**Unit 2 Building and testing sustainable software.**

**Overview:**

My current workplace deals with car finance so there is a need in my team to be able to filter the applications based on a range of parameters. My methods will be a very simplified version of these filters. They will filter pre-determined ('hardcoded') applications that are passed into it rather than retrieving the applications externally.

**Age filter user story:**

As a user, I want to be able to filter applications for car finance based on the age of the applicant so that I can prioritize applications from eligible applicants only.

**Acceptance criteria:**

1. There should be a method that takes an application as input, where the application is represented by a data structure containing applicant information, including age.
2. The method should filter out the application if it does not meet the minimum and maximum age requirements for car finance.
3. The minimum and maximum age requirements should be configurable so they can be adjusted based on business needs.
4. The output of the method should be a Boolean. True if the applicant meets the age requirements and false if not.
5. The method should also be unit-tested to ensure its correctness in filtering based on age.

**Credit banding filter user story:**

As a user, I want to be able to filter applications based on the credit banding of the applicant to ensure that only applicants with credit bandings between C and E (including C and E) can proceed with their application.

**Acceptance criteria:**

1. The requirements should be implemented as a method that takes a data structure representing the application as an input. It should contain the credit banding of the applicant.
2. The method should filter out the application if it does not meet the credit banding requirements for car finance at Zuto.
3. The invalid credit bands should be configurable so they can be adjusted based on business needs.
4. The output of the method should be a Boolean. True if the applicant meets the credit banding requirements and false if not.
5. The method should also be unit-tested to ensure its correctness in filtering based on credit banding.

**Occupation filter user story:**

As a user, I aim to account for the higher wear and tear associated with vehicles owned by customers in professions involving frequent driving. To achieve this, I want to be able to filter car finance applications based on the applicant's occupation. This will enable me to exclude applicants whose profession is 'driver'.

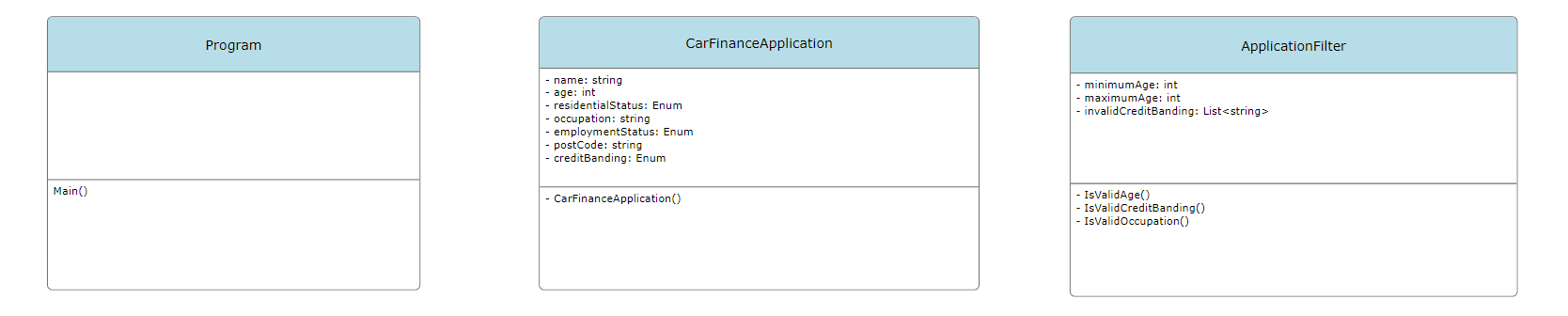
**Acceptance criteria:**

1. The requirements should be implemented as a method that takes a data structure representing the application as an input. It should contain the occupation of the applicant.
2. The method should filter out the application if their occupation contains the word driver.
3. The output of the method should be a Boolean. True if the occupation of the applicant does not contain the word driver and false if it does.
4. The method should also be unit-tested to ensure its correctness in filtering based on occupation.

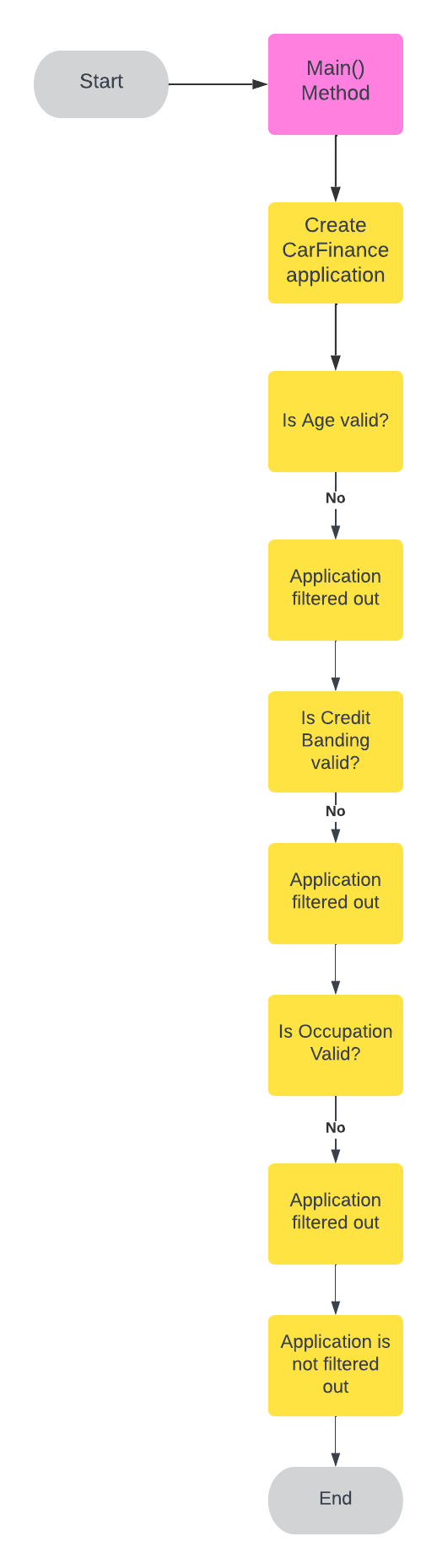
**Program Structure:**

The following section will document the structure of the program via a class diagram and a flow chart.

**UML Class diagram:**

****

The class diagram shows the classes that will comprise the program, along with each class's properties and associated methods.

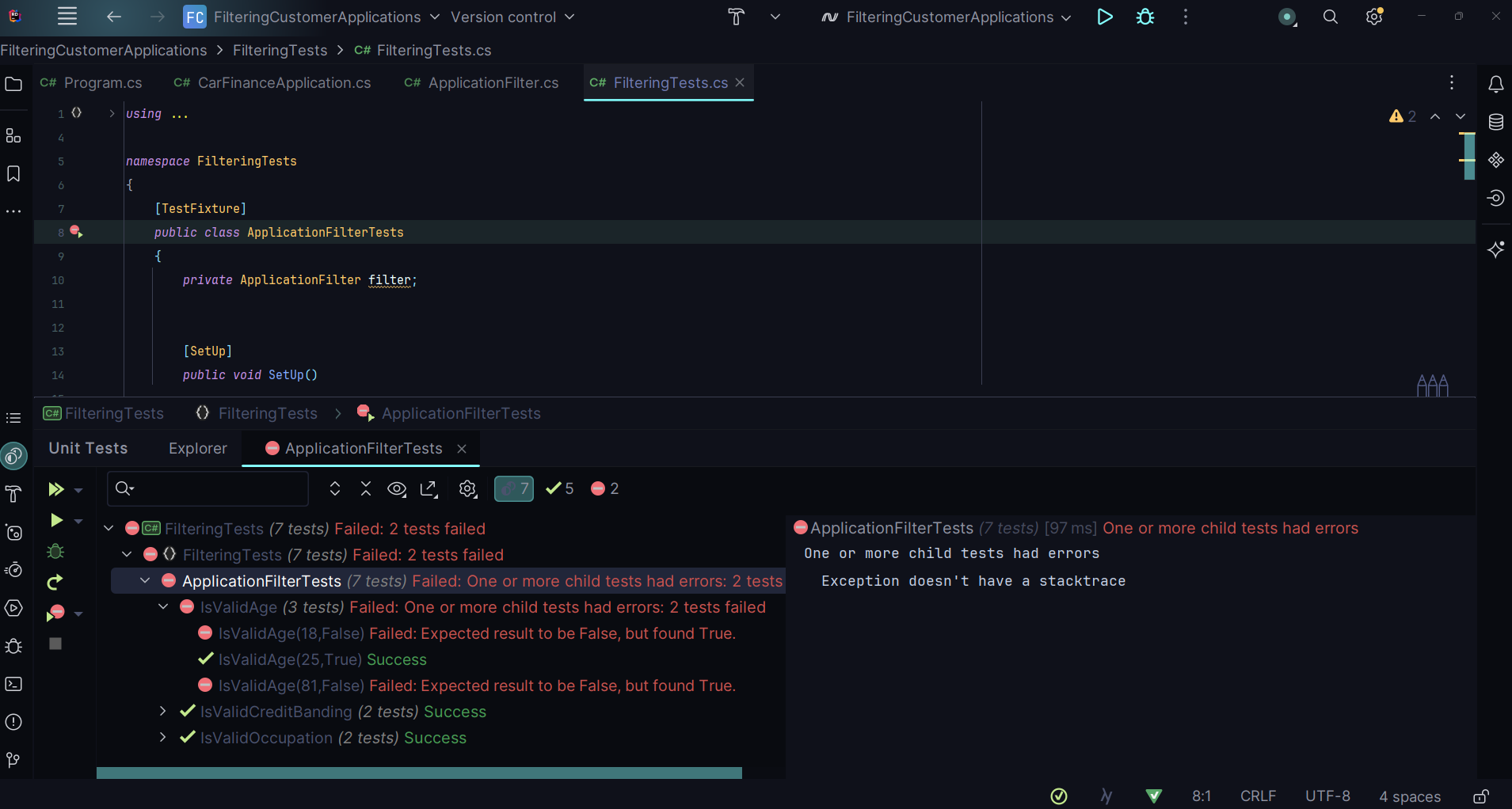
****

The flow chart shows the sequence of events that will be executed when the program is run.

1. Firstly, the user starts the program, which initialises the Main() method in Program.cs.
2. Main() creates a new instance of CarFinanceApplication using either the empty constructor or the constructor with parameters (age, name, credit banding).
3. The CarFinanceApplication instance is then passed to the ApplicationFilter class.
4. ApplicationFilter uses its methods (isValidAge(), isValidCreditBanding(), isValidOccupation()) to validate the CarFinanceApplication instance.
5. Each validation method in ApplicationFilter checks the corresponding property in the CarFinanceApplication instance:
   * isValidAge() checks if the applicant’s age is within the valid range (minAge, maxAge).
   * isValidCreditBanding() checks if the applicant’s credit banding is not in the invalidCreditBanding list.
   * isValidOccupation() checks if the applicant’s occupation meets certain criteria (you didn’t specify what these are).
6. If the CarFinanceApplication instance passes all the checks, it’s considered valid, and the application process continues. If it fails any check, the application is rejected.

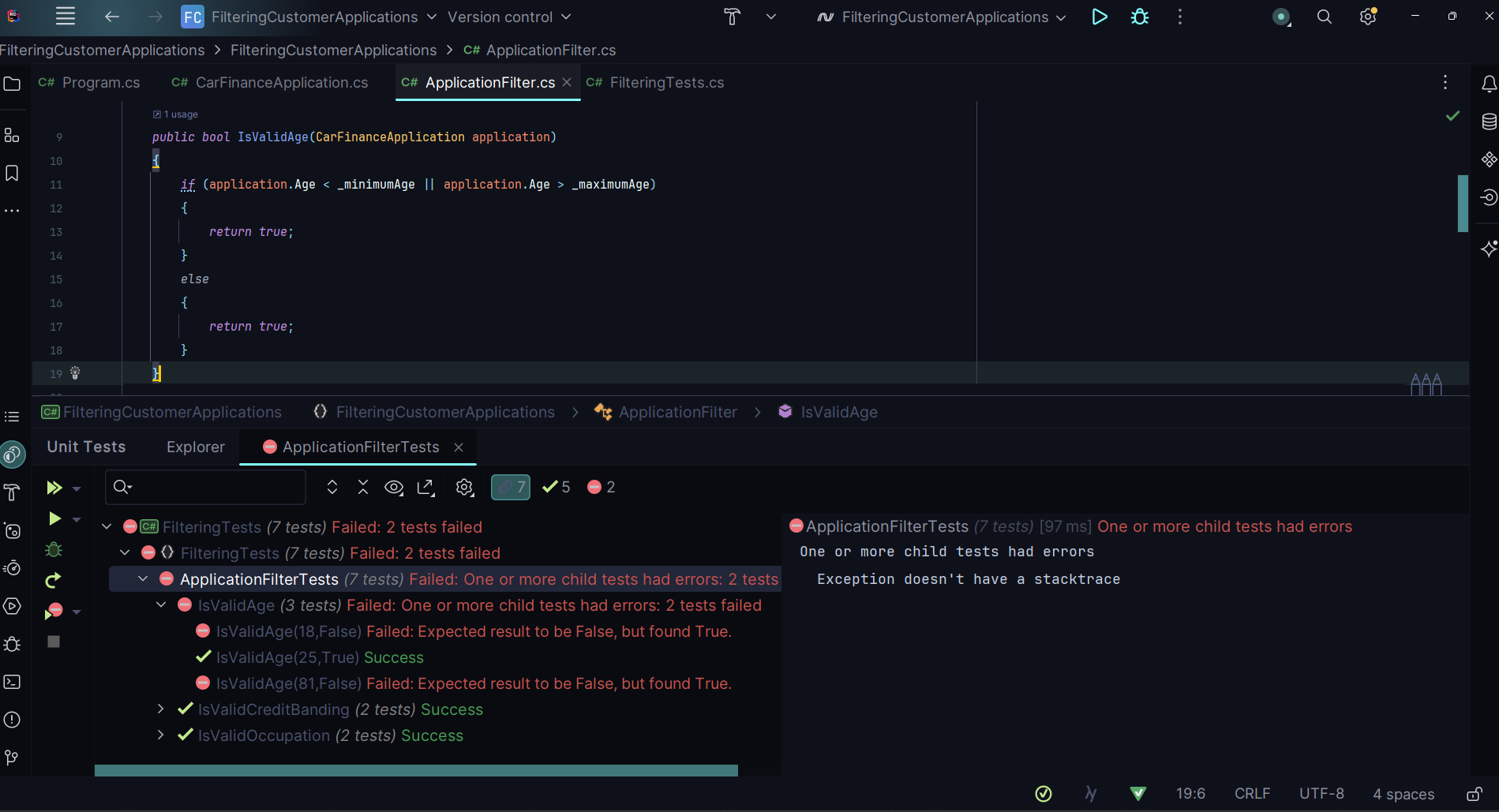
**Testing table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Type** | **Test Data** | **Reason** | **Expected Outcome** | **Actual Outcome** | **Pass / Fail** |
| IsValidAge test1 | Automated | Age: 25 | Check age filter returns correct Boolean | result = true | true | pass |
| IsValidAge test2 | Automated | Age: 18 | Check age filter returns correct Boolean | result = false | true | fail |
| IsValidAge test3 | Automated | Age: 81 | Check age filter returns correct Boolean | result = false | true | fail |
| IsValidCreditBanding 2 | Automated | Credit banding.C | Check credit banding filter returns correct Boolean | result = true | true | pass |
| IsValidCreditBanding 3 | Automated | Credit banding.A | Check credit banding filter returns correct Boolean | result = false | false | pass |
| IsValidOccupation test 1 | Automated | Occupation: “Engineer” | Check occupation filter returns correct Boolean | result = true | true | pass |
| IsValidOccupation test 2 | Automated | Occupation  **“**Driver” | Check correct Boolean is returned | result = false | false | pass |



After implementing my

After writing and implementing my application filter methods following the creation of my tests, I found that two of the tests were failing. The expected outcome in the failing tests was false but proved to be true. Based on this message I realised there was an error with the logic in my ‘isValidAge’ method.



The issue was that I had accidentally written true in both the if and else blocks.

After correcting this error, all my tests were passing.

