Chapter II

In many cases it goes that familiarity is the enemy of fear. So it has been with this project. At first, it seemed overwhelming, daunting, and in some ways almost infeasible. Sure, we've written code before, that's what we do; however, this has always resulted in a product of a few lines, maybe a few hundred—never in something that had not a few hundred lines, but a few hundred files. Not only that, we were completely unfamiliar with their code and conventions. We'd never done anything like this before. These all began to combine to form in the mind the idea: we can't do this. Or at least that it would be quite consuming.

However, as time progressed, it generally began to become more manageable and less intimidating. As we interacted with the various parts of the software and the software engineering process, they became less challenging and more natural. There were things we needed to learn about the system, but that no longer fostered so much apprehension in us as it did before.

This trend that has happened overall with the project happened individually with these test cases. What follows are five test cases written for a part of the OpenMRS system and the source code is found at openmrs-core/api/src/main/java/org/openmrs

Test ID:	1
Requirement:	The union of two intersection Cohorts
Component:	Cohort
Method:	public static Cohort union(Cohort a, Cohort b)
Inputs:	[1,2,3,4,5] [3,4,5]
Expected:	[1,2,3,4,5]

Test ID:	2
Requirement:	The union of a non-empty Cohort and an empty Cohort
Component:	Cohort
Method:	public static Cohort union(Cohort a, Cohort b)
Inputs:	[1,2,3,4,5] []
Expected:	[1,2,3,4,5]

Test ID:	3
Requirement:	The union of two empty Cohorts
Component:	Cohort
Method:	public static Cohort union(Cohort a, Cohort b)
Inputs:	[] []
Expected:	[]

Test ID:	4
Requirement:	The union of two non-intersecting Cohorts
Component:	Cohort

Method: public static Cohort union(Cohort a, Cohort b)
Inputs: [1,2,3] [4,5]
Expected: [1,2,3,4,5]

TestID: 5
Requirement: The subtraction of two interesting Cohorts
Method: public static Cohort subtract(Cohort a, Cohort b)
Inputs: [1,2,3,4,5] [3,4,5]
Expected: [1,2]

Testing Process	**Test the requirements of the system using an automated test system which runs various test cases which we have written. We will begin with the Cohort class.**
Requirements	The Cohort class is means of grouping patients. Various set operations can be performed on them. Therefore, the union method should join the two Cohorts with no repeats of members.
Tested Items	The Cohort class will be tested.
Testing Schedule	The tests will be performed incrementally from now until the end of the semester.
Test Recording Procedures	The results of the tests will be compared against pre-written oracles and stored in a text file.
Hardware/Software Requirements	Ubuntu OS.
Constraints	Not every aspect of the system will be tested due to limited time.