My testing approach was structured to validate the core requirements for Contract Service and Task Service.   
For Example:  
@Test

public void testAddTask() {

Task task = new Task("123", "Sample Task", "This is a test task.");

taskService.addTask(task);

assertEquals("Sample Task", taskService.getTask("123").getName());

}  
With TaskServiceTest.java, this ensures the test cases verified that task could only be added if they had a unique ID. The tests above only valid tests are stored and meet the constraints listed in the requirement. In this section of the Junit test, it becomes complex. Test Coverage refers to how much of my code is used by JUnit. The effectiveness of my Junit test can be measured by coverage percentage. My cases were tested and somehow feel it was missing something because the code barely ran. In previous reading materials, it’s easier to ensure that I properly implement validation check and exception handling in both Task.java and TaskService.java.  
For example:  
@Test

public void testInvalidTaskId() {

assertThrows(IllegalArgumentException.class, () -> new Task(null, "Task", "Description"));

}  
The efficiency of my code is reducing unnecessary operations and making my code run. My code contains validation issues and bottlenecks. Frustrating process if I must say so I have perform my Agile techniques and run my code each 25% of the way to cover my errors.