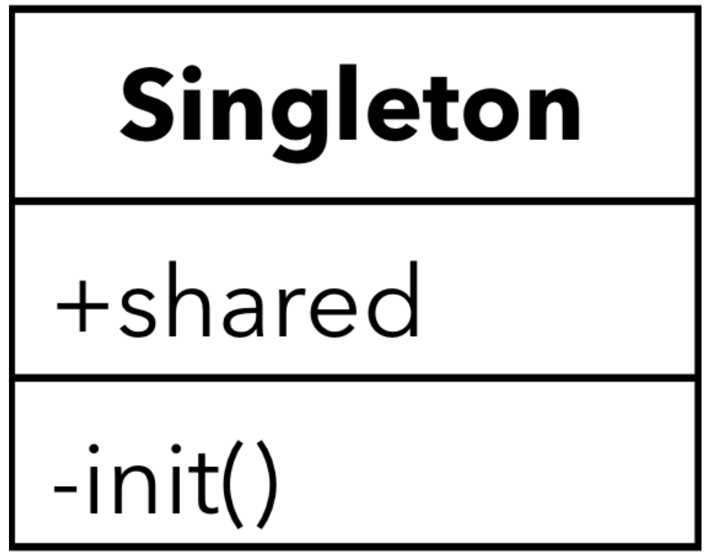
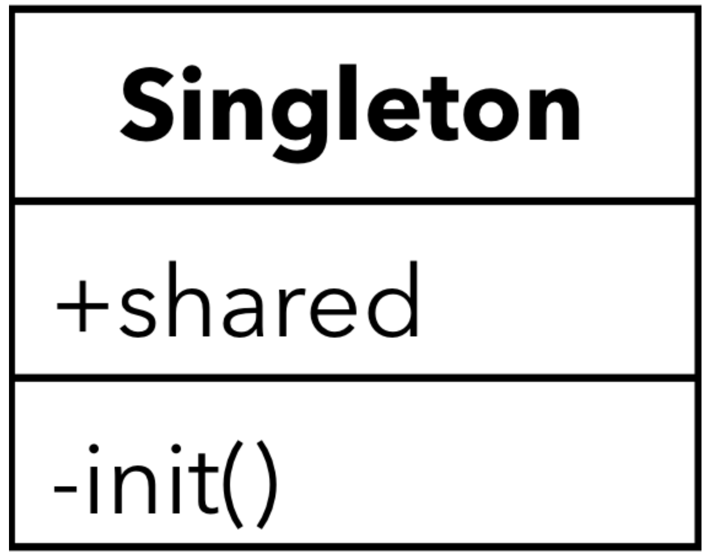
Singleton

Ensure that a class has only one instance.



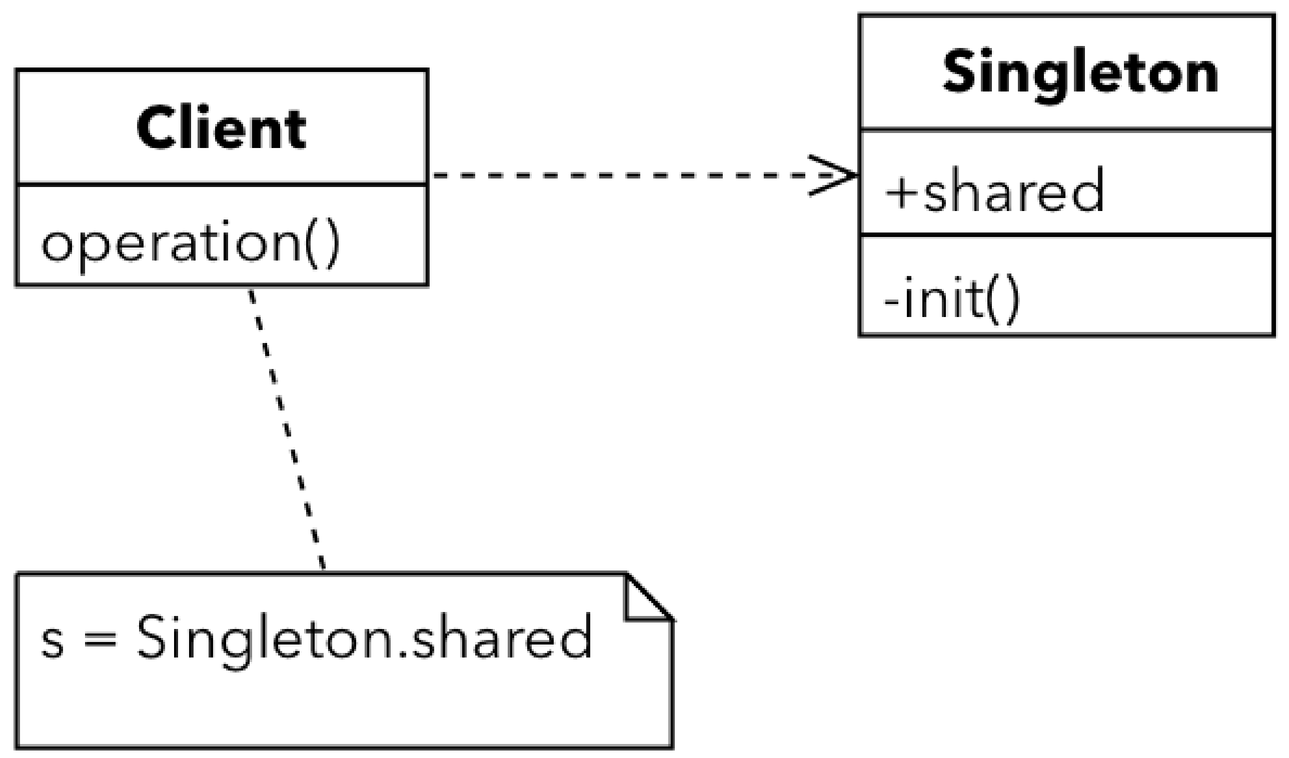
Hidden initializer to prevent instantiation



Hidden initializer to prevent instantiation

Global access point to the shared instance

Accessing the Singleton instance



Value types can’t implement the Singleton

Value types can’t implement the Singleton

Singleton classes must not adopt the *NSCopying* protocol

AppDelegate example / Xcode

When to Use the Singleton?

When to Use the Singleton?

Represent a single resource

When to Use the Singleton?

Represent a single resource

Consolidate usage across components

How to Avoid Common Pitfalls?

How to Avoid Common Pitfalls?

Ensure thread-safety

How to Avoid Common Pitfalls?

Ensure thread-safety

Adhere to the Single Responsibility Principle

How to Avoid Common Pitfalls?

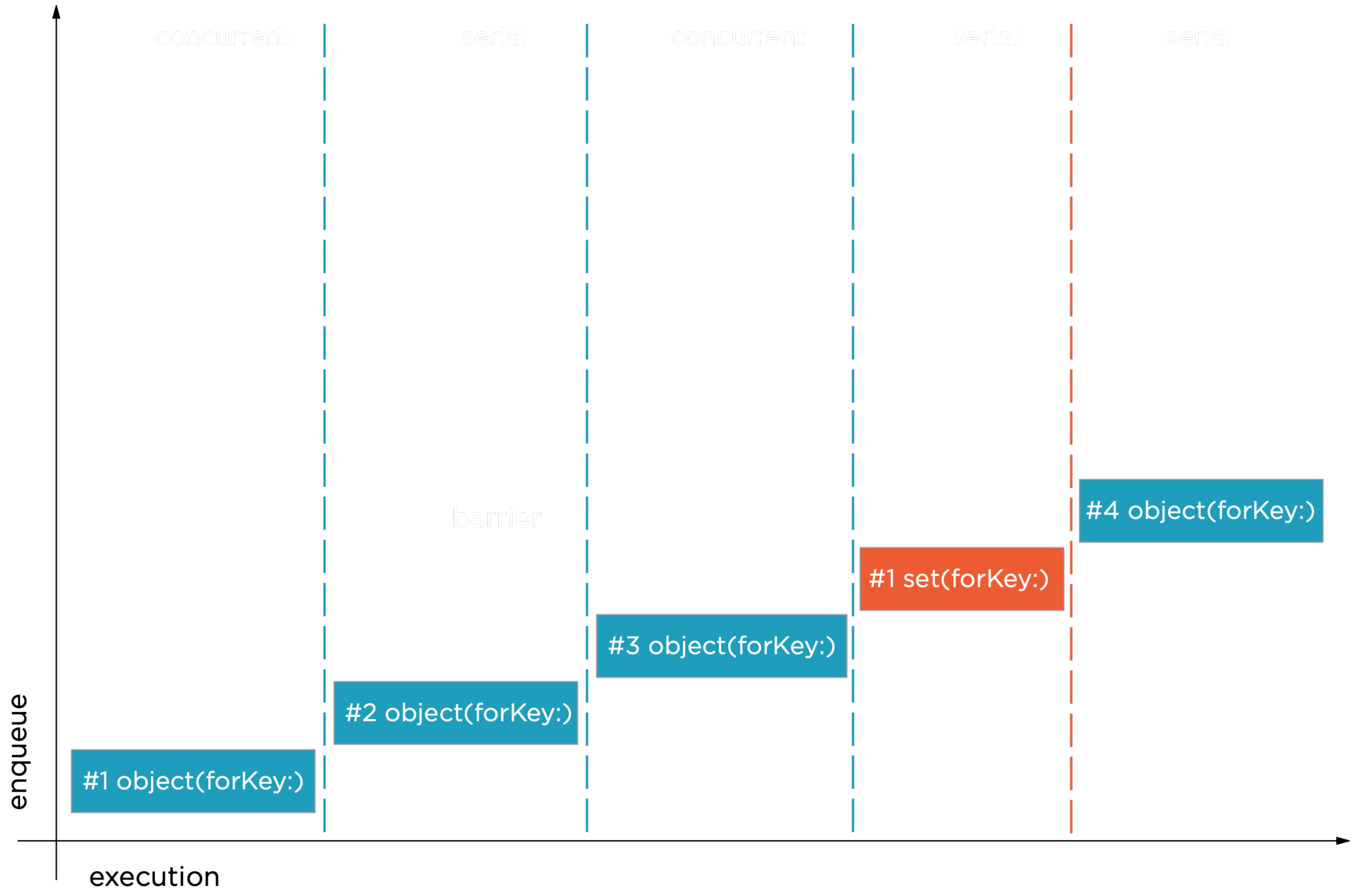
Ensure thread-safety

Adhere to the Single Responsibility Principle

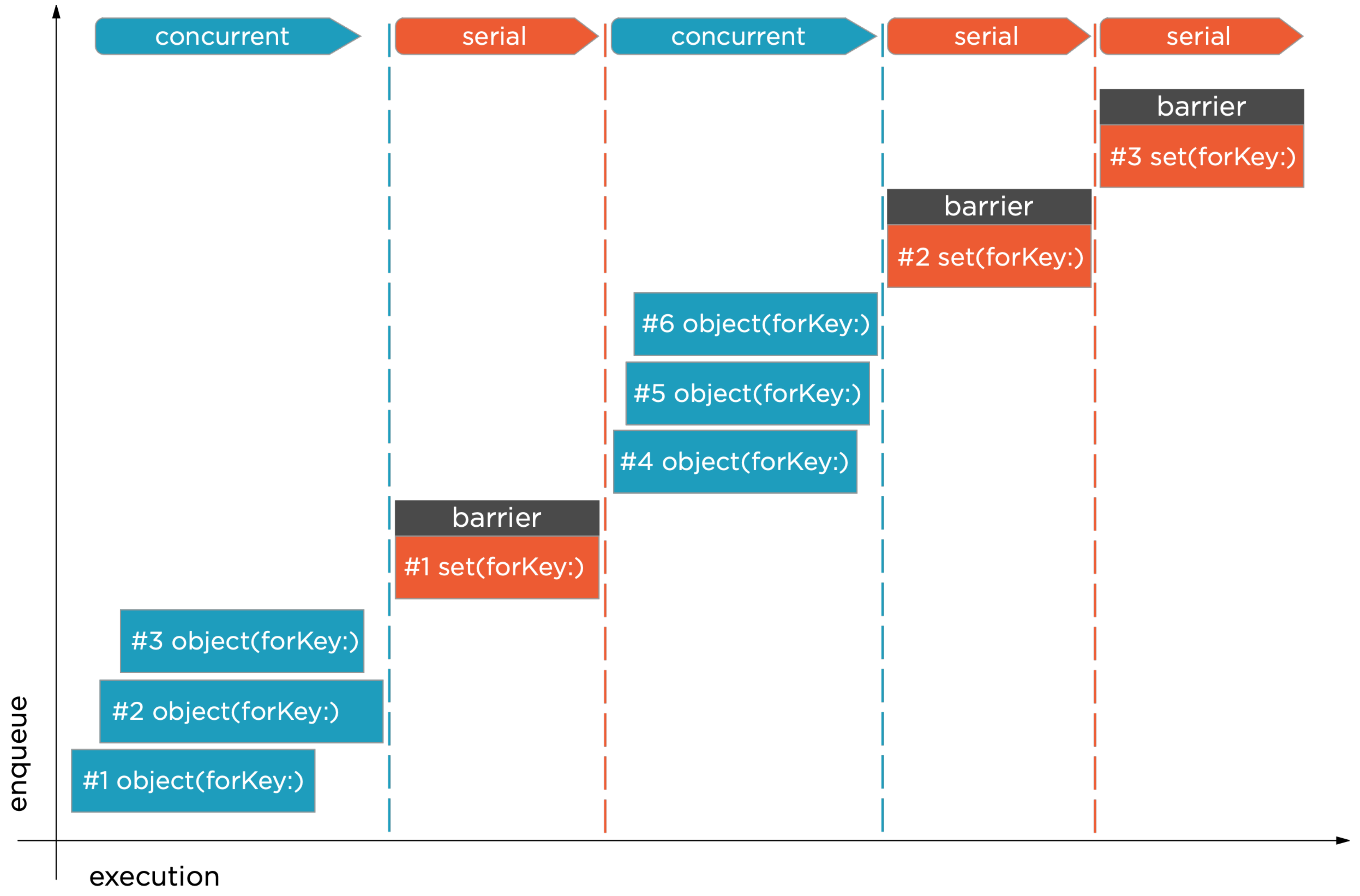
Avoid tight coupling

Changing the Singleton may impact depending types

Serial Queue



Concurrent Queue with Barrier



The Singleton

Purpose

- Ensure a class only has one instance

Prevent cloning

- No value types
- No NSCopying classes

Common pitfalls

- Lack of thread-safety
- Using singletons as global containers
- Tight-coupling