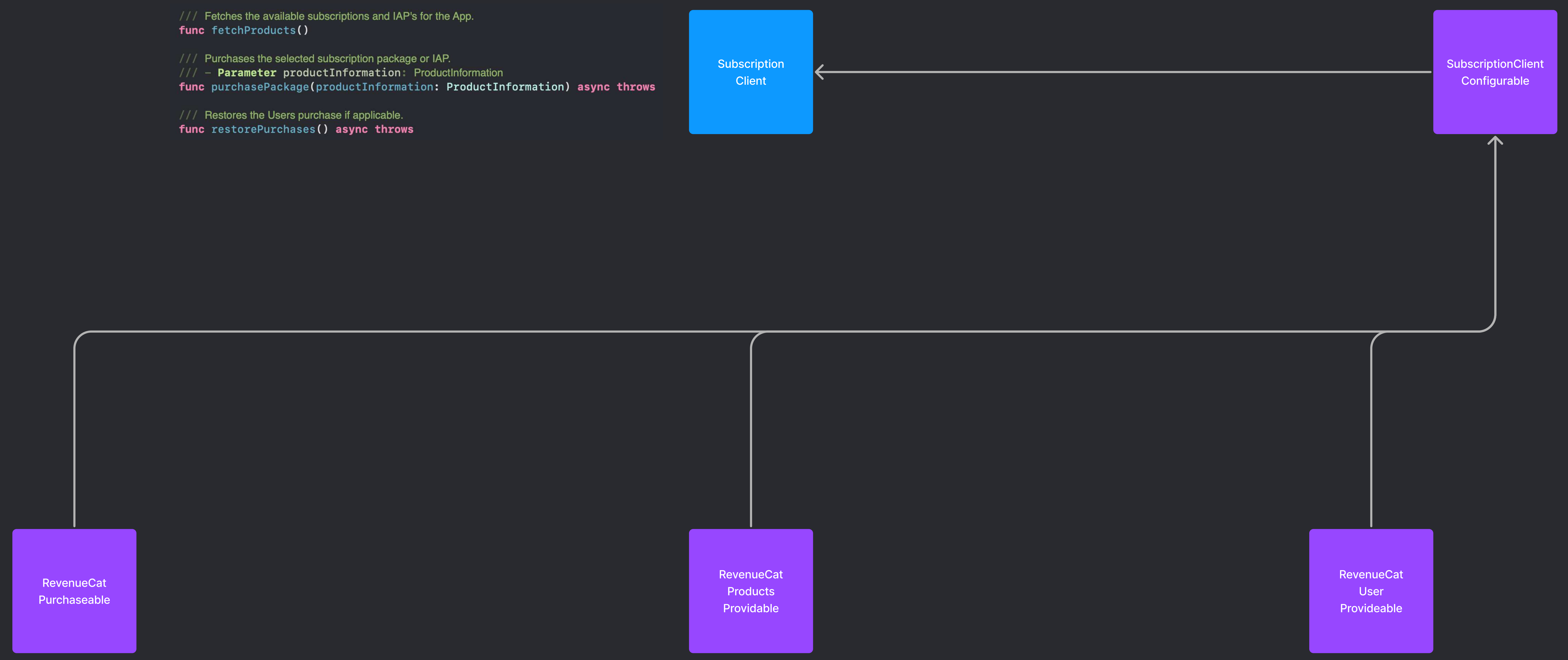
Subscription Client

Responsible for the orchestration of in-app purchases (IAP) and subscription management within your application.
 - Implements `SubscriptionClientConfigurable` for a structured approach to subscription and IAP management.
 - Uses `NetworkMonitoringService` for real-time network state monitoring.
 - Employs `RevenueCatProductsProvidable` and `RevenueCatPurchaseable` protocols to interact with RevenueCat's SDK.


Subscription Client
SubscriptionClient Configurable
RevenueCat Purchaseable
RevenueCat Products Providable
RevenueCat User Provideable

```swift
/// The SubscriptionClientConfigurable  protocol defines a comprehensive set of requirements for managing in App Purchases & Subscriptions within the App.
public protocol SubscriptionClientConfigurable {

  /// `RevenueCatPurchasable` responsible for the purchasing and restoration of Subscriptions and IAP
  var purchaseManager: RevenueCatPurchaseable { get }

  /// Responsible for updating the UI when fetching subsctiptions or IAP's.
  var productLoadingState: LoadingState { get }

  /// `RevenueCatProductsProvidable` responsible for listing the current available products/subscriptions in the App.
  var productProvider: RevenueCatProductsProvidable { get }

  /// The available products available for the user to purchase.
  var availableProducts: [ProductInformation] { get }

  /// Responsible for retrieving information about the Use.
  /// This protocol extends PackageDebuggable for debugging purposes and conforms to RevenueCat.PurchasesDelegate
  /// to handle updates or changes in purchase information directly from RevenueCat.
  var userProvider: RevenueCatUserProvidable { get }


  /// Responsible for updating the UI with changes to the purchase/restoration being made by the user.
  var subscriptionLoadingState: LoadingState { get }

  /// Fetches the available subscriptions and IAP's for the App.
  func fetchProducts()

  /// Purchases the selected subscription package or IAP.
  /// - Parameter productInformation: ProductInformation
  func purchasePackage(productInformation: ProductInformation) async throws

  /// Restores the Users purchase if applicable.
  func restorePurchases() async throws


  /// Attempts to refetch the available subscriptions or IAP"s if they are empty due to network issues etc.
  func observeNetworkStateAndFetchProductsIfNeeded()

  /// Used to monitor the state of the Users network connectivity.
  var networkMonitoringService: NetworkMonitoringService { get }

  /// Initalizes the SubscriptionClient
  /// - Parameters:
  ///   - productProvider: RevenueCatProductsProvidable
  ///   - userProvider: RevenueCatUserProvidable
  ///   - purchaseManager: RevenueCatPurchaseable
  ///   - networkMonitoringService: NetworkMonitoringService
  init(
    productProvider: RevenueCatProductsProvidable,
    userProvider: RevenueCatUserProvidable,
    purchaseManager: RevenueCatPurchaseable,
    networkMonitoringService: NetworkMonitoringService
  )
}
```

```swift
/// Fetches the available subscriptions and IAP's for the App.
func fetchProducts()

/// Purchases the selected subscription package or IAP.
/// - Parameter productInformation: ProductInformation
func purchasePackage(productInformation: ProductInformation) async throws

/// Restores the Users purchase if applicable.
func restorePurchases() async throws
```

```swift
/// RevenueCatPurchasable is responsible for the purchasing and restoration of Subscriptions and IAP
public protocol RevenueCatPurchaseable: PackageDebuggable {
  ///  Initiates the purchase process for a specified product.
  /// - Parameter productInformation: An instance containing all necessary information about the product to be
  /// purchased, such as its identifier and price.
  /// - Throws: An error if the purchase process cannot be initiated or completed. This allows for error handling to
  /// manage issues like network errors, user cancellation, or other purchase failures.
  /// This function is asynchronous, indicated by `async`, allowing it to perform network requests or other long-running
  /// tasks in a non-blocking way, awaiting their completion.
  func purchasePackage(productInformation: ProductInformation) async throws

  /// Initiates the process to restore previously made purchases.
  /// - Throws: An error if the restore process cannot be initiated or completed. This could be due to network issues,
  /// no purchases to restore, or other errors.
  func restorePurchases() async throws

  /// Initiaizer used to determine whether IOSLogging is enabled
  /// - Parameter allowsIOSDebugLog: Whether IOSLogging is enabled
  init(allowsIOSDebugLog: Bool)
}
```

```swift
/// RevenueCatProductsProvidable is the protocol for listing the current available products/subscriptions in the App
public protocol RevenueCatProductsProvidable: PackageDebuggable {
  /// Service for providing subscription pricing and display text, utilizing UserDefaults for local storage.
  var displayNameService: SubscriptionPricingProvidable { get }

  /// The current offering retrieved from the App Store via RevenueCat.
  var currentOffering: RevenueCat.Offering? { get }

  /// Collection of available products associated with the current offering.
  var availableProducts: [ProductInformation] { get }

  /// Retrieves the current offering from RevenueCat and the associated products.
  /// This asynchronous method may throw errors, handling situations where fetching offerings or products fails.
  /// - Returns: A tuple containing the current offering and an array of product information.
  func getOfferingAndAssociatedProducts() async throws -> (offering: RevenueCat.Offering, products: [ProductInformation])

  /// Fetches the current offering from RevenueCat asynchronously.
  /// This method may throw errors if there are issues retrieving the offering.
  /// - Returns: The current offering from RevenueCat.
  func getCurrentOffering() async throws -> RevenueCat.Offering

  /// Checks which available products are eligible for a trial period.
  /// This asynchronous method filters the products based on trial eligibility, useful for promotional displays.
  /// - Parameter availableProducts: An array of ProductInformation representing the current available products.
  /// - Returns: A filtered array of ProductInformation where each product is eligible for a trial.
  func checkProductIsEligibleForTrial(for availableProducts: [ProductInformation]) async -> [ProductInformation]

  /// Retrieves available products for a given offering.
  /// This asynchronous method fetches products associated with a specific offering, facilitating dynamic product
  /// displays based on offerings.
  /// - Parameter offering: The specific offering to retrieve products for. Can be nil, in which case it might fetch
  /// default or all products.
  /// - Returns: An array of ProductInformation for the products available under the specified offering.
  func getAvailableProducts(for offering: RevenueCat.Offering?) async -> [ProductInformation]

  /// Initializes a conforming instance with a display name service and a debug log option.
  /// - Parameters:
  ///   - displayNameService: The service used for managing subscription pricing and display names.
  ///   - allowsIOSDebugLog: A Boolean indicating whether iOS debug logging should be enabled, useful for development
  /// and troubleshooting.
  init(displayNameService: SubscriptionPricingProvidable, allowsIOSDebugLog: Bool)
}
```

```swift
/// Protocol responsible for retrieving information about the User in context with RevenueCat services.
/// This protocol extends PackageDebuggable for debugging purposes and conforms to RevenueCat.PurchasesDelegate
/// to handle updates or changes in purchase information directly from RevenueCat.
public protocol RevenueCatUserProvidable: PackageDebuggable, RevenueCat.PurchasesDelegate {
  /// The identifier for the entitlement which unlocks premium content within the app.
  /// This is used to determine access rights based on user purchases or subscriptions.
  var entitlementIdentifier: String? { get }

  /// Asynchronously retrieves the current customer's information from RevenueCat.
  /// This includes subscriptions, purchases, and any entitlements associated with the user.
  /// - Returns: An optional RevenueCat.CustomerInfo object containing the user's purchase history and entitlements.
  /// - Throws: An error if there's an issue fetching the customer information, such as a network error or configuration
  /// issue.
  func getCustomerInformation() async throws -> RevenueCat.CustomerInfo?

  /// Determines if the current user has a valid entitlement and active subscription.
  /// This is crucial for gating content or features that require a subscription.
  /// - Returns: A Boolean value indicating whether the user is currently subscribed through Apple's In-App Purchase.
  func isAppleSubscriber() -> Bool

  /// Sets up the RevenueCat.PurchasesDelegate and synchronizes the RevenueCat.CustomerInfo.
  /// This method is typically called during app initialization or when the user's purchase information needs to be
  /// refreshed.
  /// It ensures that the app's state is up-to-date with the latest purchase and subscription information from
  /// RevenueCat.
  func configure()

  /// Initializes a new instance of a conforming type with an optional entitlement identifier and a debug logging
  /// option.
  /// - Parameters:
  ///   - entitlementIdentifier: The identifier for the specific entitlement that unlocks app content, if applicable.
  ///   - allowsIOSDebugLog: A Boolean flag indicating whether iOS debugging logs should be enabled, useful for
  /// development and troubleshooting.
  init(entitlementIdentifier: String?, allowsIOSDebugLog: Bool)
}
```