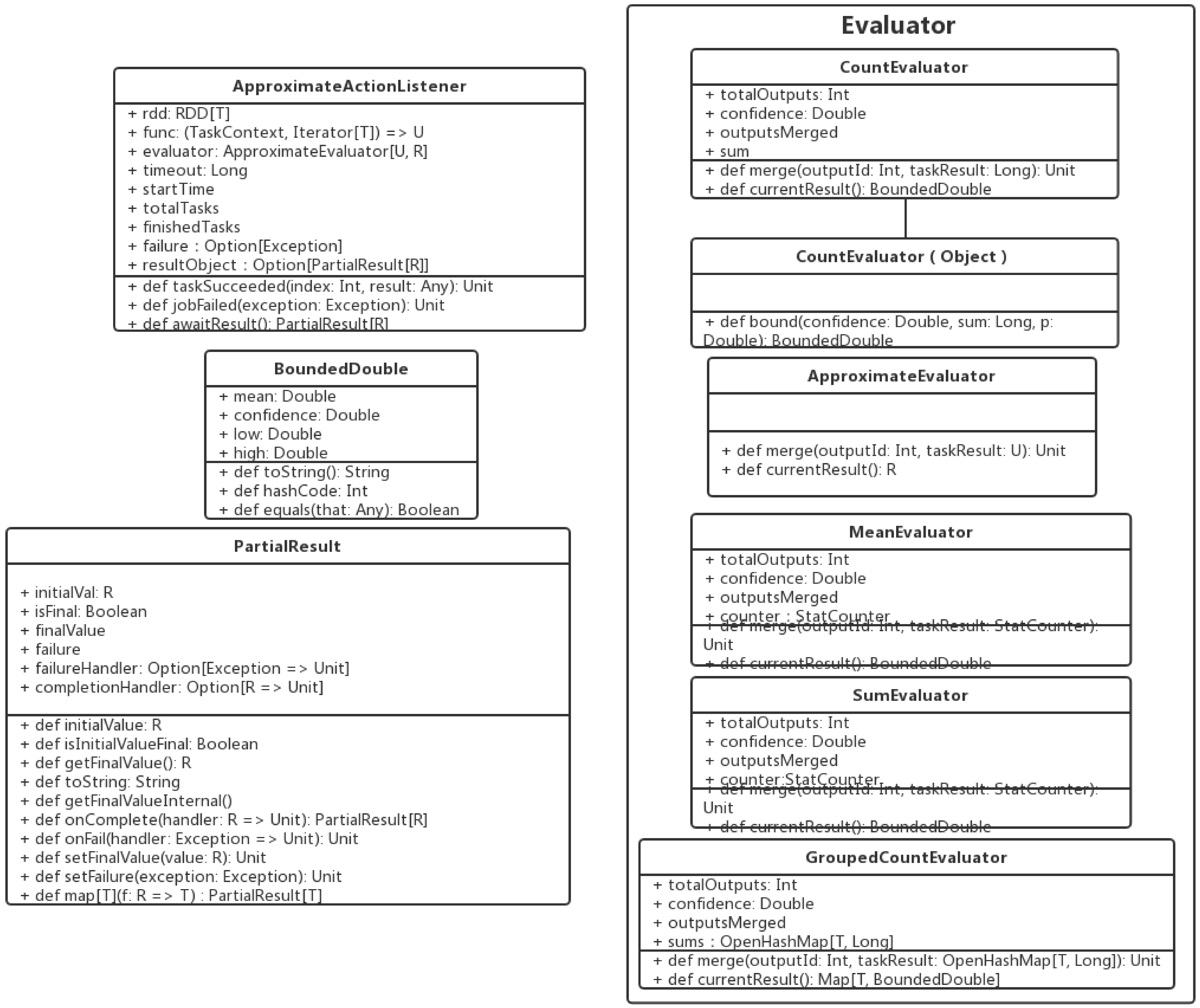
ApproximateActionListener
+ rdd: RDD[T]
+ func: (TaskContext, Iterator[T]) => U
+ evaluator: ApproximateEvaluator[U, R]
+ timeout: Long
+ startTime
+ totalTasks
+ finishedTasks
+ failure : Option[Exception]
+ resultObject : Option[PartialResult[R]]
+ def taskSucceeded(index: Int, result: Any): Unit
+ def jobFailed(exception: Exception): Unit
+ def awaitResult(): PartialResult[R]
BoundedDouble
+ mean: Double
+ confidence: Double
+ low: Double
+ high: Double
+ def toString(): String
+ def hashCode: Int
+ def equals(that: Any): Boolean
PartialResult
+ initialVal: R
+ isFinal: Boolean
+ finalValue
+ failure
+ failureHandler: Option[Exception => Unit]
+ completionHandler: Option[R => Unit]
+ def initialValue: R
+ def isInitialValueFinal: Boolean
+ def getFinalValue(): R
+ def toString: String
+ def getFinalValueInternal()
+ def onComplete(handler: R => Unit): PartialResult[R]
+ def onFail(handler: Exception => Unit): Unit
+ def setFinalValue(value: R): Unit
+ def setFailure(exception: Exception): Unit
+ def map[T](f: R => T) : PartialResult[T]
Evaluator
CountEvaluator
+ totalOutputs: Int
+ confidence: Double
+ outputsMerged
+ sum
+ def merge(outputId: Int, taskResult: Long): Unit
+ def currentResult(): BoundedDouble
CountEvaluator (Object)
+ def bound(confidence: Double, sum: Long, p: Double): BoundedDouble
ApproximateEvaluator
+ def merge(outputId: Int, taskResult: U): Unit
+ def currentResult(): R
MeanEvaluator
+ totalOutputs: Int
+ confidence: Double
+ outputsMerged
+ counter : StatCounter
+ def merge(outputId: Int, taskResult: StatCounter): Unit
+ def currentResult(): BoundedDouble
SumEvaluator
+ totalOutputs: Int
+ confidence: Double
+ outputsMerged
+ counter:StatCounter
+ def merge(outputId: Int, taskResult: StatCounter): Unit
+ def currentResult(): BoundedDouble
GroupedCountEvaluator
+ totalOutputs: Int
+ confidence: Double
+ outputsMerged
+ sums : OpenHashMap[T, Long]
+ def merge(outputId: Int, taskResult: OpenHashMap[T, Long]): Unit
+ def currentResult(): Map[T, BoundedDouble]