Lazy Initialization Pattern

Introduction

- The lazy initialization pattern embodies the just-in-time philosophy of data delivery.
- It delays the construction of values or data structures until that data is actually needed.

Implementation

- Create a private instance or private static variable that will eventually contain the data, but is initialized with a placeholder value.
- Wrap the variable in a property that provides a public getter method.
- In the getter method, test the variable for its placeholder value. If it has not been initialized, construct it and save the results. Return the newly, or previously, constructed data to the sender.

Contd.

- An object constructor that creates an object, quickly and with minimal memory demands, is called a lightweight constructor.
- The constructor can be lightweight even if the object is heavyweight

Application

- The memory or resources required to create or calculate the data is excessive and can be deferred.
- There is a significant possibility that the data will never be needed.
- The prerequisites of the data do not exist, or are not available, when an object is instantiated.
- Lazy initialization is a useful design pattern for both global and instance variables. It optimizes performance by constructing only the information that's actually needed, and defers optional and ancillary calculations to a more appropriate time.

Pros & Cons

- This pattern boosts the application start up time as the approach focuses on the necessary application objects.
- The performance of the application may degrade in terms of time as there is a need of checking regarding loading of an object is needed or not; for this the coding has become complicated.

Reference

- *(2009) Lazy Initialization Pattern. In: Learn Objective-C for Java Developers. Apress*