Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.cardio_generator.generators		0%		0%	44	44	150	150	33	33	11	11
com.cardio_generator		0%		0%	45	45	132	132	22	22	2	2
com.cardio_generator.outputs		0%		0%	19	19	62	62	17	17	5	5
com.data_management		51%		45%	13	30	40	83	5	19	1	4
com.design_pattern.Decorators		45%		26%	11	18	16	35	4	10	0	3
com.alerts		92%		50%	5	14	2	34	1	10	0	2
com.design_pattern.Factory		94%		n/a	1	12	2	18	1	12	0	6
com.design_pattern.strategy		100%		95%	3	61	0	168	0	24	0	6
Total	1,933 of 3,079	37%	100 of 187	46%	141	243	404	682	83	147	19	39

This is the general overview of the Junit test. As you can see there a few folders that where I haven't provided Junit tests, this is because most of these are either generators, or output strategies that will be modified in week 5 . The rest of the pictures will cover each of the folders for which I have provided Junit tests.

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		48%		50%	5	12	18	37	1	6	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 323	51%	12 of 22	45%	13	30	40	83	5	19	1	4

Above is the overview of the data Management folder , where I test the essential running methods of Data Storage , as the other methods are helpers and extractors which don't need to be tested for , for obvious reasons. Furthermore I don't test FileDataReader due to the fact that I am aware and have tested it previously in many occasions and know it works.

com.design_pattern.Decorators

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
RepeatedAlertDecoratorDec		40%		16%	8	10	13	23	2	4	0	1
AlertDecoratorDec		55%		n/a	2	4	2	6	2	4	0	1
PriorityAlertDecoratorDec		82%		66%	1	4	1	6	0	2	0	1
Total	97 of 179	45%	11 of 15	26%	11	18	16	35	4	10	0	3

Above is the overview of the decorators , where I only tested the method that will return or serve a direct purpose to the final output.

com.alerts

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
AlertGenerator		91%		50%	5	10	2	26	1	6	0	1
Alert		100%		n/a	0	4	0	8	0	4	0	1
Total	12 of 155	92%	4 of 8	50%	5	14	2	34	1	10	0	2

Created

com.design_pattern.Factory

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
AlertFactory		42%		n/a	1	2	2	3	1	2	0	1
BloodPressureAlertFactory		100%		n/a	0	2	0	3	0	2	0	1
HypotensiveHypoxemiaAlertFactory		100%		n/a	0	2	0	3	0	2	0	1
EmergencyButtonFactory		100%		n/a	0	2	0	3	0	2	0	1
HeartRateAlertFactory		100%		n/a	0	2	0	3	0	2	0	1
OxygenSaturationAlertFactory		100%		n/a	0	2	0	3	0	2	0	1
Total	4 of 67	94%	0 of 0	n/a	1	12	2	18	1	12	0	6

Created with JaCoCo 0.8.7

I test indirectly all of the factories through testing the strategies , and as you can see below all strategies are tested.

com.design_pattern.strategy

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
BloodPressureStrategy		100%		96%	1	20	0	49	0	5	0	1
OxygenSaturationStrategy		100%		91%	1	11	0	42	0	5	0	1
HypotensiveHypoxemiaStrategy		100%		93%	1	10	0	26	0	2	0	1
HeartRateStrategy		100%		100%	0	10	0	28	0	4	0	1
EmergencyButtonStrategy		100%		100%	0	5	0	13	0	3	0	1
Item		100%		n/a	0	5	0	10	0	5	0	1
Total	0 of 691	100%	3 of 72	95%	3	61	0	168	0	24	0	6

Created with JaCoCo 0.8.7.202105040129