**Summary of (J)Unit Testing (and meeting software requirements)**

1. **Contacts**
   1. In module three for my testing I incorporated Junit testing and imported *static org.junit.Assert*. This import gives us a set of assertion methods useful for testing. In the *ContactServiceTest* class I use methods such as *testMethodAddPass()* and *testMethodAddFail()* and the *assertEquals* method to compare and test the add methods with valid and invalid details provided. These tests make sure that proper data (type and length of data) is provided (Contact ID, First Name, Last Name, Address, and Phone number). The same tests and methods were completed with the delete and update methods as well in this class for testing.
2. **Tasks**
   1. In module 4 I incorporated Junit testing as well. In the *TaskServiceTest* Class I import *org.junit.Test*. I used the test annontation before the methods in this class to treat them as a test case testing for invalid data with methods like *invalidID()* and *invalidName().*
3. **Appointments** 
   1. In module 5 I incorporated junit test well in the appointment classes. I used the Junit import *import static org.junit.jupiter.api.Assertions.\*;* I also used import *org.junit.jupiter.api.BeforeEach*; and import *org.junit.jupiter.api.Test*;. This allowed me to use the annonation @before each, which is used to signal that the annotated method should be executed before each *@Test* method in the current class. I also used the annotation *@SuppressWarnings("deprecation")* because deprecation tells the compiler to ignore when we're using a deprecated method or type. I then used method commands such as *assertNotNull, assertEquals*, and *assertNotEquals* to compare and test appointment Ids, dates, and descriptions. I tested for adding and also deleting appointments.
4. **To meet all software requirements** the Contact, Task, and Appointment classes all were structured in a way that allowed them to do the things provided in the software requirements. In most cases a method was made to be able to do a single one of those tasks. Constants were defined to keep variables at a proper length as outlined in requirements. For example, Methods were made to do the following in respective classes to meet requirements (these are some not all):
   1. **Contact**
      1. Main Variables to fulfill requirements: contactID, firstName, lastName, numberAddress, and phoneNumber.
      2. Methods used: Update Methods, Add/remove Contacts, newUniqueId, alreadyPresent, and a contacts List
   2. **Task**
      1. Main variables used: id, name, description
      2. Methods used: setters/getters, update methods, newUniqueId, newTask, DeleteTask, Validate methods, and a Task List.
   3. **Appointment**
      1. Main variables used: appointmentId, ApointmentDate, and description.
      2. Methods used: setters/getters, update methods, newUniqueId, newAppointment, deleteAppointment, and an Appointment List.

**Additional Junit Testing Information**

JUnit developed the idea of first testing (design the test case) and then coding that assure setting up of the test data and defining the expected output and then coding (WideSkills). This procedure increases productivity and stability of program code and reduce the time for debugging (WideSkills). Junit and unit testing can catch bugs better than programmers (WideSkills). Sometimes a bug is introduced when new features are added or when existing code is changed (WideSkills)**.**JUnit tests over the old code can make us sure that they will still pass and avoid costly regressions (WideSkills). Due to its simplicity JUnit should be first option when testing the java applications (WideSkills). JUnit can also be used separately or integrated with build tools like Maven and Ant and third party extensions (WideSkills). **JUnit** is widely adopted by organizations around the world for performing unit testing in Java programming language SITE.

**Software Testing Techniques used in this Project**

I talked previously about testing techniques used in each module. As a whole I will briefly go over some of those things I used throughout this project in a list format:

* Coming up with test cases
* @test annotation with test methods and other annotations
* Assertions
  + Static methods in *org.junit.jupiter.api.Assertions* class
  + assertEquals/notEquals
  + assertNotNull

**Software Testing Techniques not used for this Project**

I used a lot of assert methods in my work, but I did not use all the assert methods that you can use. Junit assert methods are made to do testing, where only failed assertions are recorded. I used mainly assertion methods such as *Assertequals* and *AssertnotEquals* but I did not use methods such as *assertTrue, assertFalse, assertSame, assertNotSame*, or *assertArayEquals*. I also used Junit annotations in my work. One thing available with Junit that I did not use is Junit Runner. With Junit Runner you can implement abstract methods of the Runner class using imports like *org.junit.runner.JUnitCore, org.junit.runner.Result*, or *org.junit.runner.notification.Failure.*

**Mindset (Caution, Bias, and discipline.**

While working on this project, I had to first develop the code for the project. That main code consisted of coming up with structure of classes, variables, getters/setters, Lists, and methods to make the code do what was required. But also, I had to include in test methods, and have a mindset of testing. This testing mindset while coding could catch things I had put in the code and point it out to me in an easier way to see. JUnit is used to test an actual class. This procedure increases productivity and stability of program code and reduce the time for debugging. It seems like while coding the project test methods, I was adding in a lot of code, and if everything goes well and no errors in code then it didn’t seem like it mattered. But the whole point of doing it was to create a more secure, quality, error-free version of the project, so that it will achieve what I am trying to get it to do without error and in the most efficient way. Junit helps make sure that that will happen. It may take discipline in the future to add this extra code in, but it is worth the extra effort, to enhance code quality.

Below is a list of ways we can help eliminate bias in programming (list resource: (*How to reduce bias in AI with a focus on training data* 2021)

1. **Think about all of your end-users**  
   Understand that your end-users won’t simply be like you or your team. Avoid bias by learning to anticipate how people who aren’t like you will interact with your technology and what problems might arise in their doing so.
2. **Annotate with diversity**  
   The more spread out the pool of human annotators, the more diverse your viewpoints. That can help reduce bias both at the launch.
3. **Test and deploy with feedback in mind**  
   Models are rarely static for their entire lifetime. Opening up a discussion and forum for feedback will continue to ensure your model is maintaining optimal performance levels for everyone.
4. **Have a concrete plan to improve your model with that feedback**  
   You’ll want to continually review your model using not just customer feedback, but also independent people auditing for changes, edge cases, instances of bias you might’ve missed, and more. Make sure you get feedback from your model and give it feedback of your own to improve its performance.
5. **Define and narrow the business problem you’re solving**  
   Trying to solve for too many scenarios often means you’ll need a ton of labels across an unmanageable number of classes. Narrowly defining a problem, to start, will help you make sure your model is performing well for the exact reason you’ve built it.
6. **Structure data gathering that allows for different opinions**There are often multiple valid opinions or labels for a single data point. Gathering those opinions and accounting for legitimate, often subjective, disagreements will make your model more flexible

**RESOURCES**

WideSkills. (n.d.). Advantages of using Junit. Retrieved February 21, 2021, from https://www.wideskills.com/junit/advantages-using-junit#:~:text=JUnit%20developed%20the%20idea%20of,reduce%20the%20time%20for%20debugging.

How to reduce bias in AI with a focus on training data. (2021, February 17). Retrieved February 21, 2021, from https://appen.com/blog/how-to-reduce-bias-in-ai/