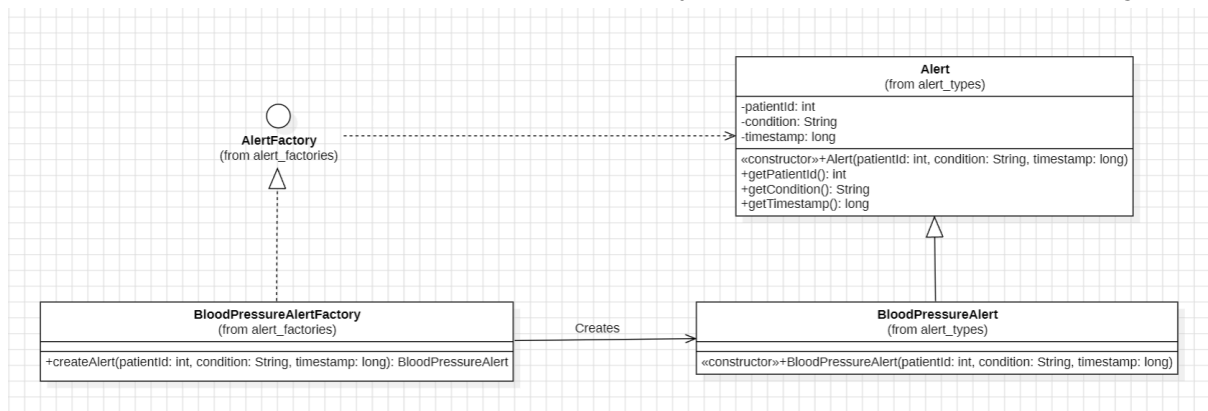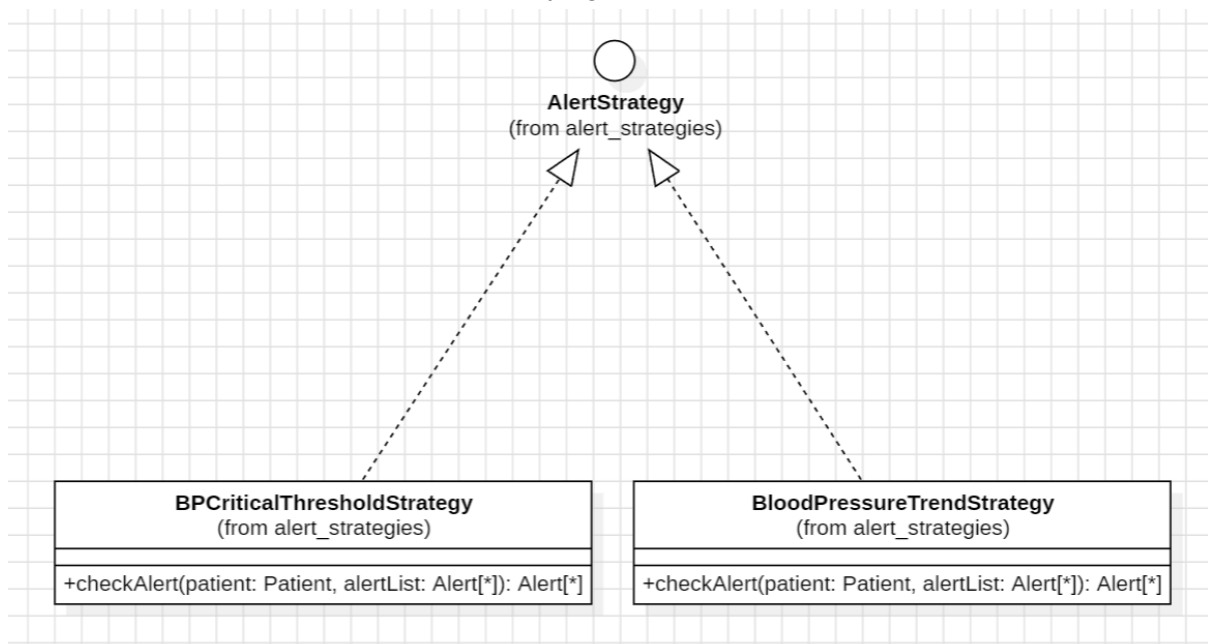# Week 7 tasks

- [X] Implement Factory design pattern
- [X] Implement Strategy pattern
- [ ] Implement Decorator pattern
- [X] Implement Singleton pattern
    - [X] Implement for DataStorage
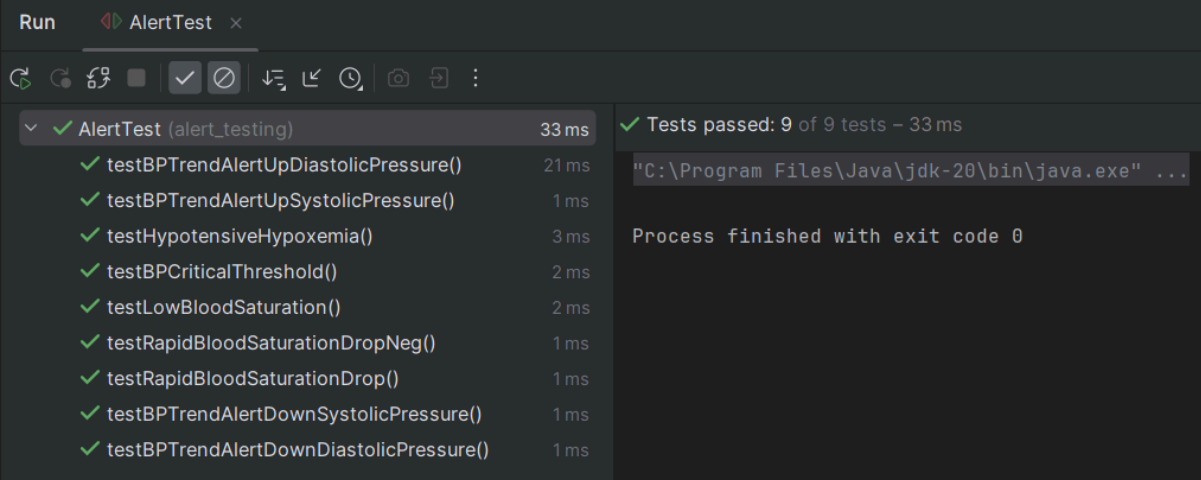   - [X] Implement for HealthDataSimulator

Example application of the factory method design pattern. For blood pressure alerts, you have a specific **'BloodPressureAlert'** class made by the **'BloodPressureAlertFactory"**

Use of the strategy design pattern. The classes below encapsulate the logic for creating their specific alerts. The checkAlert method receives an existing list of alerts and, depending on the inner logic, appends a newly created alert of the appropriate type. In that way, you can create more alert conditions without modifying the **AlertGenerator** class
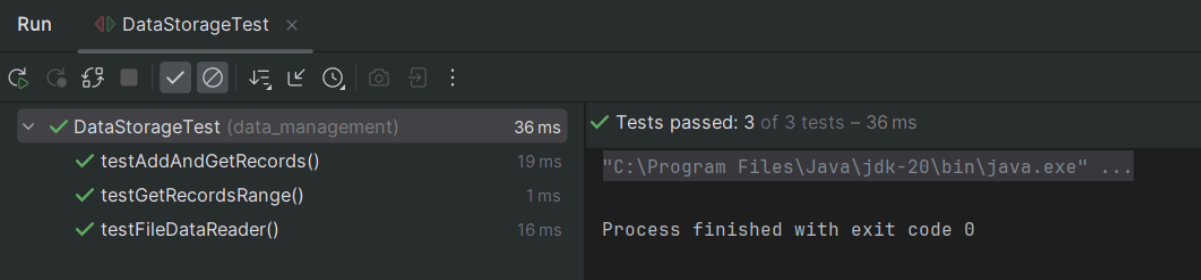
After modifications, tests still pass:

Run	AlertTest ✕

AlertTest (alert_testing)	33 ms	✓ Tests passed: 9 of 9 tests – 33 ms
testBPTrendAlertUpDiastolicPressure()	21 ms	"C:\Program Files\Java\jdk-20\bin\java.exe" ...
testBPTrendAlertUpSystolicPressure()	1 ms	
testHypotensiveHypoxemia()	3 ms	Process finished with exit code 0
testBPCriticalThreshold()	2 ms	
testLowBloodSaturation()	2 ms	
testRapidBloodSaturationDropNeg()	1 ms	
testRapidBloodSaturationDrop()	1 ms	
testBPTrendAlertDownSystolicPressure()	1 ms	
testBPTrendAlertDownDiastolicPressure()	1 ms	

DataStorage test:

Run	DataStorageTest ✕

DataStorageTest (data_management)	36 ms	✓ Tests passed: 3 of 3 tests – 36 ms
testAddAndGetRecords()	19 ms	"C:\Program Files\Java\jdk-20\bin\java.exe" ...
testGetRecordsRange()	1 ms	
testFileDataReader()	16 ms	Process finished with exit code 0