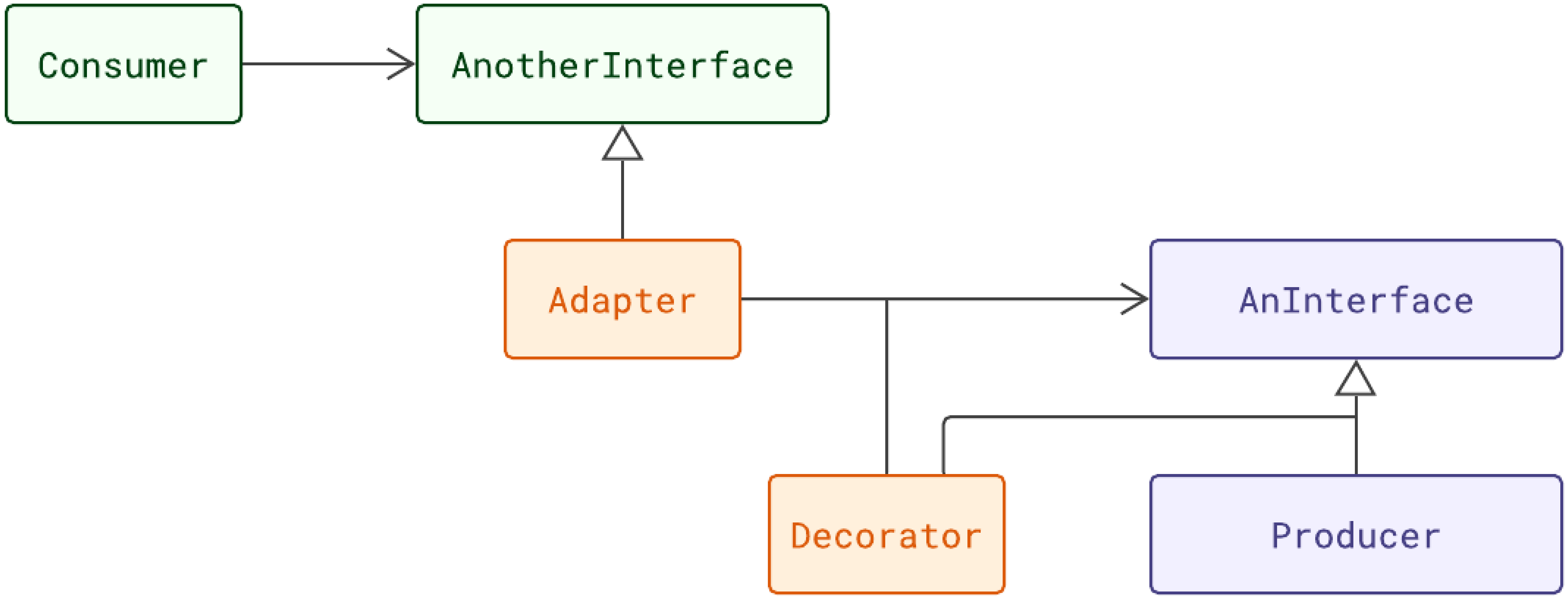
Design Patterns in C# Made Simple

Module 3 Adapting to a Different Interface with the Adapter Pattern

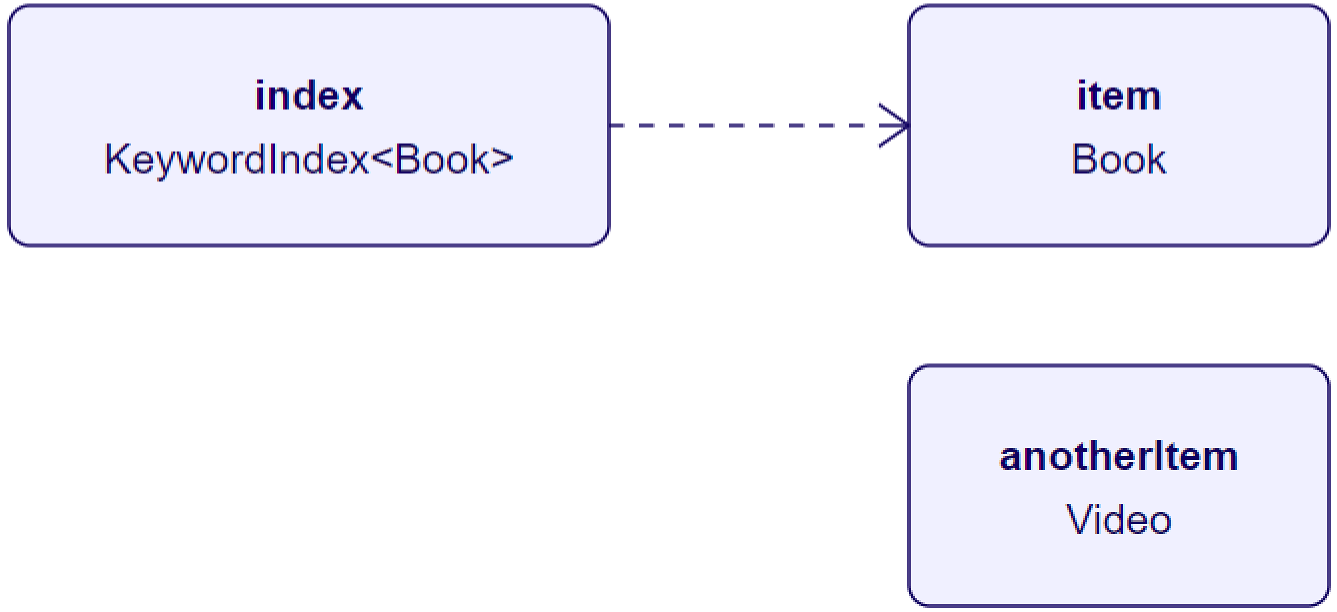
Zoran Horvat
CEO at Coding Helmet

http://codinghelmet.com
zh@codinghelmet.com
zoranh75

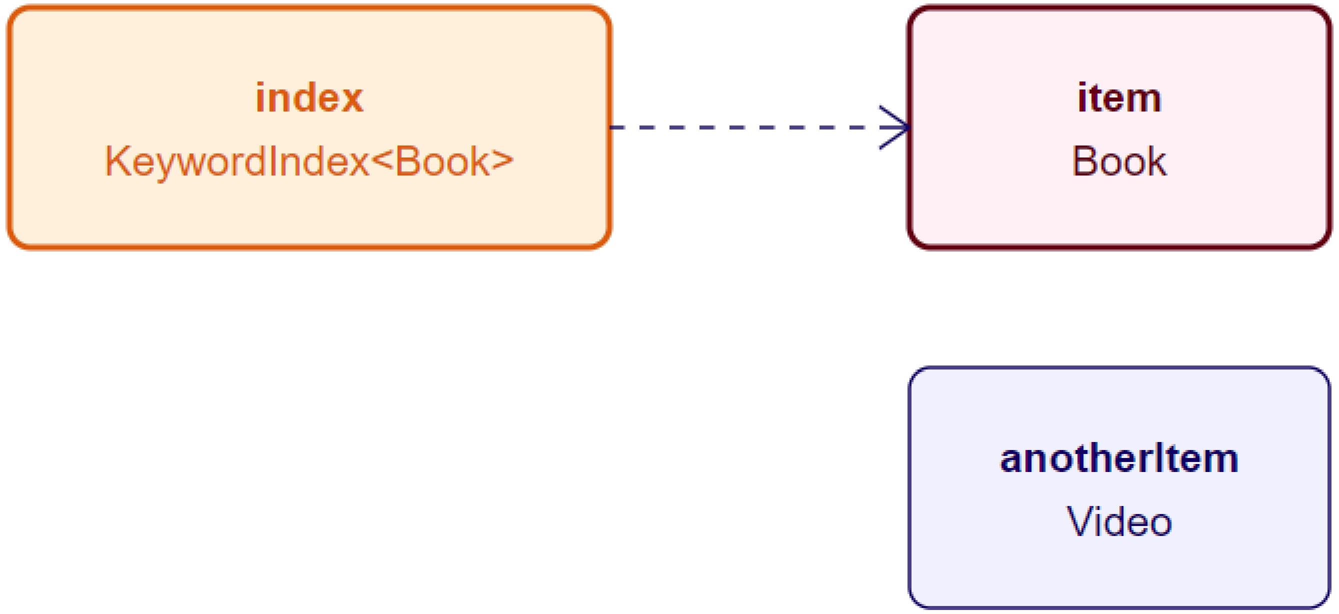
From Decorator to Adapter



Motivation to Apply the Adapter Pattern

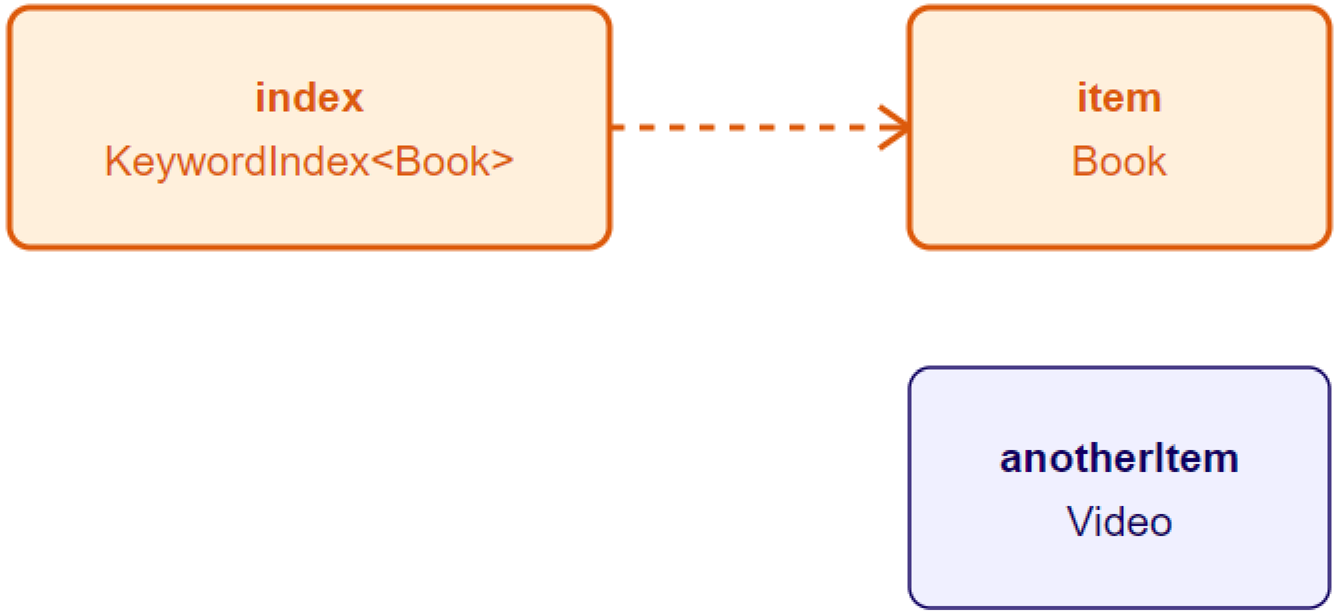


Motivation to Apply the Adapter Pattern



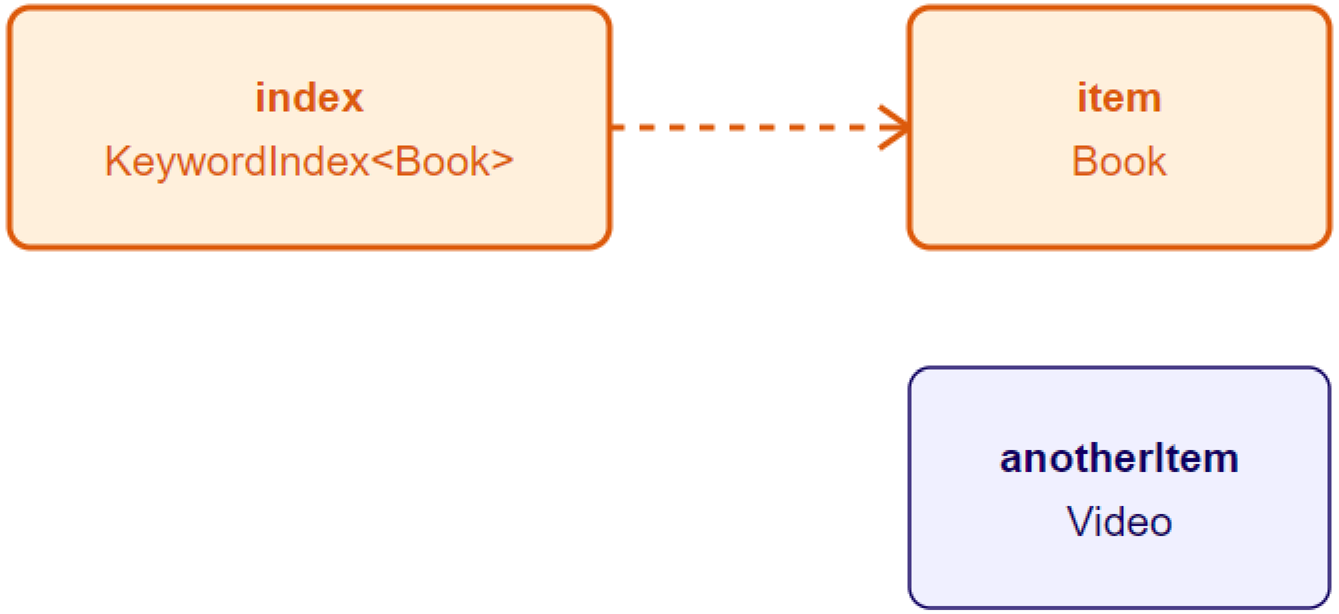
External code calls index.Add(item)

Motivation to Apply the Adapter Pattern



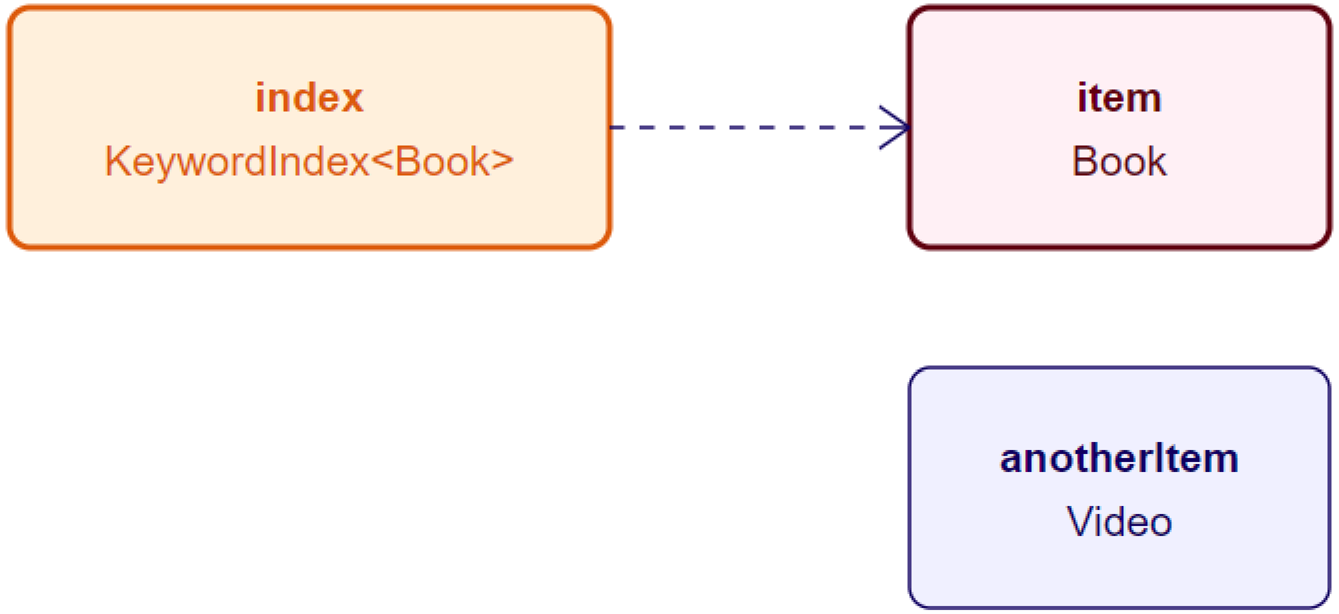
index calls item.Keywords

Motivation to Apply the Adapter Pattern

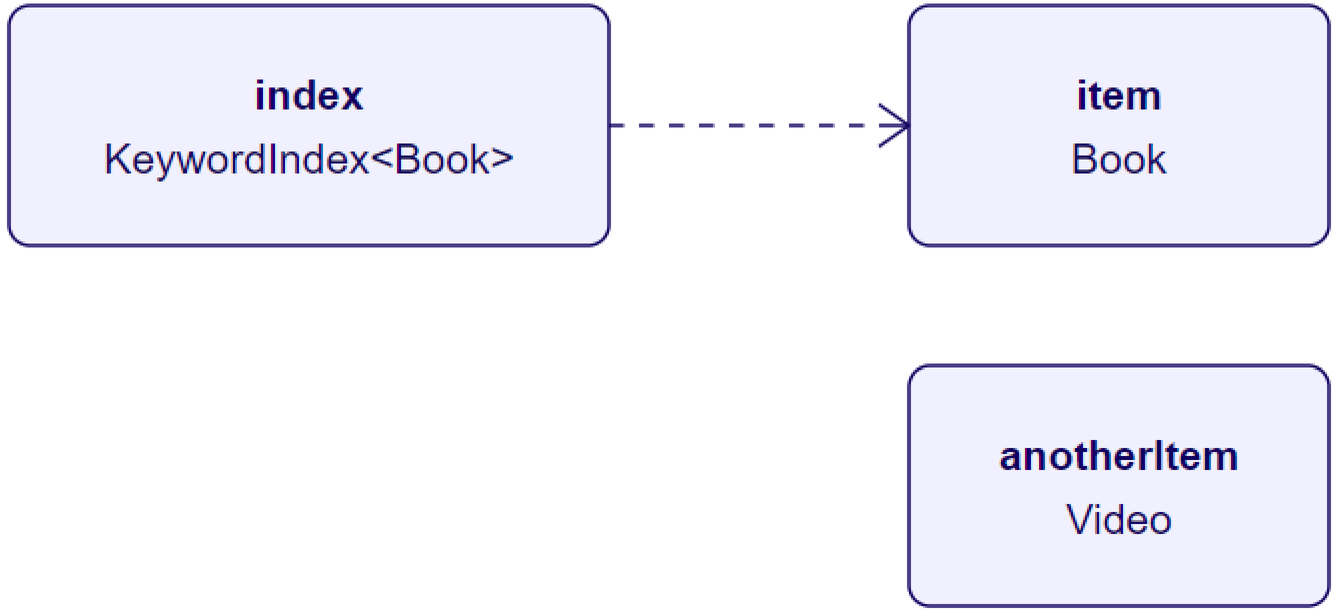


item.Keywords returns ["long", "boring"]

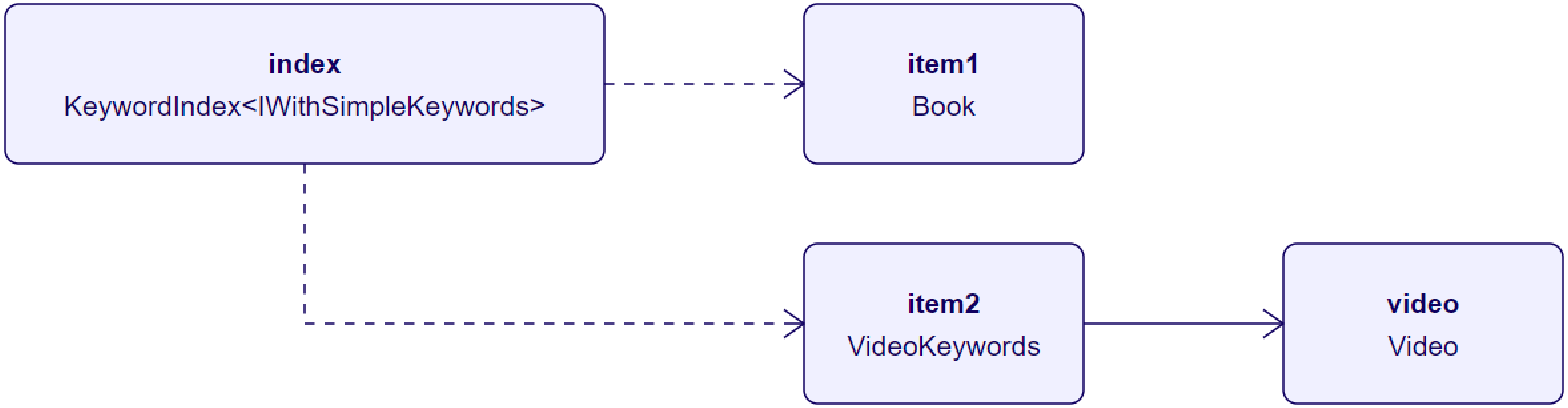
Motivation to Apply the Adapter Pattern



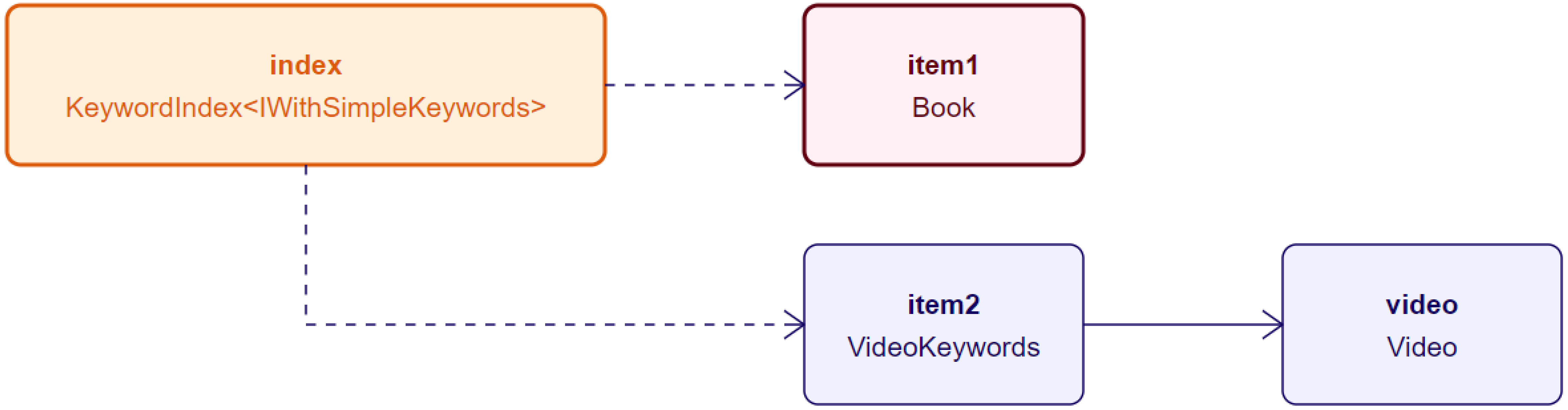
Motivation to Apply the Adapter Pattern



Common Adapter Implementation

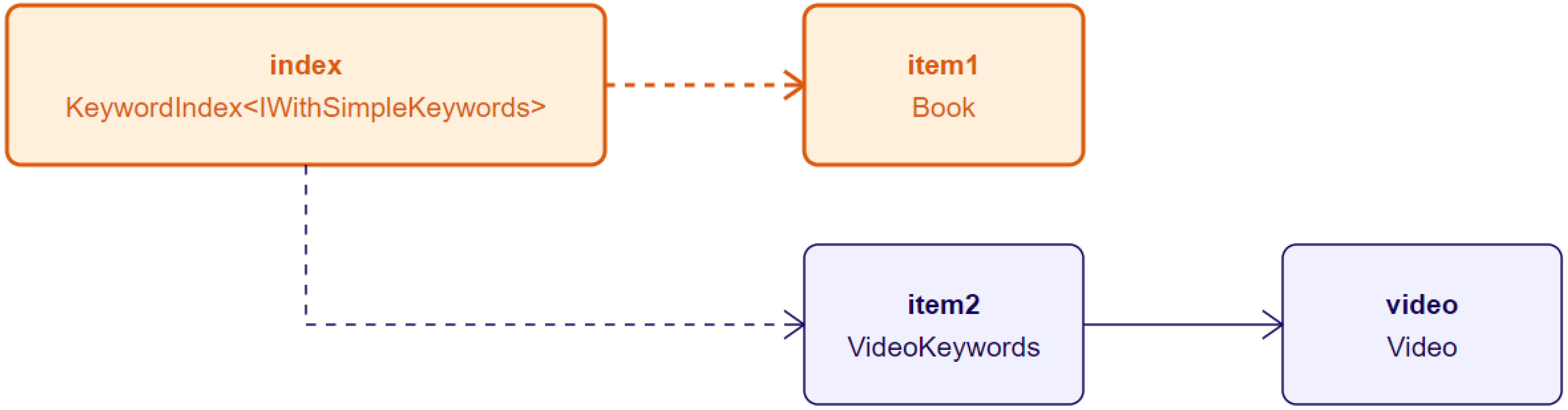


Common Adapter Implementation



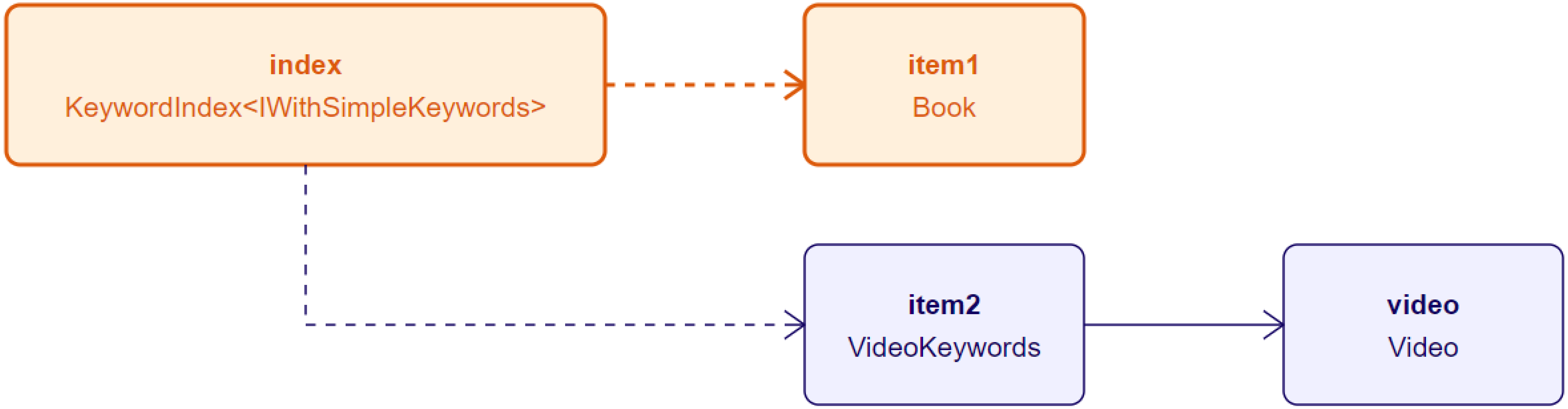
External code calls index.Add(item1)

Common Adapter Implementation



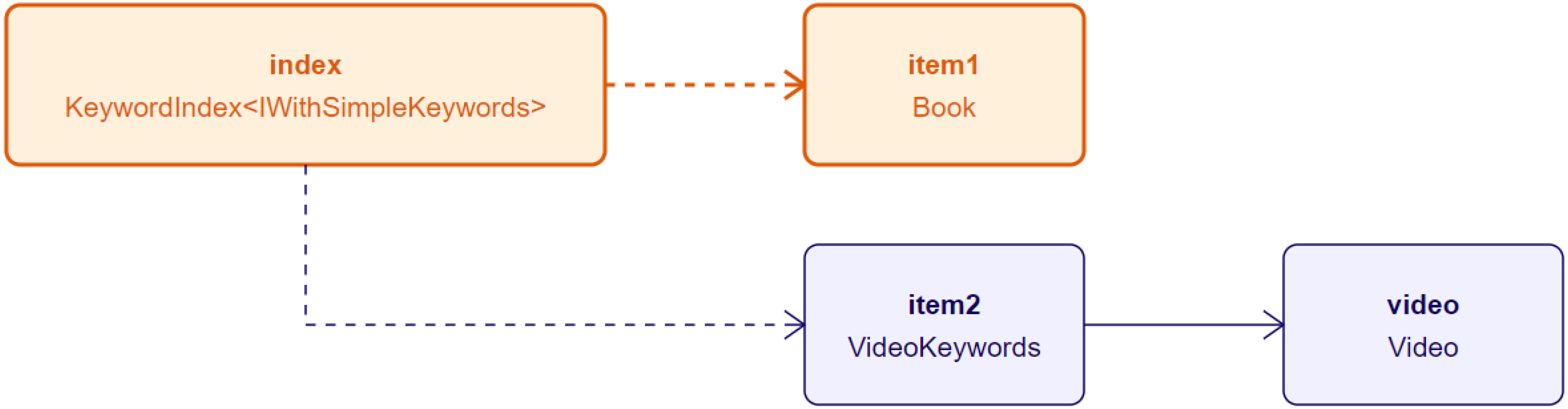
index calls item1.Keywords

Common Adapter Implementation



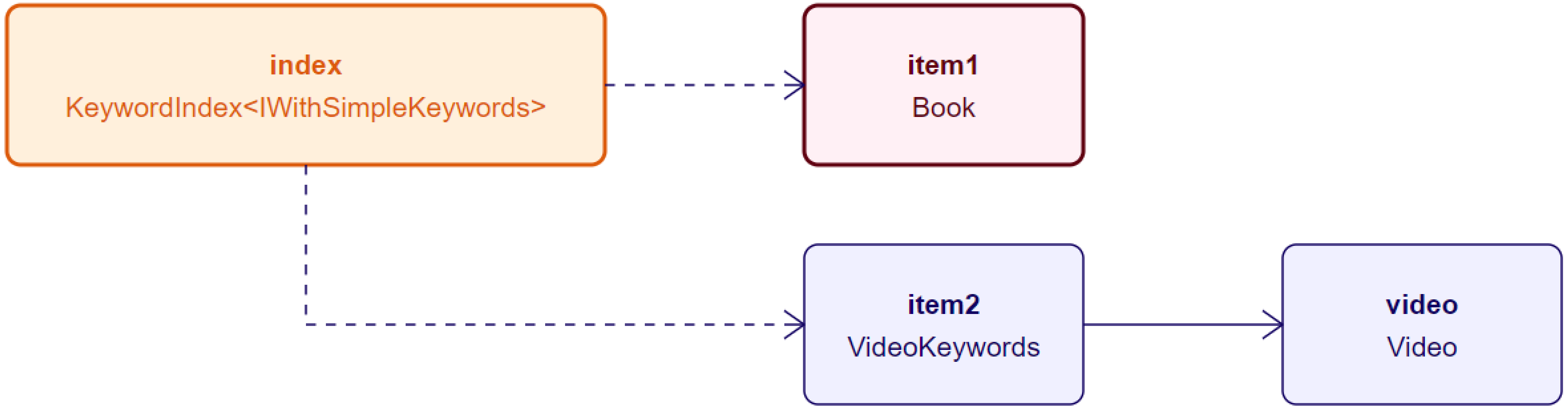
index calls item1.Keywords

Common Adapter Implementation

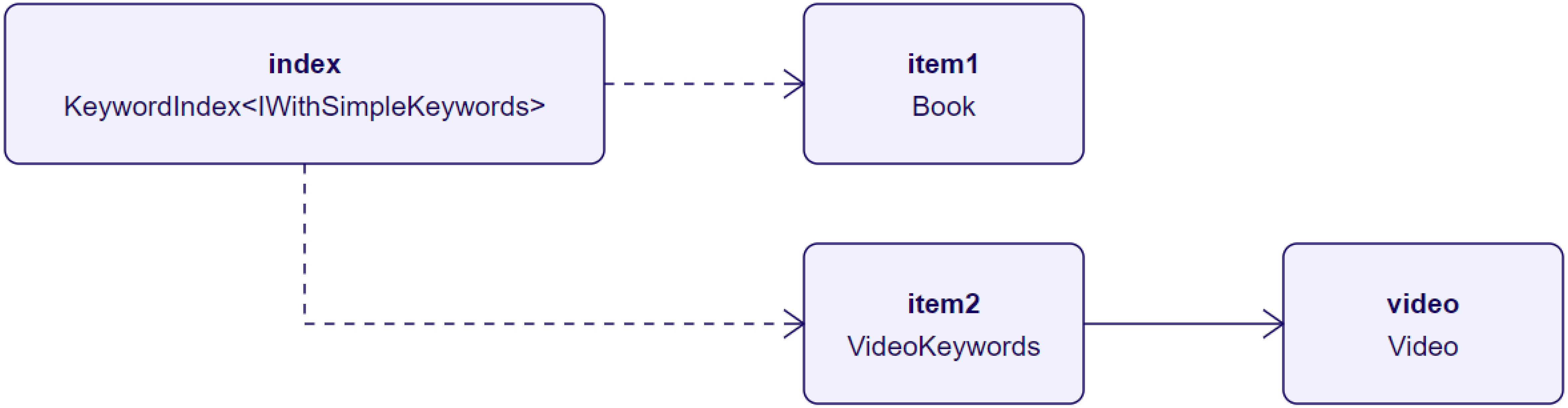


item1.Keywords returns ["long", "boring"]

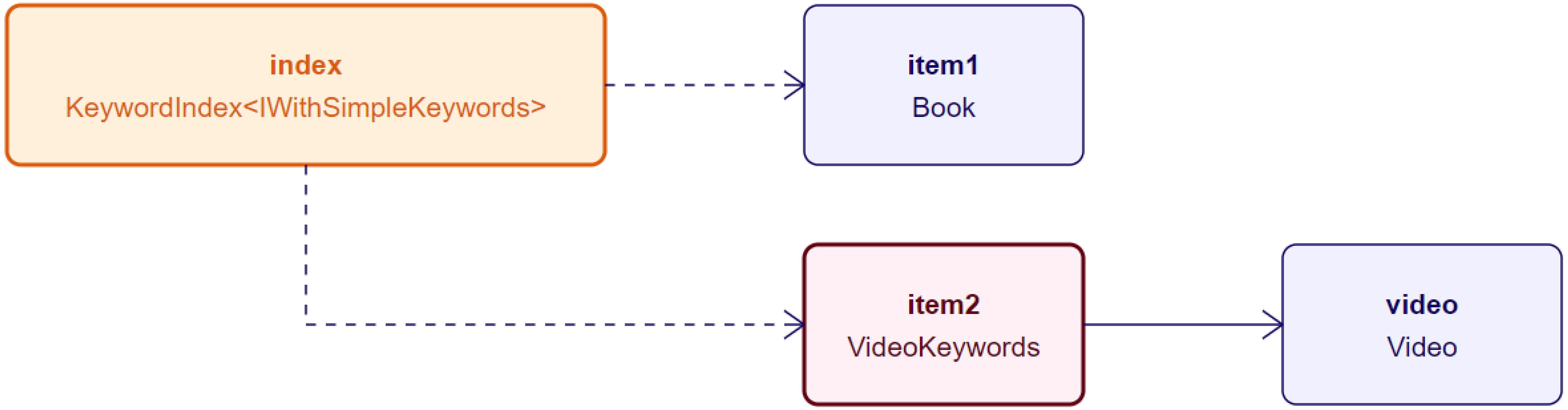
Common Adapter Implementation



Common Adapter Implementation

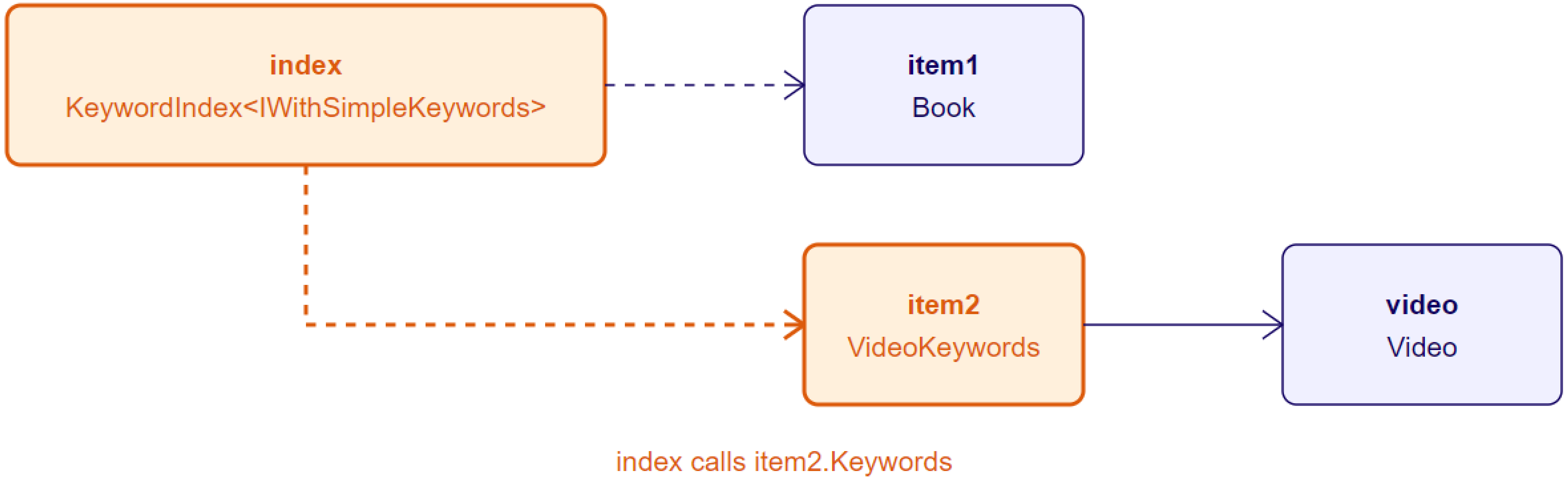


Common Adapter Implementation

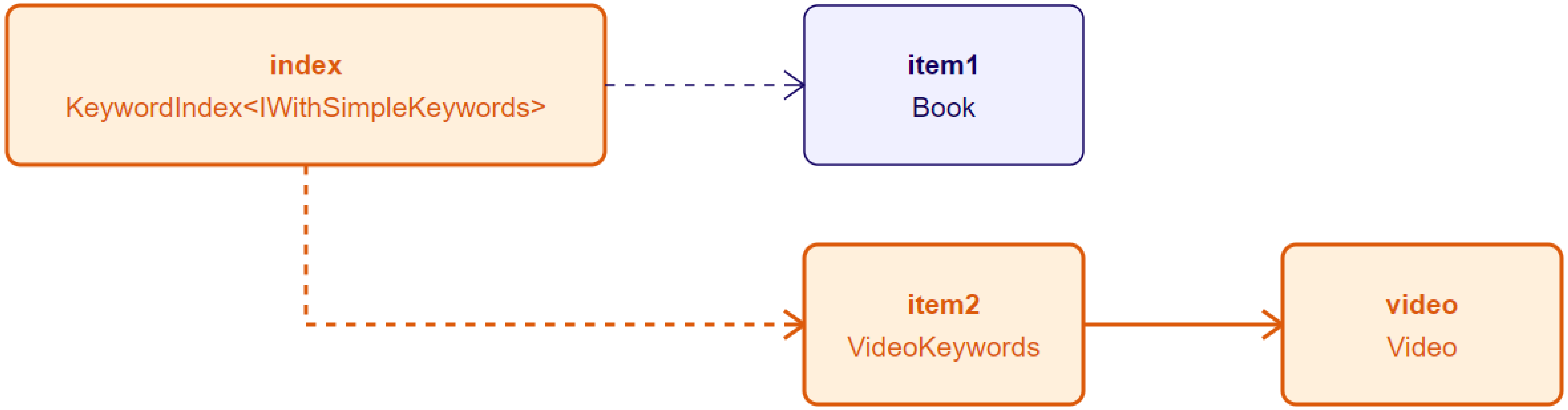


External code calls index.Add(item2)

Common Adapter Implementation

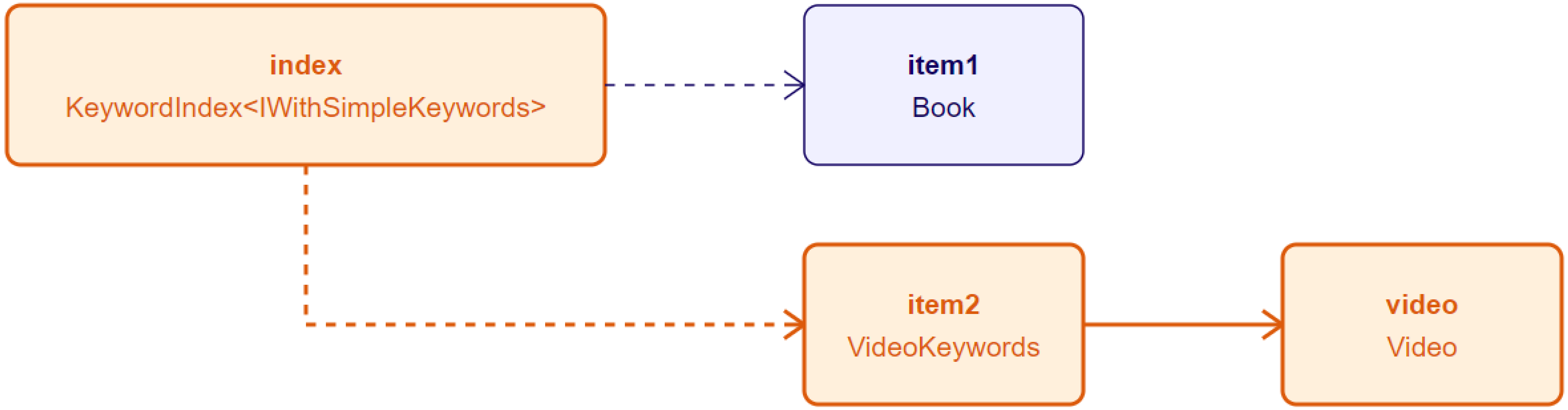


Common Adapter Implementation



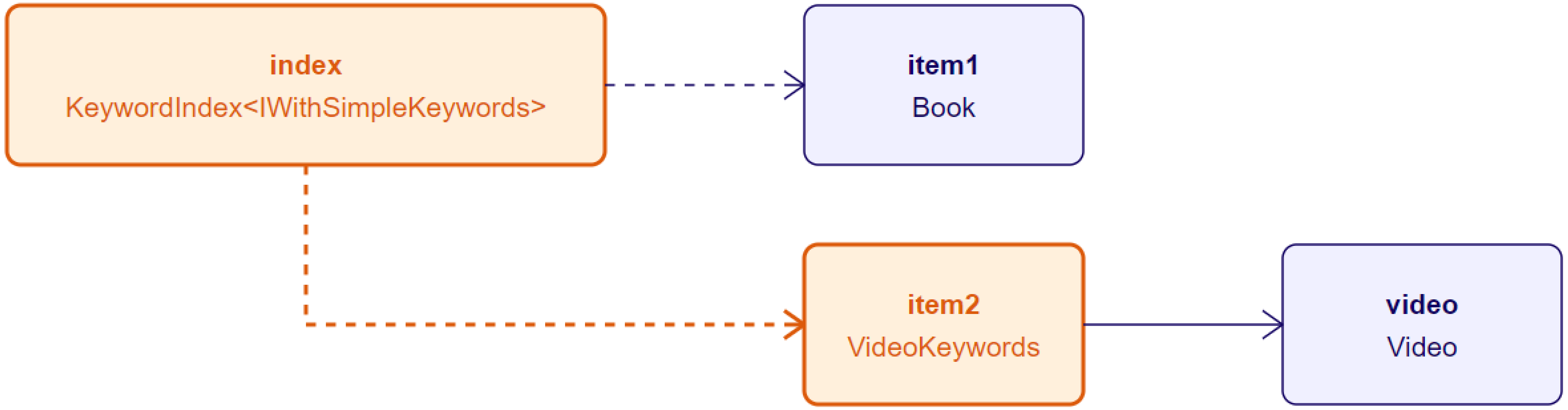
item2 calls video.Title

Common Adapter Implementation

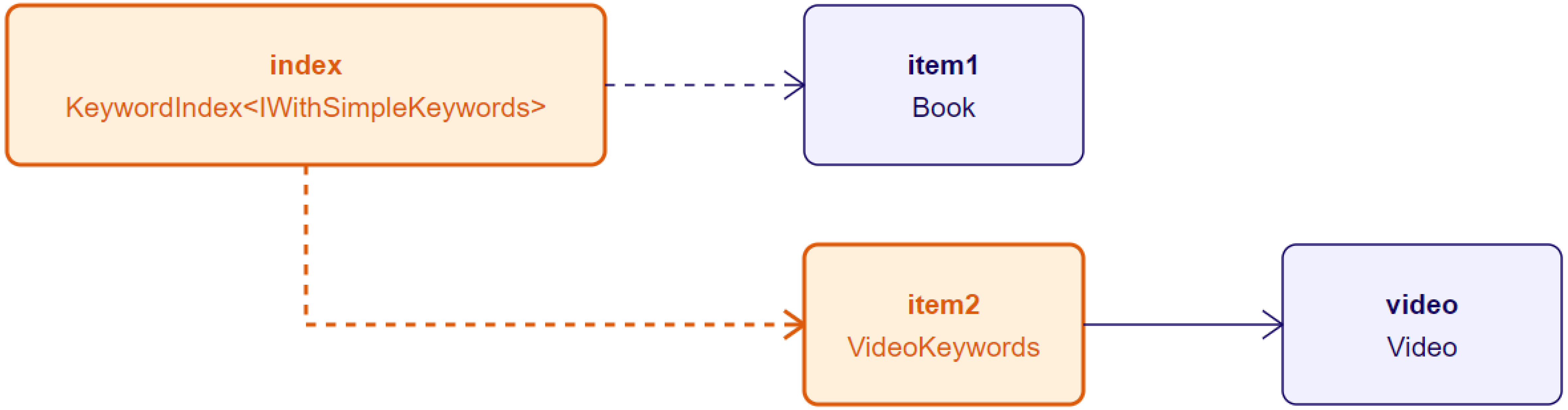


video.Title returns "The Longest Video Ever"

Common Adapter Implementation

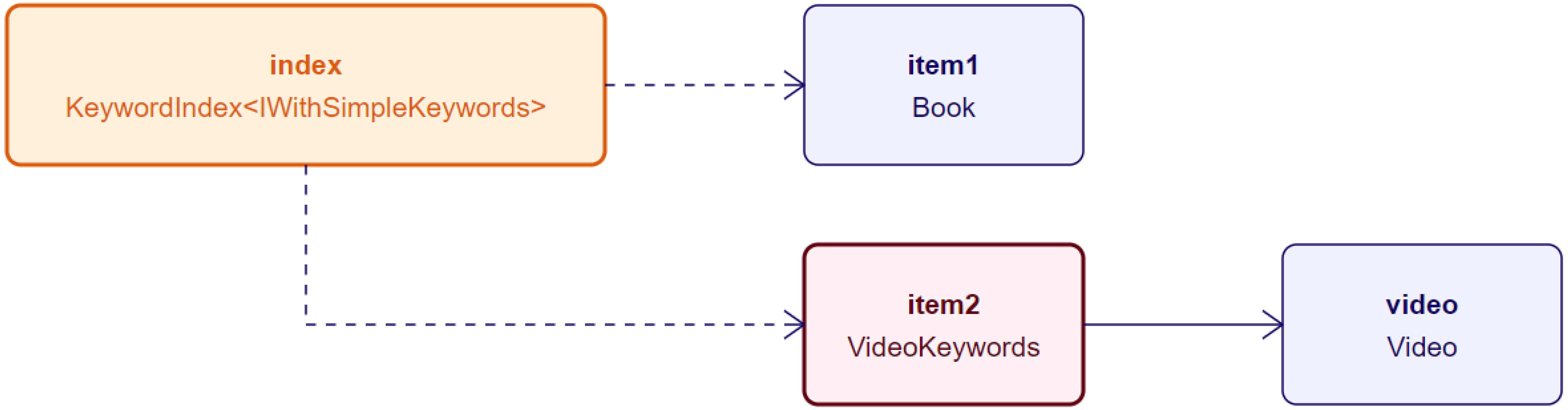


Common Adapter Implementation

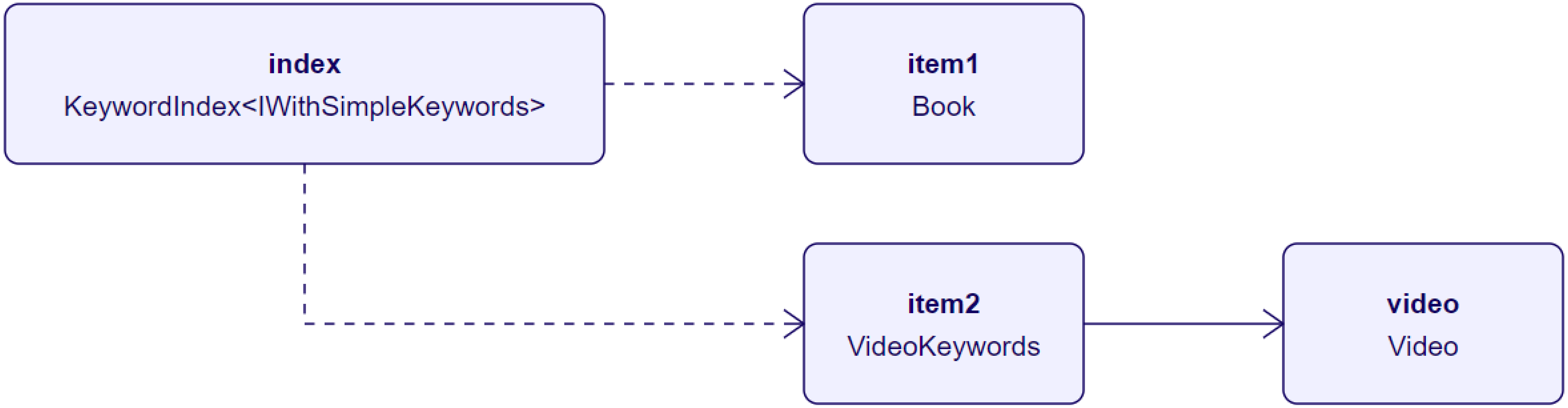


item2.Keywords returns ["longest", "video", "ever"]

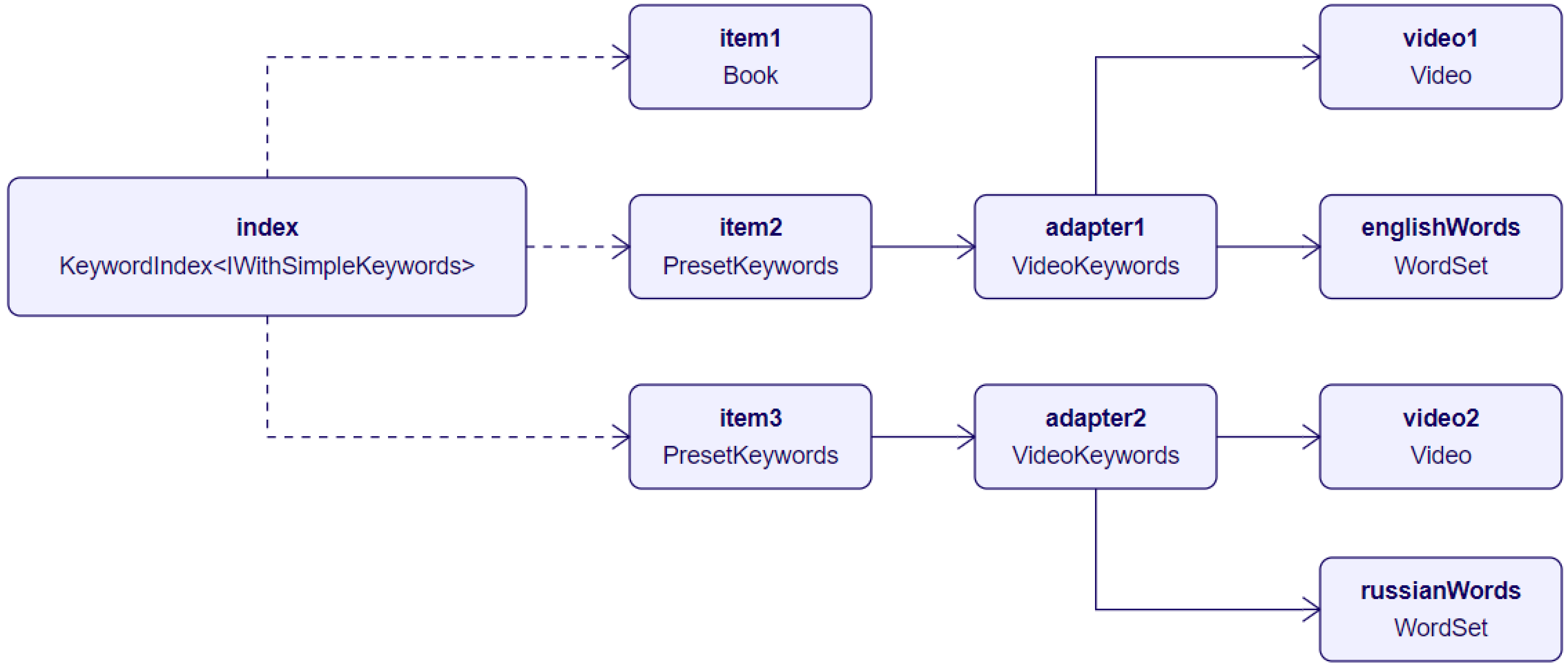
Common Adapter Implementation



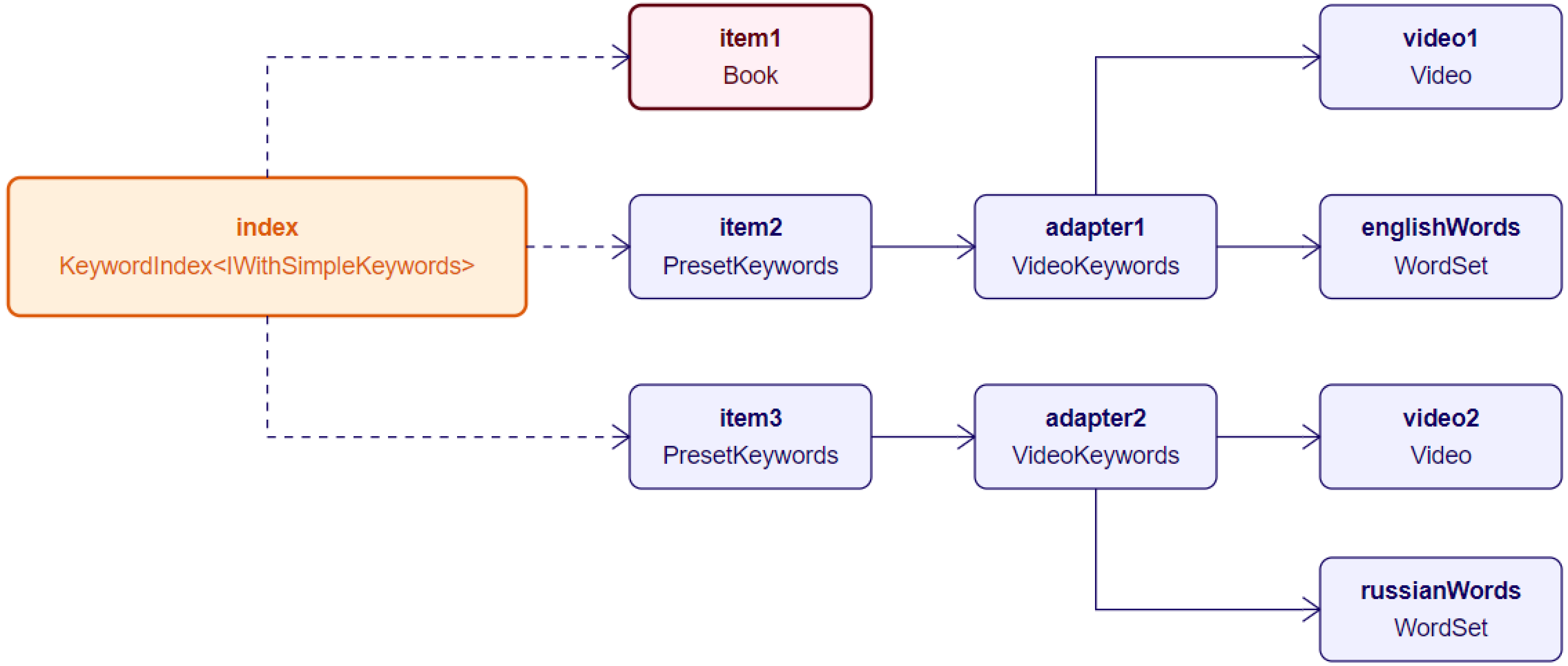
Common Adapter Implementation



Decorating the Adapter

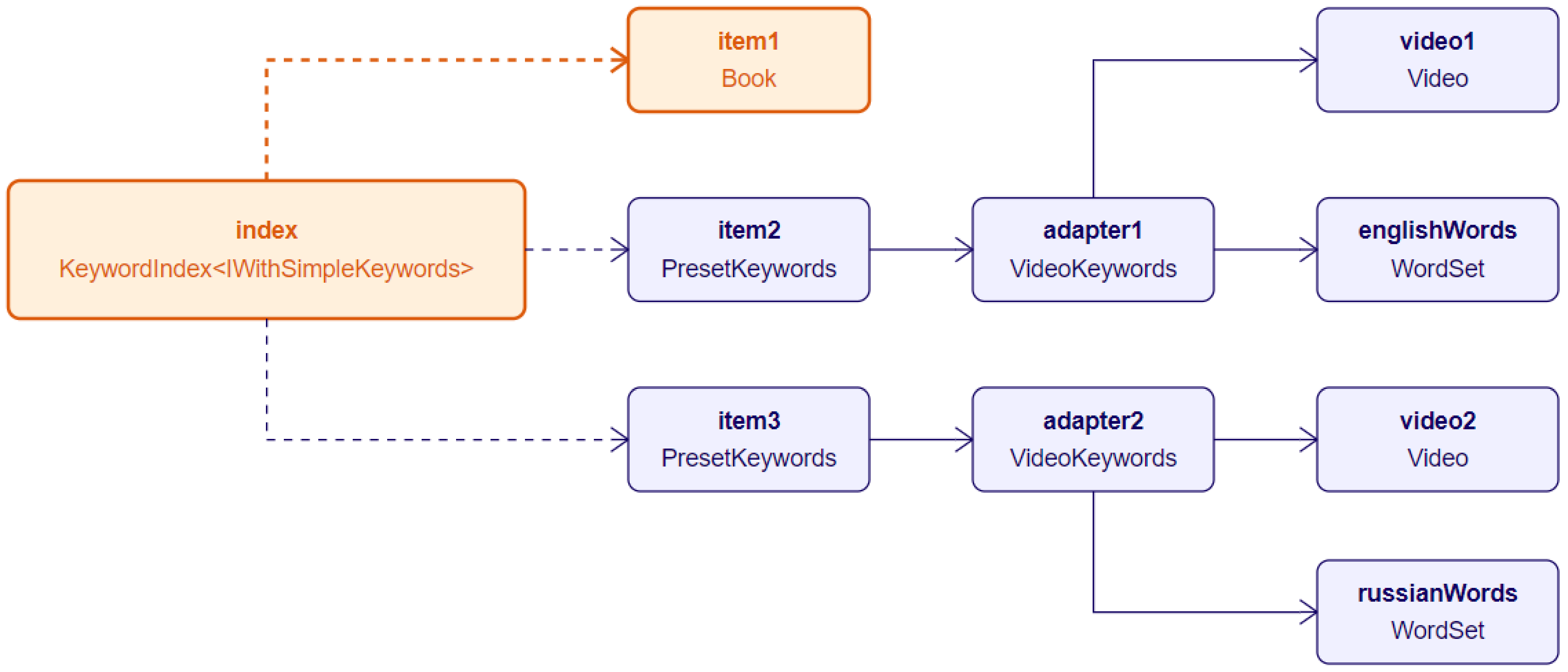


Decorating the Adapter



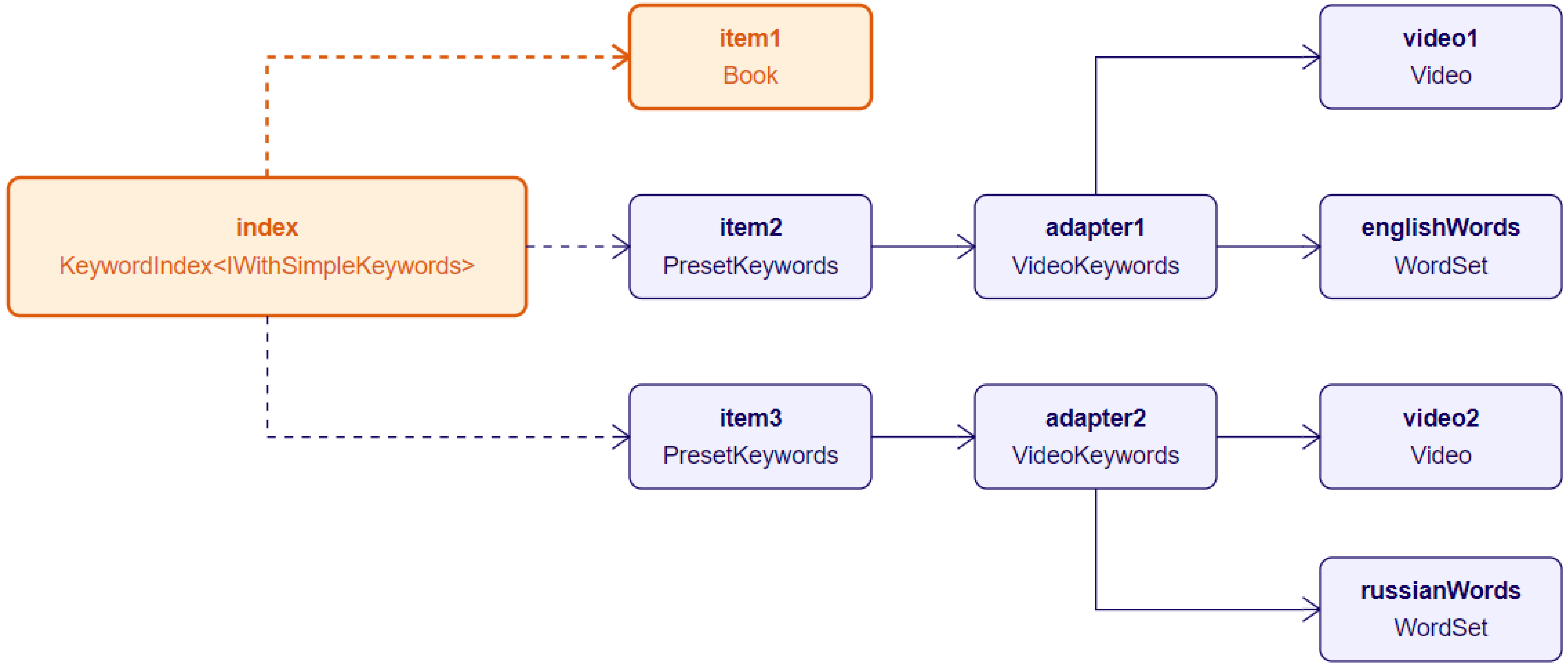
External code calls index.Add(item1)

Decorating the Adapter



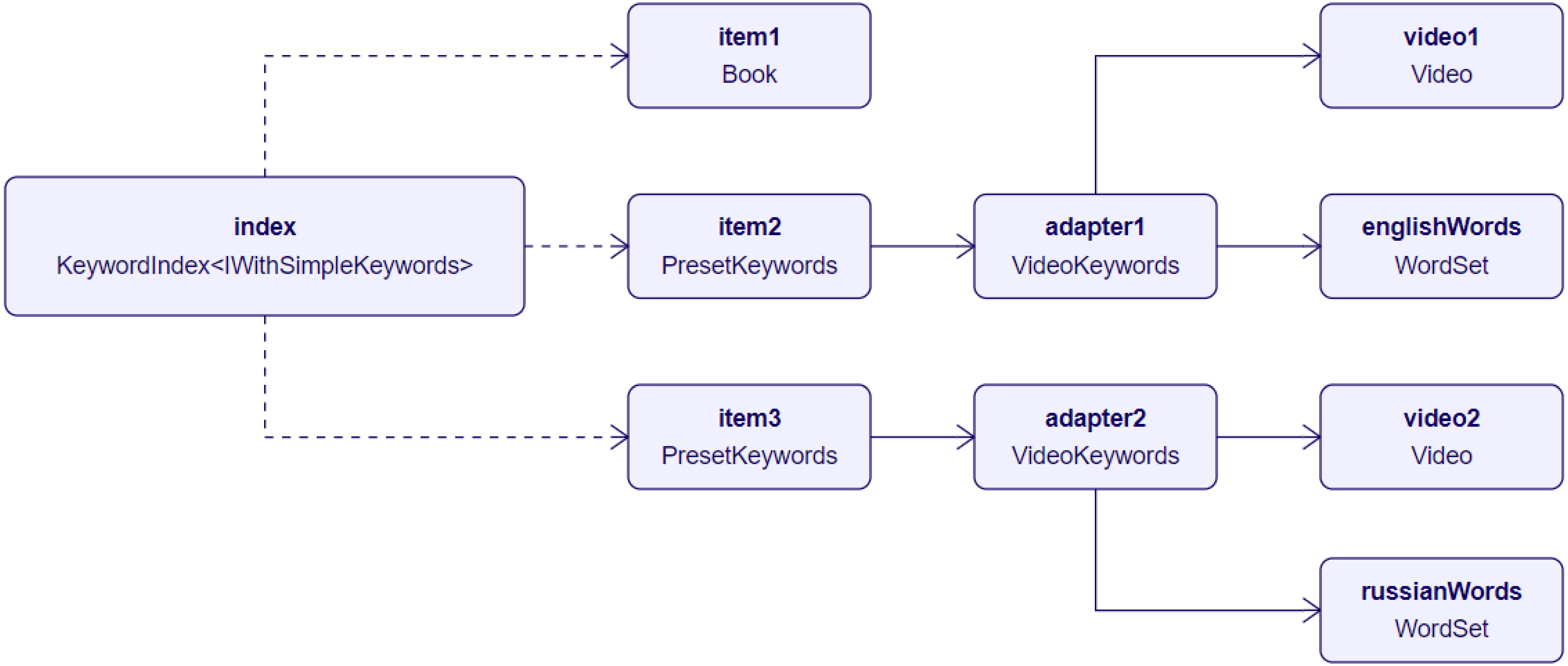
index calls item1.Keywords

Decorating the Adapter

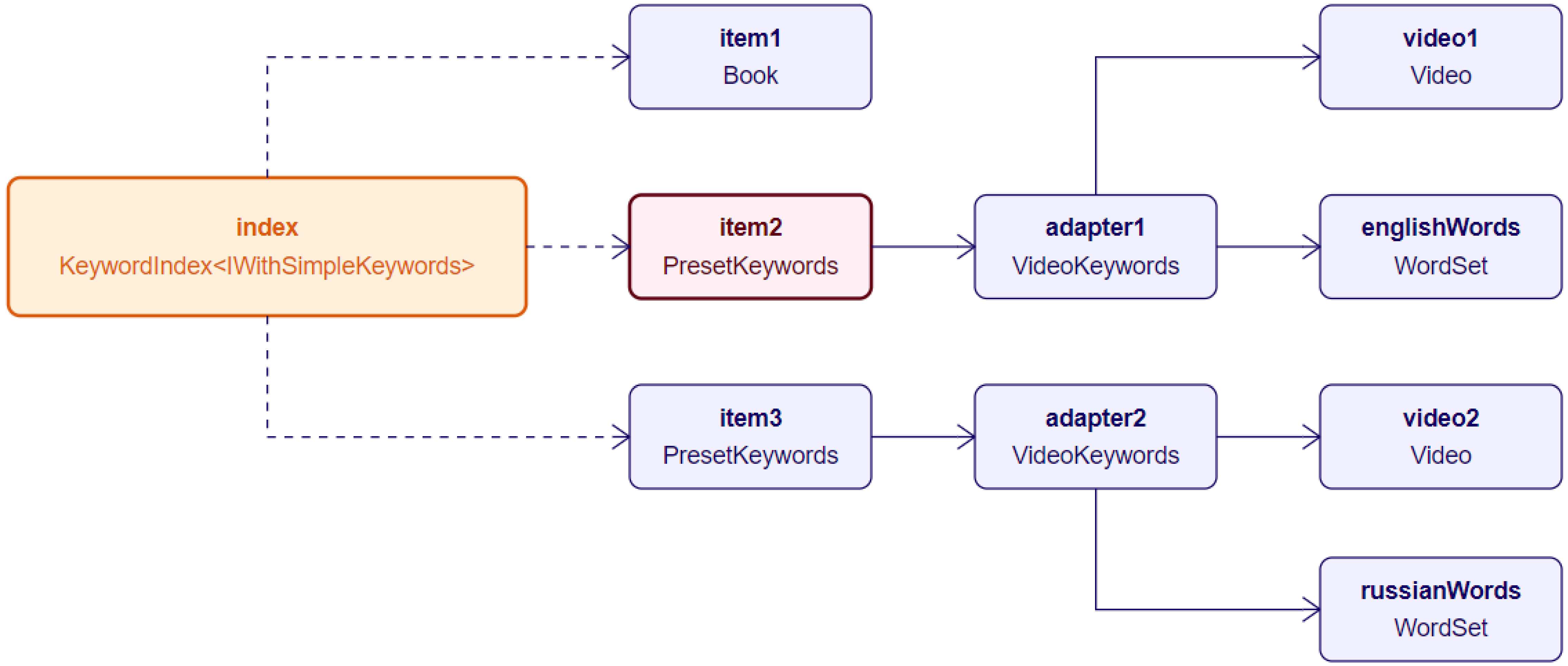


item1.Keywords returns ["boring", "long"]

Decorating the Adapter

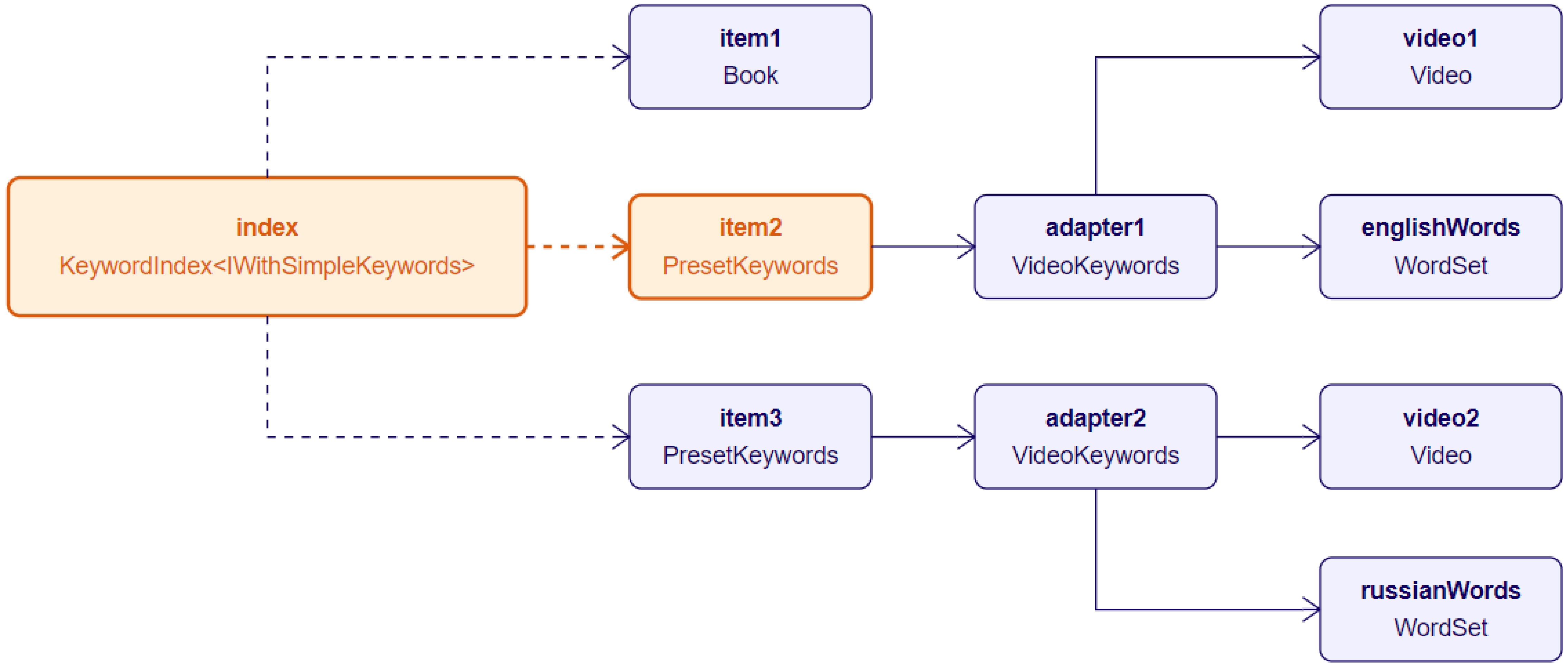


Decorating the Adapter



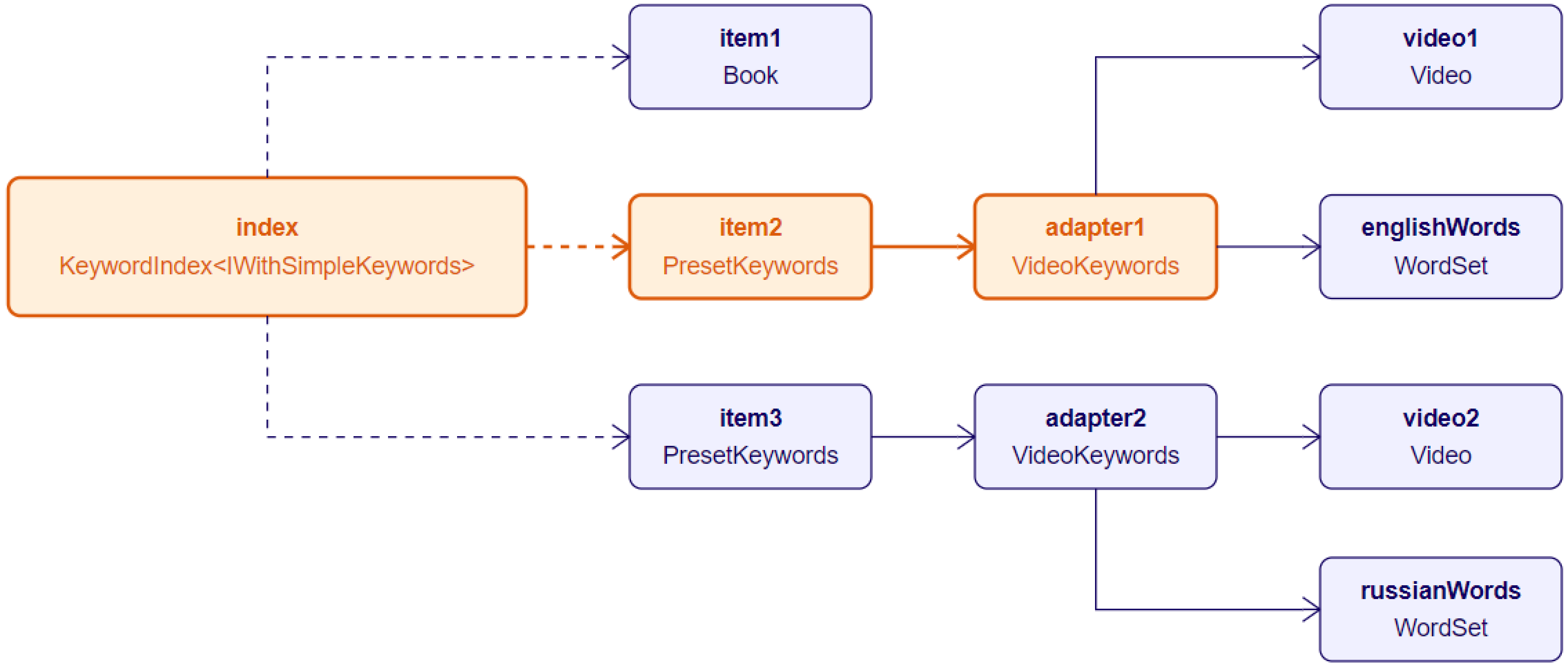
External code calls index.Add(item2)

Decorating the Adapter



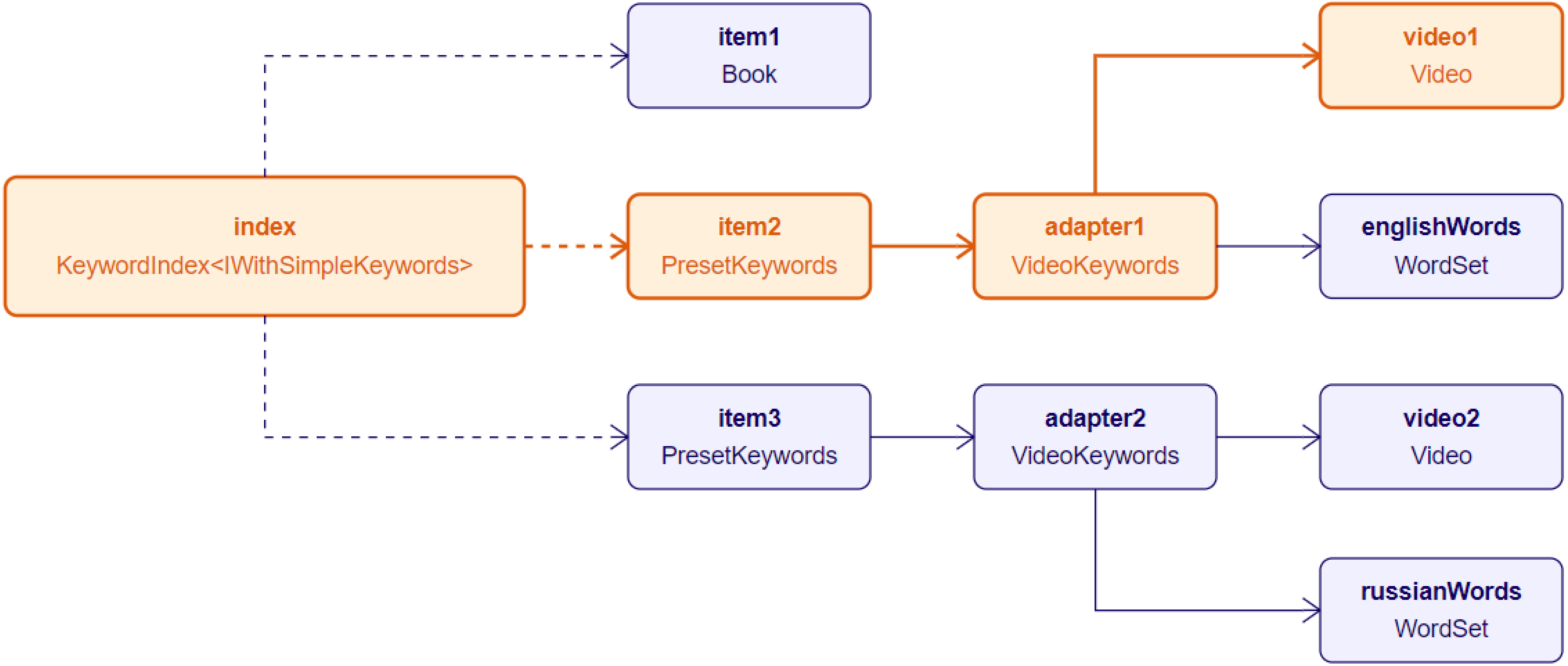
index calls item2.Keywords

Decorating the Adapter



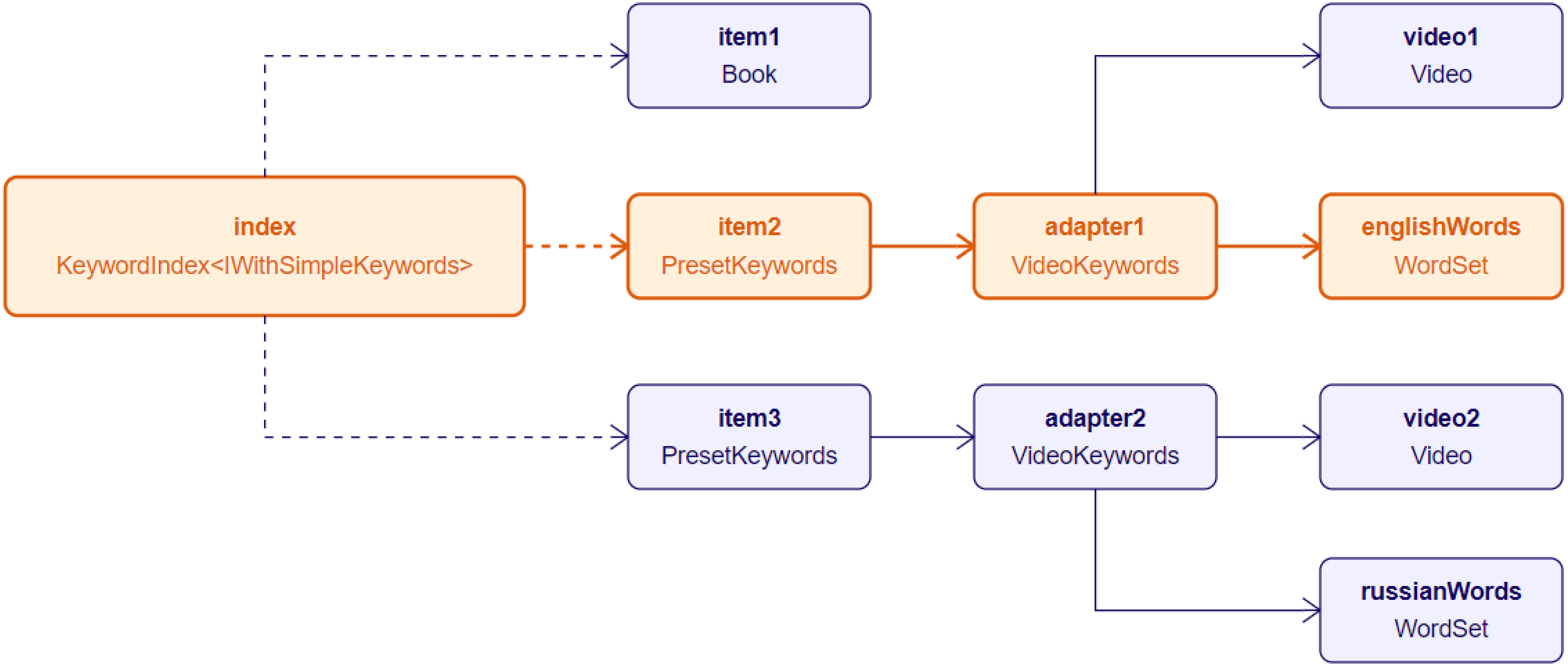
item2 calls adapter1.Keywords

Decorating the Adapter



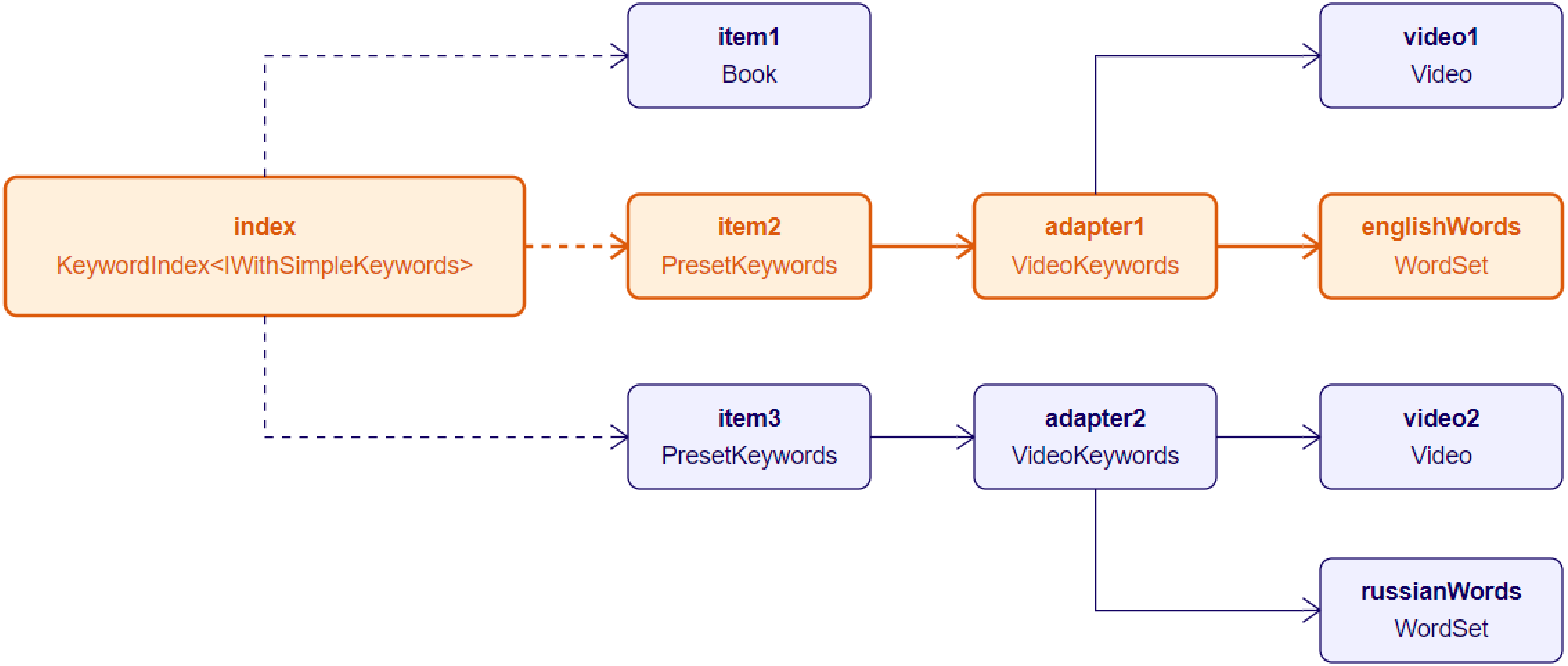
video1.Title returns "Making the Long, Long Ad"

Decorating the Adapter



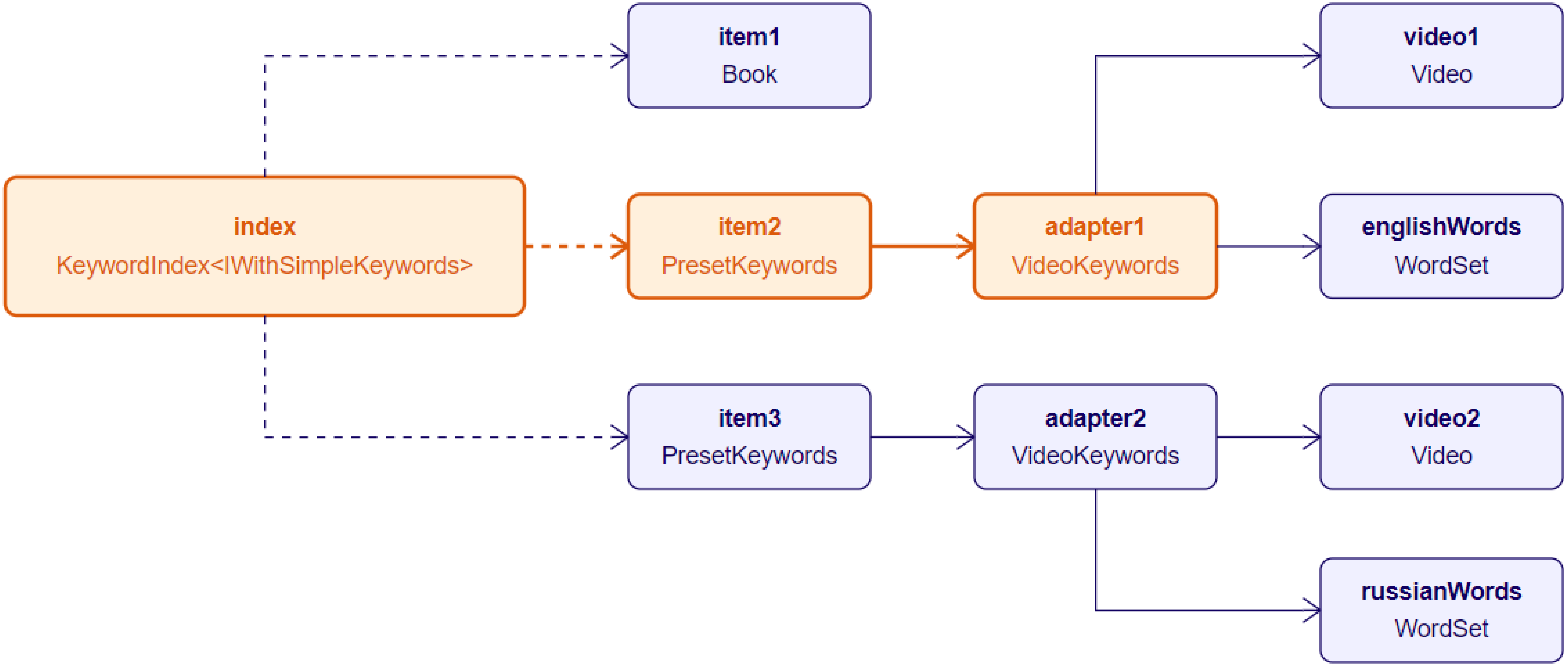
adapter1 calls englishWords.AddText("Making the Long, Long Ad")

Decorating the Adapter



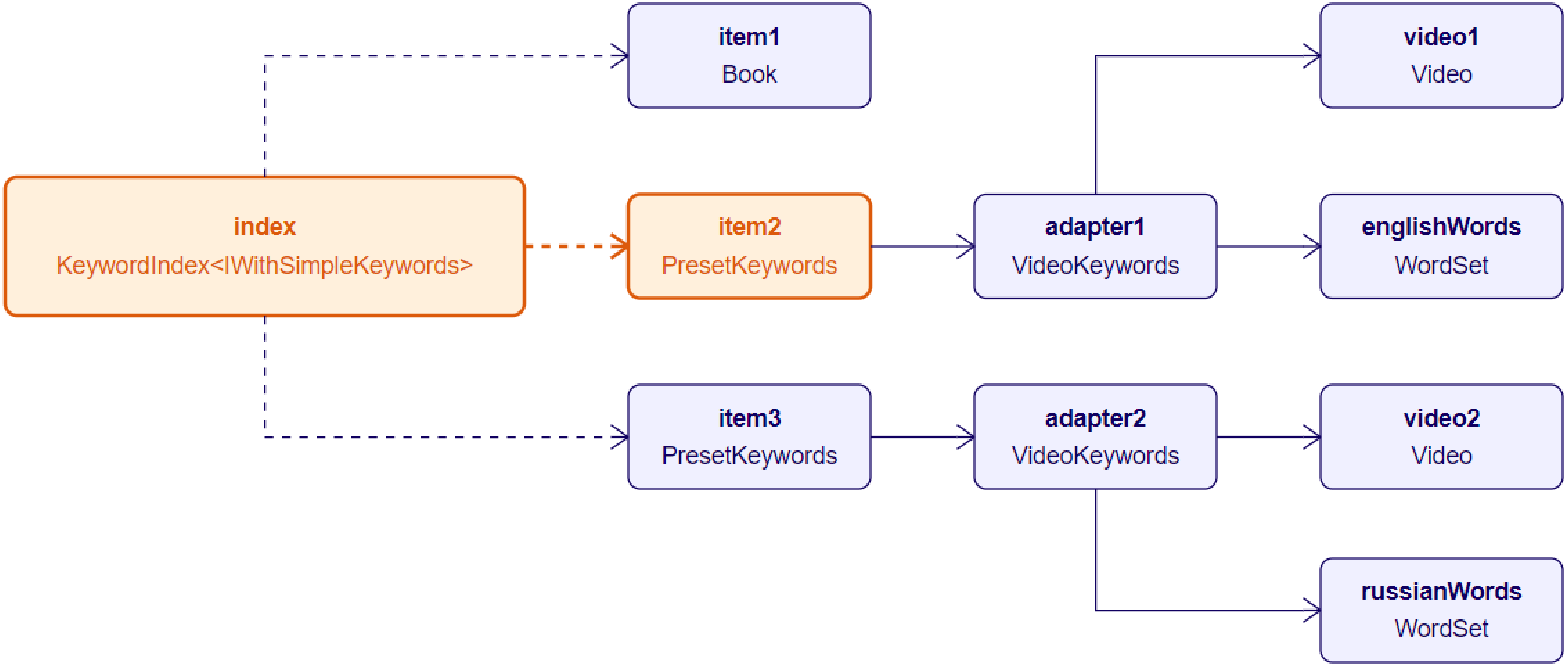
englishWords.AddText() returns { Ad, Making, Long }

Decorating the Adapter



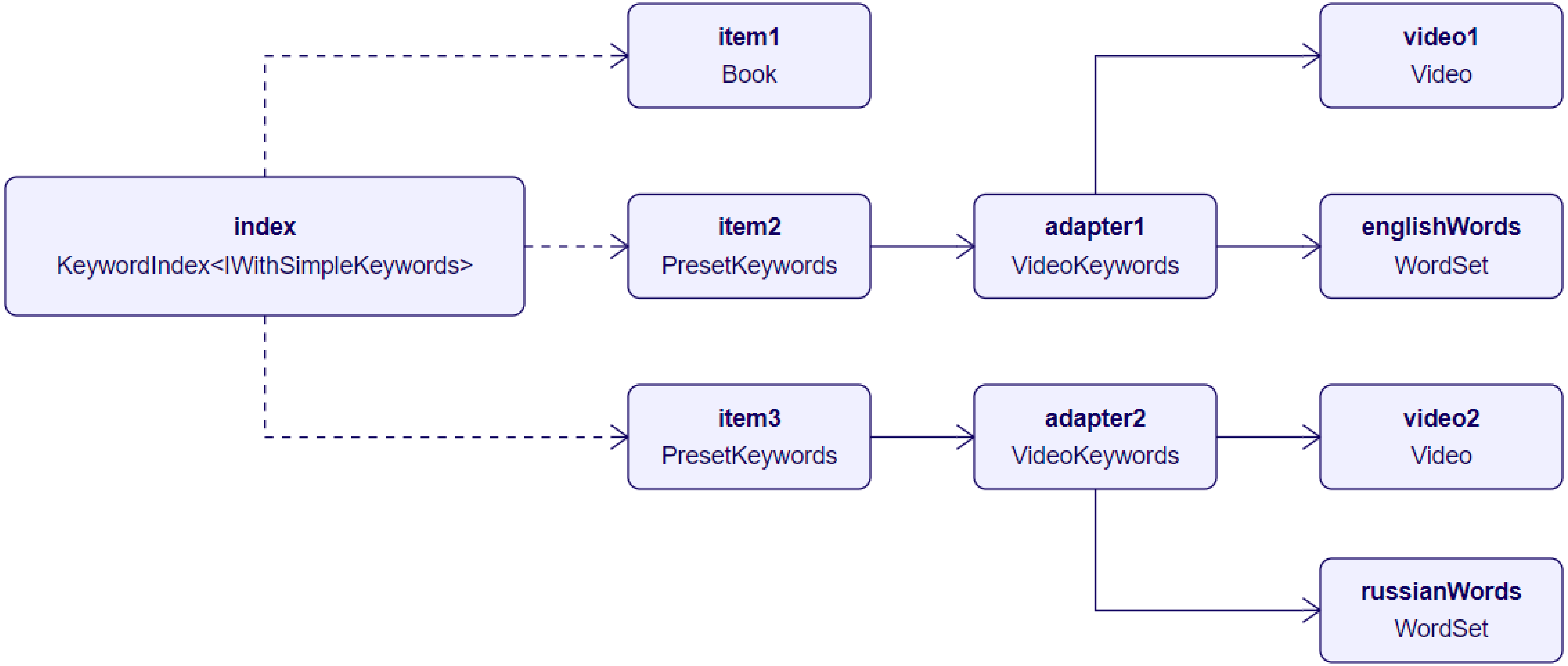
adapter1.Keywords returns { Ad, Making, Long }

Decorating the Adapter

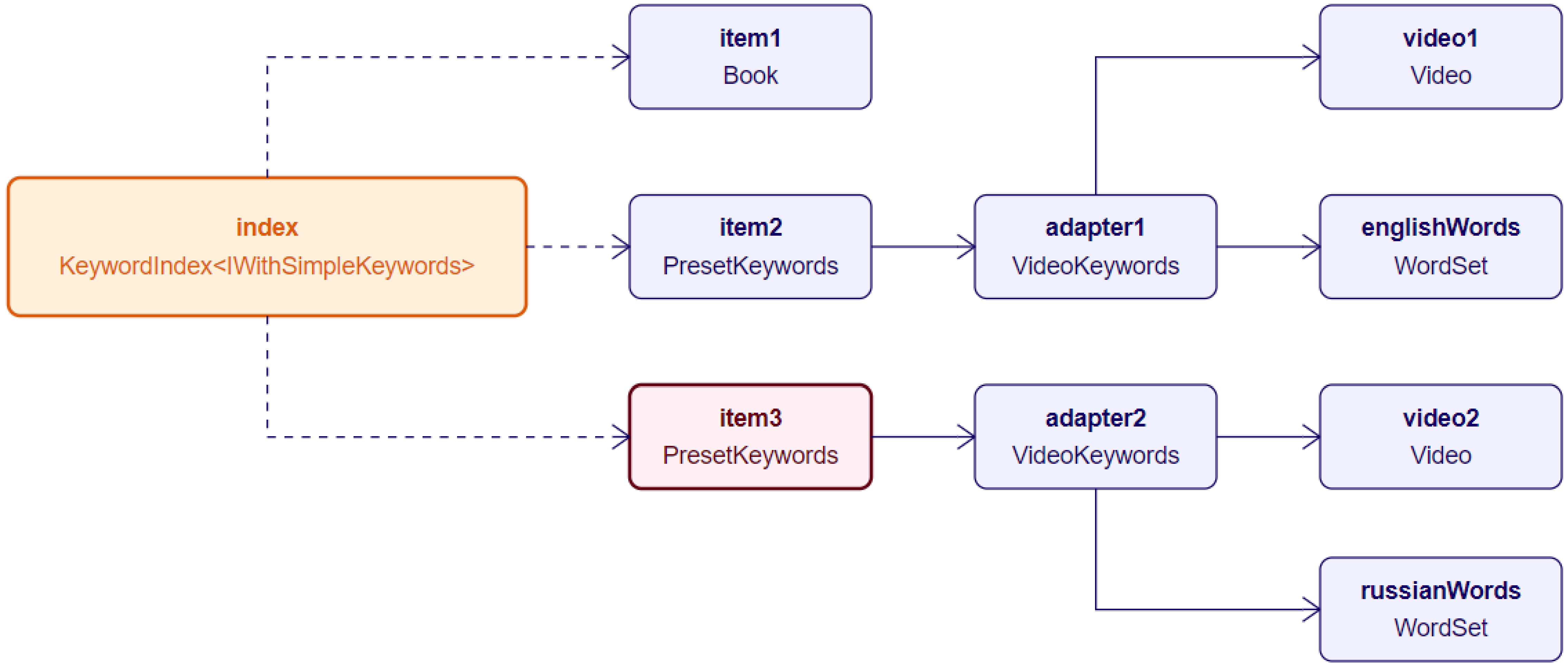


item2.Keywords returns ["Ad", "Making", "Long"]

Decorating the Adapter

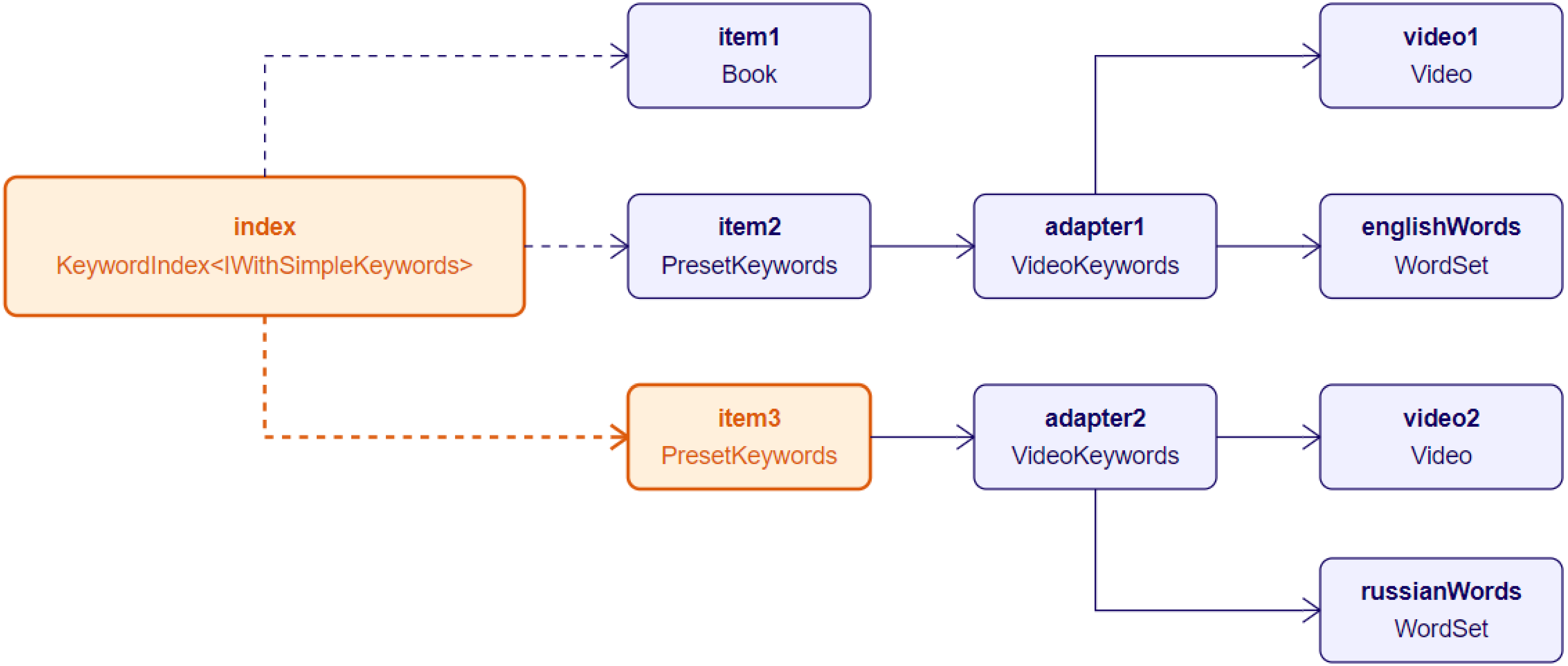


Decorating the Adapter



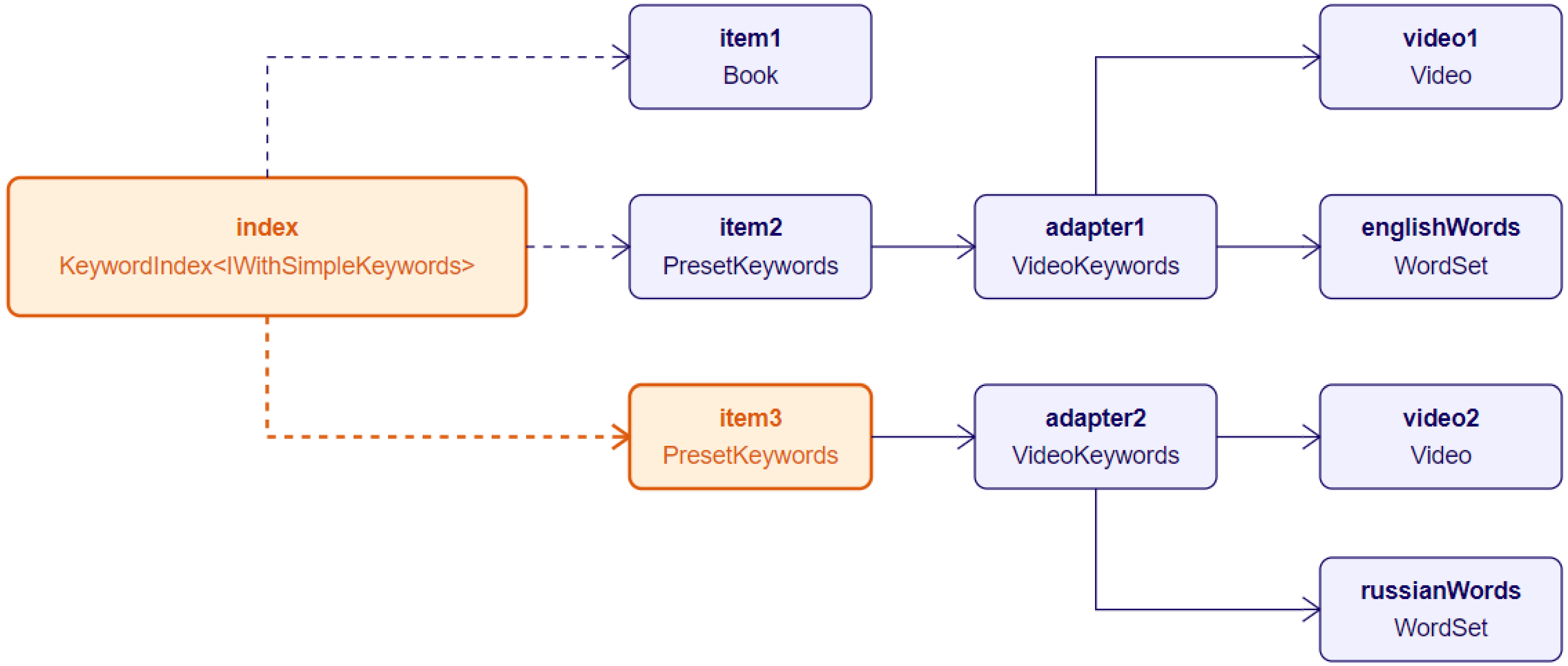
External code calls index.Add(item3)

Decorating the Adapter



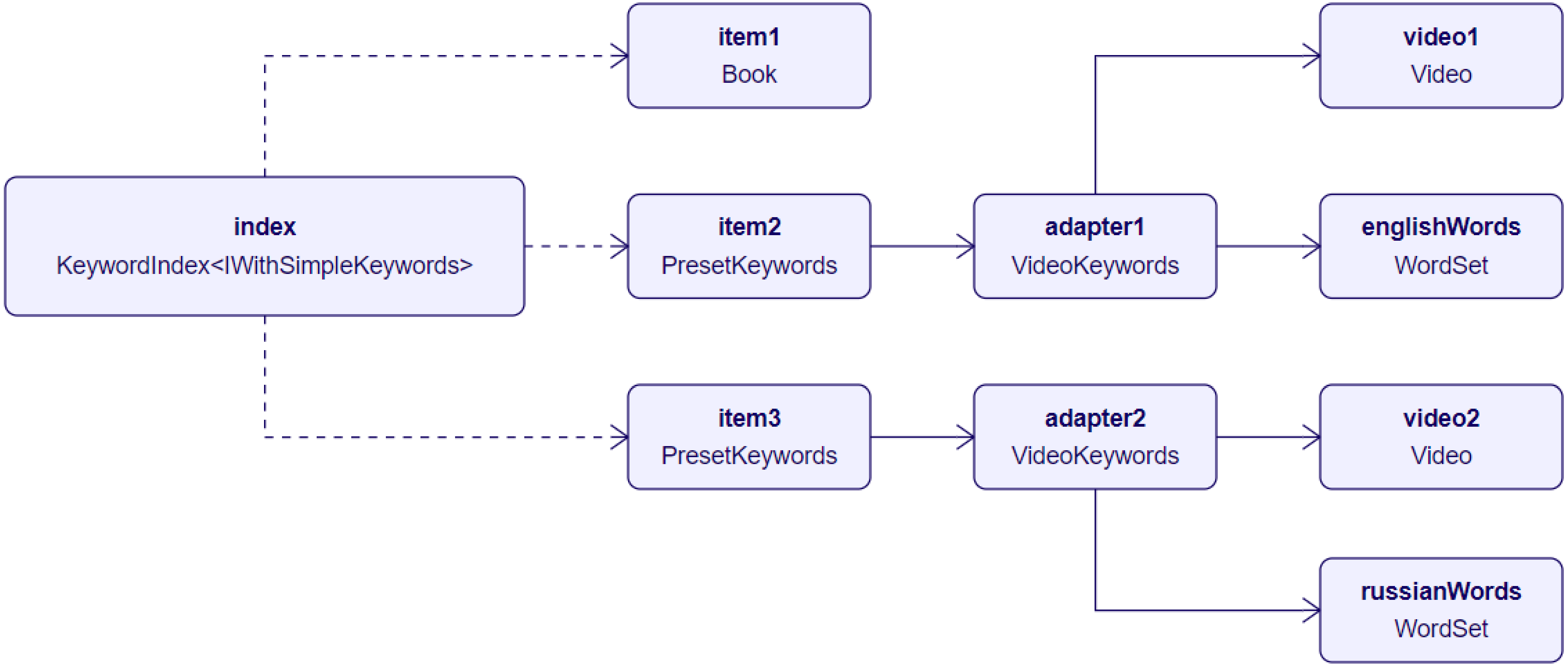
index calls item3.Keywords

Decorating the Adapter



item3.Keywords returns ["karamazov", "brothers"]

Decorating the Adapter



Summary

The Adapter pattern

- Similar to the Decorator pattern
- Adapts an object to a different interface
- Maps outer calls to the wrapped object
- Outer caller talks to the object implementing the desired interface
- Also called the Wrapper pattern

Summary

Motivation to apply the Adapter pattern

- An object with mismatched interface could not be consumed
- Solved by wrapping that object with an adapter
- Adapter uses lightweight mappings to inner calls

Summary

Coping with the bloated adapter

- More and more features added to the adapter
- Adapter can easily become bloated
- Adapter does not suit well complex domain logic
- Split the bloated adapter into cooperating objects
- Move specialized logic out into domain classes

Summary

Combining patterns

- Adapter can easily combine with the Decorator
- Apply the adapter to produce the desired interface
- Then apply one or more decorators to add behavior
- Combined patterns yield smaller and focused classes