Exercise: integrate 3rd party "enhancement" library

Introduction

This exercise is based on a case that came up during the development of the albelli photo book application. That particular case dealt with integrating a 3rd party photo enhancement library. This exercise deals with a simplified version of such a library; instead of processing a photo it just manipulates a character string.

The background story

The Mac development team is working on a macOS Cocoa application, written in Objective-C and Swift. Development is done with the latest versions of Xcode and Apple's SDK. The application is designed to run on Mac OS X 10.11 (El Capitan) and later.

A license has been acquired for a 3rd party library ("GPEnhance") for realizing a complicated feature, which would have been cost-prohibitive to develop in-house. Being available for multiple platforms (macOS, Windows, Linux), the library has a C-style interface. The Mac team has been provided with the static library for macOS.

Your assignment

Exercise notes:

- You use the latest version of Xcode (10.x) and Swift (4.2 or 5).
- Use local git repository and create commit sequence that would represent your best day-to-day practice.
- After completing the exercise, zip the local repo folder and email it to us, or give us the access to the hosted repository.

Part 1

The `GPEnhance` library comes with an "old school" C interface. Everyone on the team agrees that stuff like `char *` is not part of the Objective-C, let alone Swift, idiom, so the library's C interface should be hidden as much as possible from the rest of the application.

Your task: design and implement a Swift class that encapsulates the C-style interface of the `GPEnhance` library. The other source files in the application just have to use your class, and are fully unaware of the C library's header file.

For the benefit of the team members, include markup comments to document your class' interface.

Also create some unit tests for your class.

You will find the documentation of the `GPEnhance` interface in its header file.

Part 2

While using the GP Enhance library, the development team discovered that it has serious restrictions with respect to multi-threading. First of all, the function `gp_enhance_create` cannot be called on multiple threads at the same time. Furthermore, the opaque `GPEnhancer` object, allocated by `gp_enhance_create`, can be used in only one thread at the same time.

Yet to meet the performance requirements, the application needs to run the `gp_enhance_execute` function in parallel, with a maximum of 8 simultaneous threads. The library manufacturer has confirmed that this can be accomplished by using multiple `GPEnhancer` objects, one for each thread.

Your task: design and implement a Swift class that exposes, to the rest of the application, an `enhance` operation that can be called simultaneously on multiple threads, effectively removing the thread restrictions. This class should use the encapsulation class you created in part 1 of the exercise.

For the benefit of the team members, include markup comments to document your class' interface.

Also create code that demonstrates the multi-threading to work properly. For your convenience, the `GPEnhance` library provided with this exercise has a high probability of crashing if the threading restrictions are not respected.