WHAT IS ACTOR?

Swift introduced **Actor (Swift 5.5)** to solve **concurrency issues** that classes cannot handle safely.

By Shubam Gupta

What is a Class in Swift?

A class is a **reference type** used to **model shared, mutable state**.

```swift
class Counter {
    var value = 0

    func increment() {
        value += 1
    }
}
```

⚠️ Problem with class

When multiple threads access the same class instance:

- Data race
- Inconsistent state
- Crashes that are hard to reproduce

```swift
DispatchQueue.global().async {
    counter.increment()
}
```

➡️ Swift does NOT protect **classes from concurrent access.**

Why Actor was introduced? 🚀

Swift introduced Actor (Swift 5.5) to solve concurrency issues that classes cannot handle safely.

❌ **Problems with Class in Concurrency**

- Manual locking (DispatchQueue, NSLock)
- Deadlocks
- Callback hell
- Hard-to-debug race conditions.

✅ ACTOR SOLVES THIS BY:

- Isolating mutable state
- Allowing only one task at a time to access its properties

What is an Actor?

An actor is a **reference type** like class, but:

It protects its internal state automatically.

```swift
actor Counter {
    var value = 0

    func increment() {
        value += 1
    }
}
```

Key Rule

👉 Actor properties are not directly accessible from outside.

How Actor works internally 🧠

- Actor maintains its own serial executor
- Only one task can touch actor's state at a time
- Compiler enforces safety using await

➡️ Thread safety by design

What problem does Actor solve that Class cannot?

Problem	Class	Actor
Data race	❌	✅
Thread safety	Manual	Automatic
Locking needed	Yes	No
Deadlock risk	High	None
Compiler safety	❌	✅

Where & Why should we use Actor?

Use Actor when you have **shared mutable state accessed concurrently**.

✅ Real-world use cases

- Network response cache
- Analytics event logger
- Database / persistence layer
- In-memory cache
- Shared app state

```swift
actor ImageCache {
    private var cache: [String: Data] = [:]

    func save(_ data: Data, for key: String) {
        cache[key] = data
    }

    func get(for key: String) -> Data? {
        cache[key]
    }
}
```

Actor vs Class (Quick Comparison)

Feature	Class	Actor
Type	Reference	Reference
Thread safety	❌	✅
Concurrency safe	❌	✅
Requires await	❌	✅
Compiler enforced	❌	✅

Advantages of Actor ✅

- Built-in thread safety
- No locks, no queues
- Cleaner async/await code
- Compiler prevents unsafe access
- Easier to reason about state

Disadvantages of Actor ⚠️

- Requires async/await
- Slight performance overhead (context switching)
- Cannot subclass actors
- Not ideal for simple, non-shared logic
- iOS 13+ only (full concurrency support from iOS 15)

When NOT to use Actor

❌ Simple data models

❌ UI-related logic (use @MainActor)

❌ Performance-critical synchronous code

❌ No shared mutable state

One-Liner 🎯

“Actor was introduced to provide compile-time enforced thread safety for shared mutable state, which classes cannot guarantee.”

🔚 Conclusion

- Class → **Flexible but unsafe in concurrency**
- Actor → **Safe, predictable, modern concurrency tool**
- Swift moved from **manual thread safety to compiler-enforced safety**.

Thank you
By Shubam Gupta