# Starting TDD

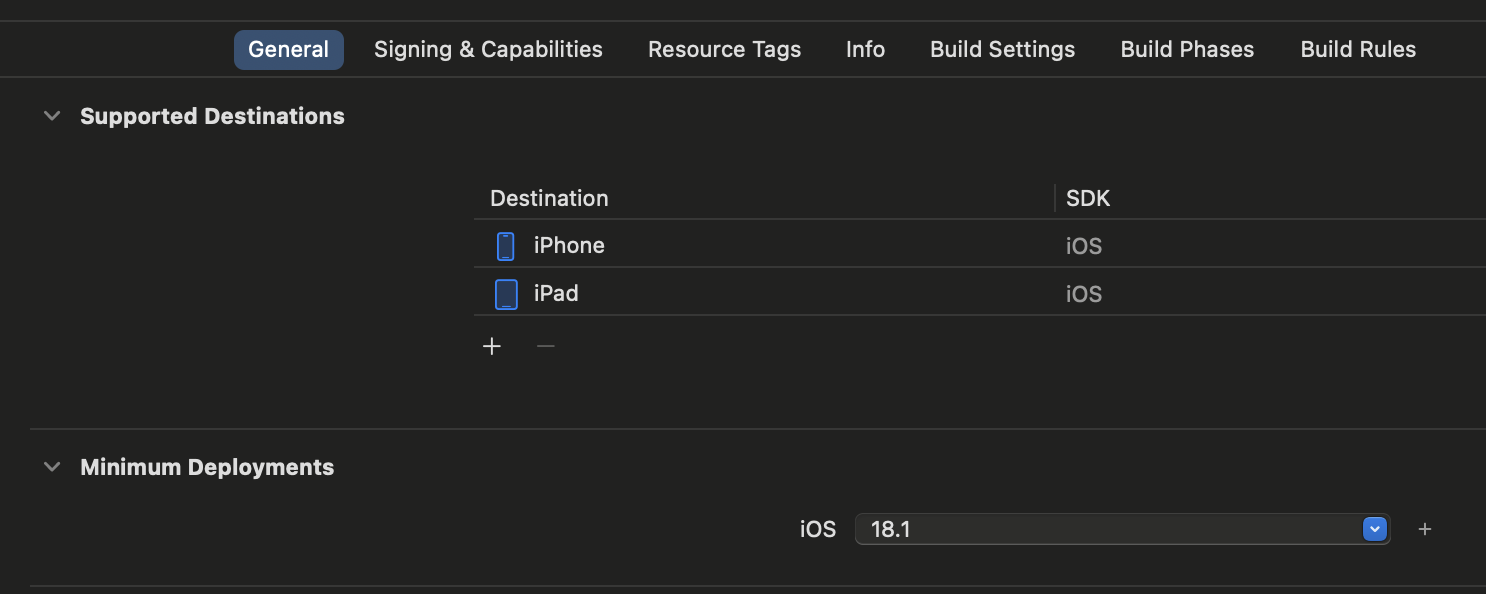
# Scenario (QLC-1):

Before we begin, please ensure your project is set to use **Swift 6**. You can check or change this in Xcode by selecting your project in the Project Navigator, navigating to the **Build Settings** tab, and searching for **Swift Language Version**. (Search Swift Language Version in the bar)

Make sure **Swift 6** is selected, as shown in the image below:

# 

Next, let’s set our **minimum deployment target** and the **supported device types**.  
This ensures that when you build and run your app, it targets the correct iOS version and devices. In Xcode, go to the **General** tab of your project settings. Under **Supported Destinations**, select the devices you want to support. Now, set the **Minimum Deployment** version to match the earliest iOS version you want your app to run on.



### Objective

In this exercise, you will practice **Test-Driven Development (TDD)** by writing a simple test for a TaskListViewModel.  
Your goal is to write a test that verifies a newly initialized TaskListViewModel starts with an empty list of tasks.

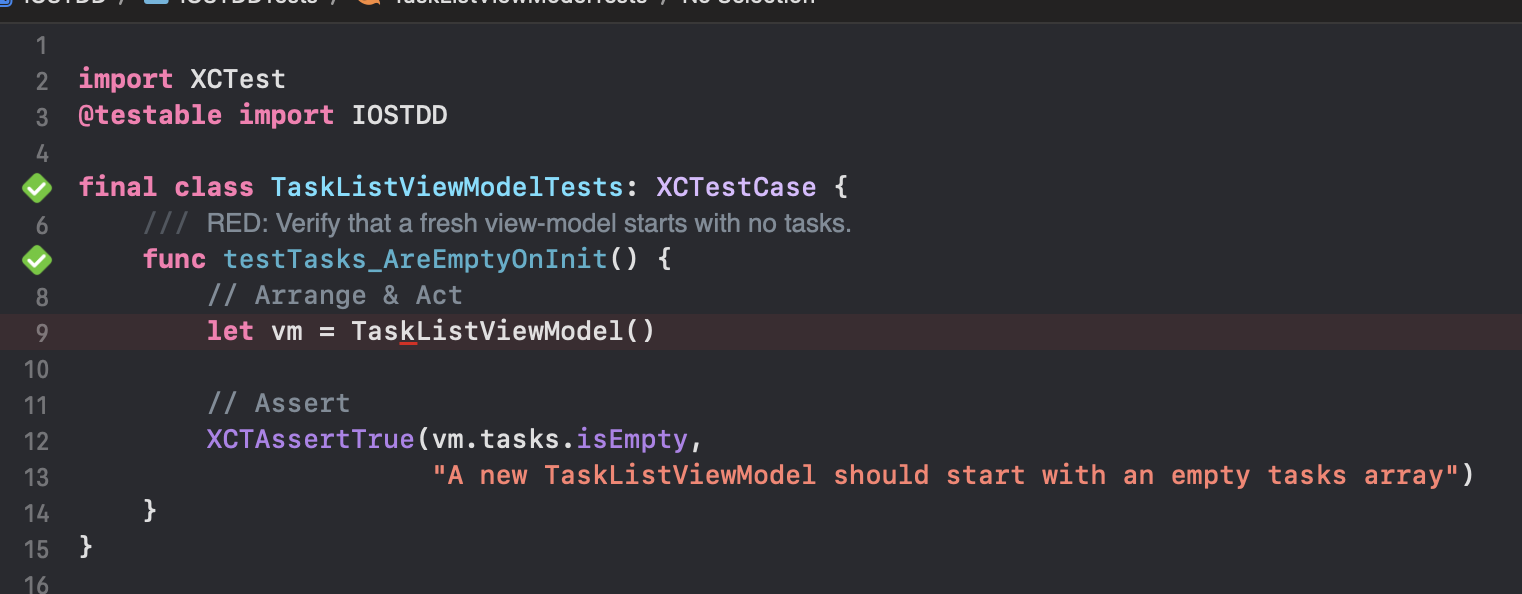
### Step 1: Define the Problem and Requirements

We want to ensure that when we create a new instance of TaskListViewModel,  
its tasks array is empty by default.

### Step 2: Write the Initial Test

Following the **RED** phase of **TDD**, start by writing a failing unit test that expects this behavior.

Example:



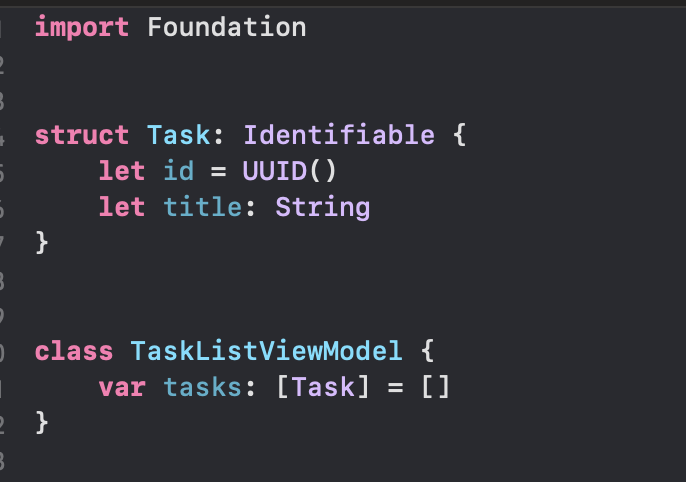
### Step 3: Run the Test

Run your test.

* If the test fails, that's **good**! You're on the right path with TDD.
* Now fix the code until the test passes.

### Step 4: Implement the Code

After writing the test, implement the TaskListViewModel with a tasks property that is empty on initialization.



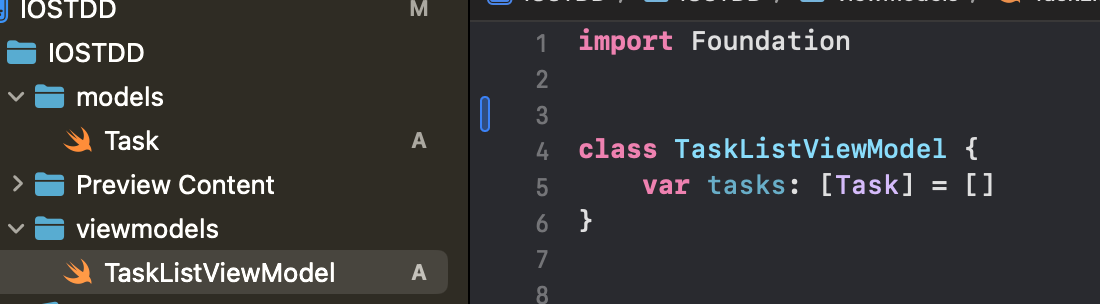
## Step 5: Refactor (if needed)

Now that our test passes, it’s time for the **Refactor phase** of TDD.

In our case, the code is simple, but instead of leaving it all in one place, we’ll improve its **structure and readability** by organizing it better for future growth:

* Move the Task model into its folder and file, e.g., Model/Task.swift.
* Move the TaskListViewModel into a dedicated ViewModels folder with a file named TaskListViewModel.swift.
* Keep the TaskListViewModelTests inside a Tests group or folder.

This setup will make it easier to maintain and extend your app as it grows.



## Step 6: Explanation

Test-Driven Development (TDD) is a core practice in software development. We start with writing the tests first and then implement just enough code to make the tests pass.  
  
- Red: Write a test that initially fails.  
- Green Write just enough code to pass the test.  
- Refactor: Refactor the code without changing its functionality.

### Step 6.1: Implementing Additional Features

When practicing **TDD**, it's important to move in **small, focused steps**.  
Instead of writing the entire class or multiple methods at once, we’ll work incrementally, adding and testing **one behavior at a time**.

#### The Process

1. **Write one test** for a new feature or condition.  
   Think carefully:  
   “What is the next smallest behavior this TaskListViewModel should support?”  
   For example:
   * *“It should allow adding a new task with a title.”*
   * *“It should allow removing a task by its ID.”*
2. **Implement just enough code** to make that test pass.  
   Don’t build a full-featured method yet, handle only the specific case you’re testing.
3. **Run the tests.**
   * The **new test** should pass.
   * All **existing tests** must still pass.
4. **Refactor.**Once the test passes, examine your code:
   * Is there duplication?
   * Can it be clearer or better structured?
   * Are your method names meaningful?
5. Refactor as needed while keeping your tests green.
6. **Repeat for the next feature.**

This methodical, test-driven approach will help you grow your TaskListViewModel logically, feature by feature, with confidence that everything works as expected at every step.