






cardio_generator

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed	Methods	Missed	Classes		
 com.cardio_generator.generators	<div><div></div></div> 90%	<div><div></div></div> 90%	81%	2	24	17	103	0	16	0	5	
 com.cardio_generator.outputs	<div><div></div></div> 92%	<div><div></div></div> 92%	100%	1	17	10	59	1	16	0	5	
 com.data_management	<div><div></div></div> 94%	<div><div></div></div> 94%	80%	5	34	8	88	1	24	0	5	
 com.cardio_generator	<div><div></div></div> 97%	<div><div></div></div> 97%	87%	4	26	2	81	1	13	0	1	
 com.alerts	<div><div></div></div> 100%	<div><div></div></div> 100%	92%	6	54	0	88	0	13	0	2	
Total	98 of 2,006	95%	16 of 144	88%	18	155	37	419	3	82	0	18

The total missed instructions is 98 of 2006(5%), the missed branches is 16 of 144(12%).

To be more specific, missed is catch (Exception e).

For example:

```

10. public class BloodLevelsDataGenerator implements PatientDataGenerator {
11.     private static final Random random = new Random();
12.     private final double[] baselineCholesterol;
13.     private final double[] baselineWhiteCells;
14.     private final double[] baselineRedCells;
15.
16.     public BloodLevelsDataGenerator(int patientCount) {
17.         // Initialize arrays to store baseline values for each patient
18.         baselineCholesterol = new double[patientCount + 1];
19.         baselineWhiteCells = new double[patientCount + 1];
20.         baselineRedCells = new double[patientCount + 1];
21.
22.         // Generate baseline values for each patient
23.         for (int i = 1; i <= patientCount; i++) {
24.             baselineCholesterol[i] = 150 + random.nextDouble() * 50; // Initial random baseline
25.             baselineWhiteCells[i] = 4 + random.nextDouble() * 6; // Initial random baseline
26.             baselineRedCells[i] = 4.5 + random.nextDouble() * 1.5; // Initial random baseline
27.         }
28.     }
29.
30.     @Override
31.     public void generate(int patientId, OutputStrategy outputStrategy) {
32.         try {
33.             // Generate values around the baseline for realism
34.             double cholesterol = baselineCholesterol[patientId] + (random.nextDouble() - 0.5) * 10; // Small variation
35.             double whiteCells = baselineWhiteCells[patientId] + (random.nextDouble() - 0.5) * 1; // Small variation
36.             double redCells = baselineRedCells[patientId] + (random.nextDouble() - 0.5) * 0.2; // Small variation
37.
38.             // Output the generated values
39.             outputStrategy.output(patientId, System.currentTimeMillis(), "Cholesterol", Double.toString(cholesterol));
40.             outputStrategy.output(patientId, System.currentTimeMillis(), "WhiteBloodCells",
41.                 Double.toString(whiteCells));
42.             outputStrategy.output(patientId, System.currentTimeMillis(), "RedBloodCells", Double.toString(redCells));
43.         } catch (Exception e) {
44.             System.err.println("An error occurred while generating blood levels data for patient " + patientId);
45.             e.printStackTrace(); // This will print the stack trace to help identify where the error occurred.
46.         }
47.     }
48. }

```

```

25. }
26.
27. @Override
28. public void generate(int patientId, OutputStrategy outputStrategy) {
29.     // TODO Check how realistic this data is and make it more realistic if necessary
30.     try {
31.         double ecgValue = simulateEcgWaveform(patientId, lastEcgValues[patientId]);
32.         outputStrategy.output(patientId, System.currentTimeMillis(), "ECG", Double.toString(ecgValue));
33.         lastEcgValues[patientId] = ecgValue;
34.     } catch (Exception e) {
35.         System.err.println("An error occurred while generating ECG data for patient " + patientId);
36.         e.printStackTrace(); // This will print the stack trace to help identify where the error occurred.
37.     }
38. }
39.

```