Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.cardio_generator.generators		0%		0%	29	29	120	120	18	18	6	6
com.cardio_generator		0%		0%	31	31	90	90	16	16	2	2
com.cardio_generator.outputs		0%		0%	19	19	62	62	17	17	5	5
com.data_management		49%		44%	12	27	39	77	6	18	1	4
com.design_pattern.Factory		0%		n/a	14	14	21	21	14	14	7	7
com.alerts		92%		88%	11	66	14	176	1	20	0	2
Total	1,543 of 2,492	38%	74 of 162	54%	116	186	346	546	72	103	21	26

Created with JaCo

This is the general overview of the test for WEEK 3 , as you can see there are no test performed for the generators as these were prebuilt .

com.data_management

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
FileDataReader		0%		0%	6	6	22	22	4	4	1	1
DataStorage		42%		50%	4	9	17	31	2	5	0	1
Patient		100%		66%	2	7	0	14	0	4	0	1
PatientRecord		100%		n/a	0	5	0	10	0	5	0	1
Total	156 of 306	49%	10 of 18	44%	12	27	39	77	6	18	1	4

DataStorage

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
main(String[])		0%		0%	3	3	16	16	1	1
getAllPatients()		0%		n/a	1	1	1	1	1	1
addPatientData(int, double, String, long)		100%		100%	0	2	0	6	0	1
getRecords(int, long, long)		100%		100%	0	2	0	5	0	1
DataStorage()		100%		n/a	0	1	0	3	0	1
Total	77 of 133	42%	4 of 8	50%	4	9	17	31	2	5

Created with JaCoCo 0.8.7

As you can see a vast part of the data management hasn't been tested , this is because most of it is code provided by the teacher but also some of these methods especially in data storage are a way to pass the already tested methods such as the main which is why there was no need to test this . Furthermore, there is no way to test this apart from just checking that it runs which it obviously does.

Patient

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
getRecords(long, long)		100%		66%	2	4	0	6	0	1
addRecord(double, String, long)		100%		n/a	0	1	0	3	0	1
Patient(int)		100%		n/a	0	1	0	4	0	1
getRecords()		100%		n/a	0	1	0	1	0	1
Total	0 of 67	100%	2 of 6	66%	2	7	0	14	0	4

Created with JaCoCo 0.8.

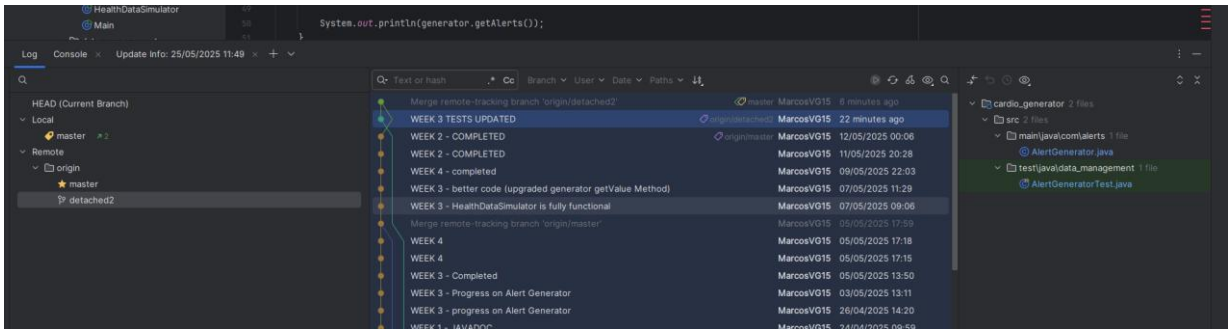
Patient has been indirectly tested for as it contains all the information we want to analyse

AlertGenerator

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
evaluateData(Patient)		0%		n/a	1	1	11	11	1	1
getSpecificValues(String, List)		90%		66%	2	4	1	8	0	1
checkBloodPressure(String, double[], long, String)		97%		81%	2	7	1	19	0	1
bloodSaturationAlerts(List)		99%		91%	1	7	1	26	0	1
lambda$checkBloodPressure$3(double[], int)		94%		50%	1	2	0	1	0	1
bloodPressureDataAlert(List)		100%		90%	2	12	0	40	0	1
hypotensiveHypoxemiaAlert(List)		100%		84%	2	8	0	19	0	1
ECGAlert(List)		100%		100%	0	5	0	22	0	1
ButtonEmergency(List)		100%		100%	0	3	0	8	0	1
triggerAlert(Alert)		100%		n/a	0	1	0	6	0	1
getAverage(double[])		100%		100%	0	2	0	4	0	1
lambda$checkBloodPressure$1(double[], int)		100%		100%	0	2	0	1	0	1
lambda$checkBloodPressure$2(double)		100%		100%	0	3	0	1	0	1
lambda$checkBloodPressure$0(double)		100%		100%	0	3	0	1	0	1
AlertGenerator(DataStorage)		100%		n/a	0	1	0	4	0	1
getAlerts()		100%		n/a	0	1	0	1	0	1
Total	61 of 839	92%	10 of 90	88%	11	62	14	168	1	16

Created with JaCoCo

This is the overview of all the tests. As you can see I haven't tested for the evaluate Data , As this method contains all the other methods, and similarly to the Data Storage , it would be redundant to test this method too.



I have had some complications as my initial tests as they didn't use assert function to analyse the code, so I created another branch that does contain this, which is why you can see a detached branch. I believe that leaving as another branch is the best way to preserve this change as you can access the branch and see my code easily and fully functionally, and secondly, I don't know where the best location would be to branch the code without having to refactor a lot of the code.