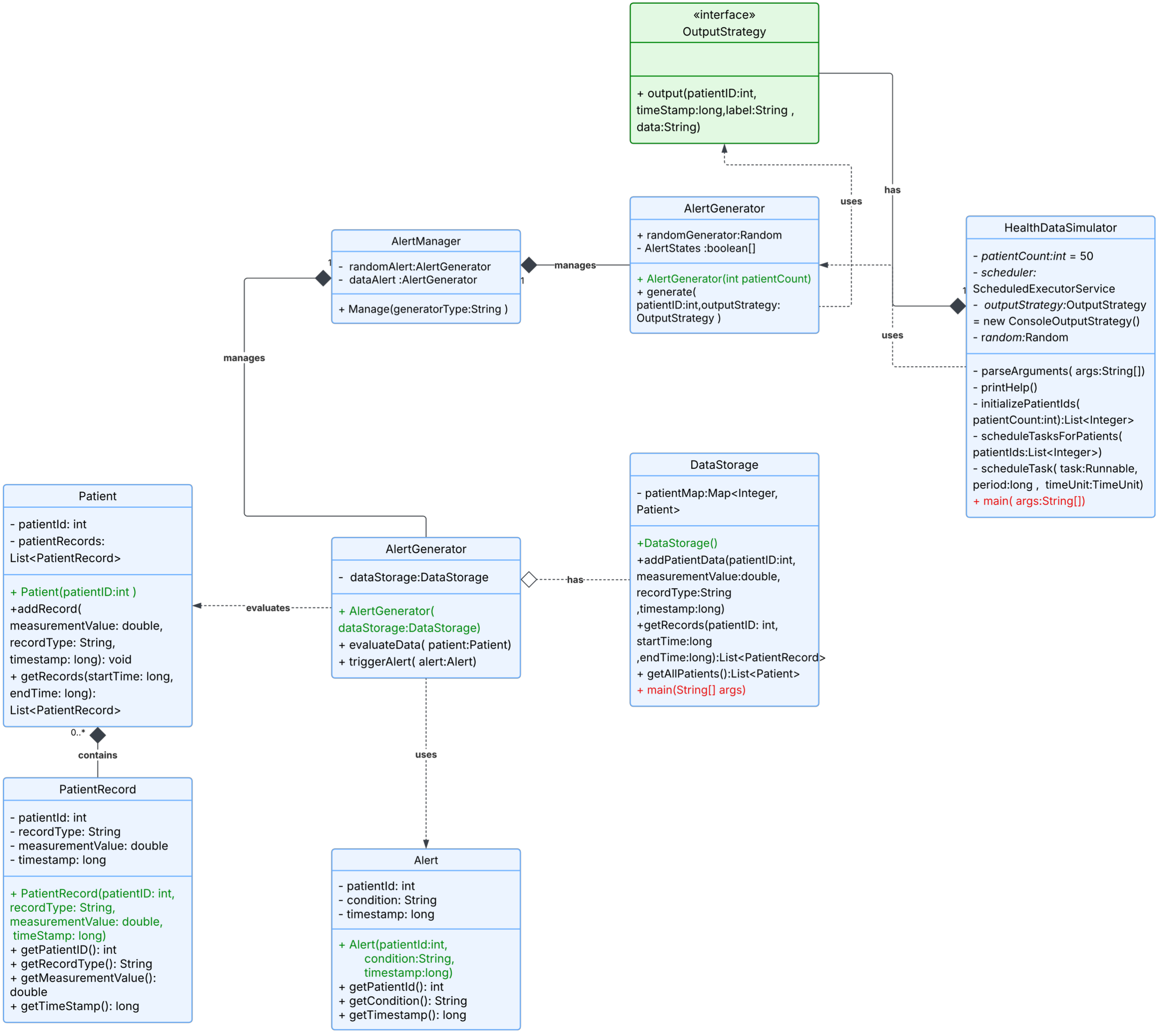
«interface»
OutputStrategy
+ output(patientID:int, timeStamp:long,label:String , data:String)
AlertManager
- randomAlert:AlertGenerator
- dataAlert :AlertGenerator
+ Manage(generatorType:String)
manages
AlertGenerator
+ randomGenerator:Random
- AlertStates :boolean[]
+ AlertGenerator(int patientCount)
+ generate(patientID:int,outputStrategy: OutputStrategy)
uses
has
HealthDataSimulator
- patientCount:int = 50
- scheduler: ScheduledExecutorService
- outputStrategy:OutputStrategy = new ConsoleOutputStrategy()
- random:Random
- parseArguments(args:String[])
- printHelp()
- initializePatientIds(patientCount:int):List<Integer>
- scheduleTasksForPatients(patientIds:List<Integer>)
- scheduleTask(task:Runnable, period:long , timeUnit:TimeUnit)
+ main(args:String[])
manages
Patient
- patientId: int
- patientRecords: List<PatientRecord>
+ Patient(patientID:int)
+addRecord(measurementValue: double, recordType: String, timestamp: long): void
+ getRecords(startTime: long, endTime: long): List<PatientRecord>
0..*
contains
PatientRecord
- patientId: int
- recordType: String
- measurementValue: double
- timestamp: long
+ PatientRecord(patientID: int, recordType: String, measurementValue: double, timeStamp: long)
+ getPatientID(): int
+ getRecordType(): String
+ getMeasurementValue(): double
+ getTimeStamp(): long
evaluates
AlertGenerator
- dataStorage:DataStorage
+ AlertGenerator(dataStorage:DataStorage)
+ evaluateData(patient:Patient)
+ triggerAlert(alert:Alert)
uses
DataStorage
- patientMap:Map<Integer, Patient>
+DataStorage()
+addPatientData(patientID:int, measurementValue:double, recordType:String ,timestamp:long)
+getRecords(patientID: int, startTime:long ,endTime:long):List<PatientRecord>
+ getAllPatients():List<Patient>
+ main(String[] args)
has
Alert
- patientId: int
- condition: String
- timestamp: long
+ Alert(patientId:int, condition:String, timestamp:long)
+ getPatientId(): int
+ getCondition(): String
+ getTimestamp(): long

1. Alert subsystem :

The purpose of the UML diagram is to showcase the architecture and flow of information in a health alert system that has been simulated. It will monitor health data from a patient and issue alerts on abnormal conditions.

Everything is executed from the HealthDataSimulator class which simulates patient activity and periodically schedules tasks to generate or evaluate alerts. It uses one of 2 alertGenerator either randomAlert or dataAlert both of them using the OutputStrategy interface making the input flexible, the first being a randomized alert to ensure easy testing while the second is based on each patient current condition and measurement at each timestamp.

The AlertManager serves as a controller that manages which type of alert generator to use, depending on the input configuration.

Patient health data is modeled using the Patient, PatientRecord, and DataStorage classes. Each Patient holds a list of PatientRecord, representing recorded health metrics like blood pressure or heart rate. DataStorage acts as a repository for all patients and their corresponding records, helping with data retrieval for specific time windows, which is needed for evaluating health.

Alerts are encapsulated in the Alert class to ensure patient data security, which includes patient identification, the condition triggering the alert, and a timestamp.