SWE 4743:

Object-Oriented Design

Jeff Adkisson

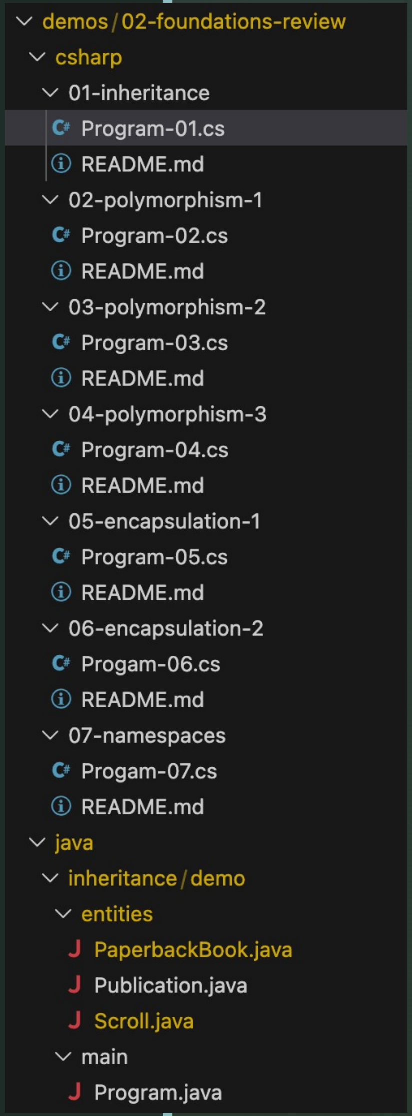
OO Foundations Review

Why Review OO Concepts?

- Students arrive with uneven OO backgrounds
- We need shared terminology
- Design discussions assume these concepts
- The *following* lecture demonstrates how to diagram OO relationships using Mermaid.js

Agenda

- Polymorphism, encapsulation, inheritance, interfaces
- Access modifiers and visibility boundaries
- Classes, fields, methods, properties
- Primitive, abstract, and concrete types
- Namespaces, modules, and basic organizational structure



```
demos / 02-foundations-review
  csharp
    01-inheritance
      Program-01.cs
      README.md
    02-polymorphism-1
      Program-02.cs
      README.md
    03-polymorphism-2
      Program-03.cs
      README.md
    04-polymorphism-3
      Program-04.cs
      README.md
    05-encapsulation-1
      Program-05.cs
      README.md
    06-encapsulation-2
      Progam-06.cs
      README.md
    07-namespaces
      Progam-07.cs
      README.md
  java
    inheritance / demo
      entities
        PaperbackBook.java
        Publication.java
        Scroll.java
      main
        Program.java
```

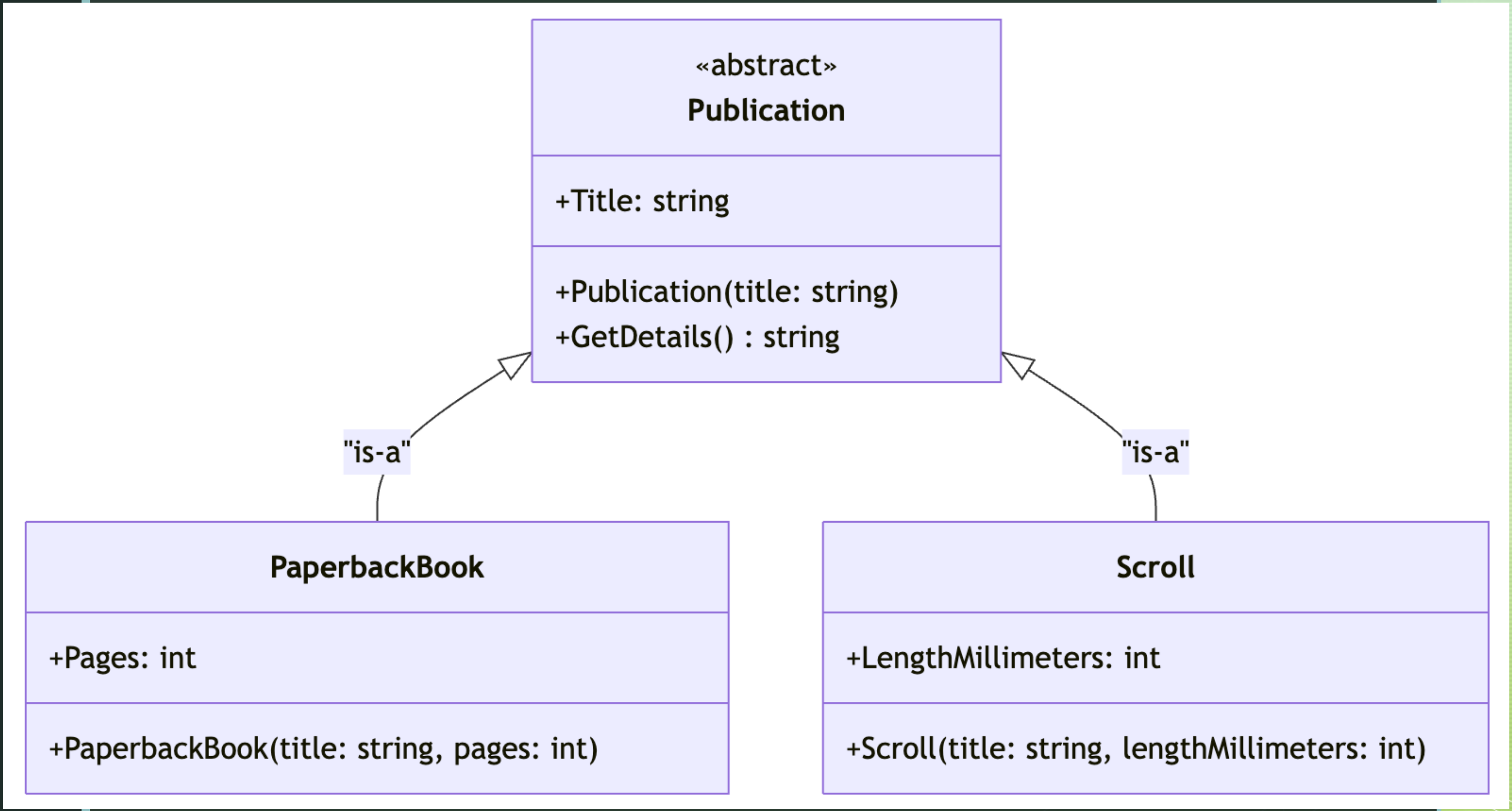
Demo Code

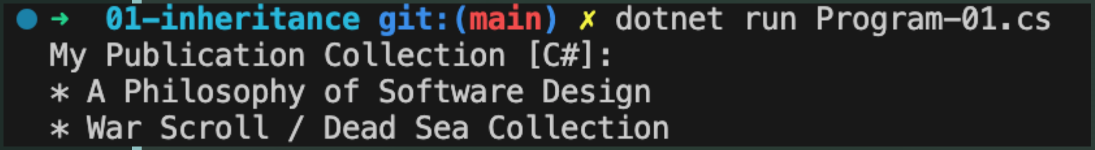
- Demo code is located in the course repository under **demos/02-foundations-review**
- Most demo code is in C#.
- There is one version of Progam-01.cs in Java to demonstrate the difference in how Java handles namespaces (must be in separate folders and files).

Inheritance

- Allows one class to reuse another class’s behavior
- Represents an "is-a" relationship
- Creates a relationship between types

Inheritance Program-01.cs





```
● → 01-inheritance git:(main) ✗ dotnet run Program-01.cs
  My Publication Collection [C#]:
  * A Philosophy of Software Design
  * War Scroll / Dead Sea Collection
```

Inheritance
Program-01.cs

Architecture

- Defines an **abstract base class (Publication)** that represents a general domain concept.
- Concrete subclasses (PaperbackBook, Scroll) **inherit shared state** from the base class.
- The base class provides **non-abstract shared behavior** (GetDetails) that subclasses may override.
- Client code operates on a **collection of the base type**.

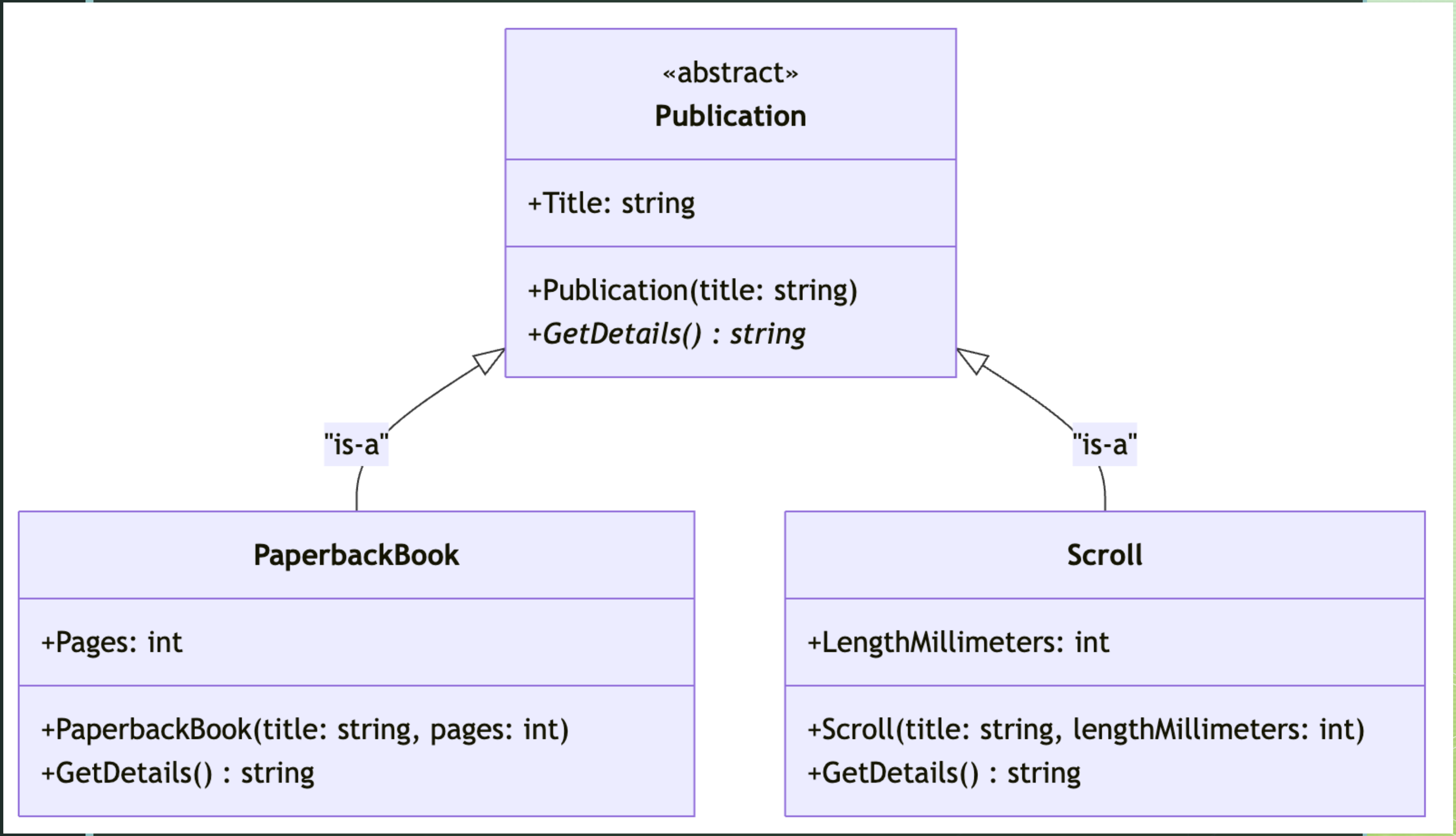
OO Concepts Illustrated

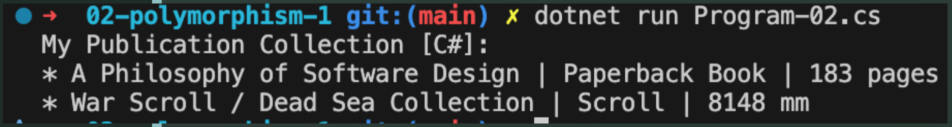
- Abstract classes
- Inheritance
- Code reuse through shared base implementation

Polymorphism

- Code works with many related types
- Uses a base type reference
- Enables flexible behavior
- There are various polymorphic techniques
  - Abstract classes
  - Interfaces
  - Abstract methods (children must implement)
  - Virtual methods (descendants *can* override parent implementation)

Polymorphism 1
Program-02.cs





```
● ➜  02-polymorphism-1 git:(main) ✗ dotnet run Program-02.cs
  My Publication Collection [C#]:
  * A Philosophy of Software Design | Paperback Book | 183 pages
  * War Scroll / Dead Sea Collection | Scroll | 8148 mm
```

Polymorphism 1
Program-02.cs

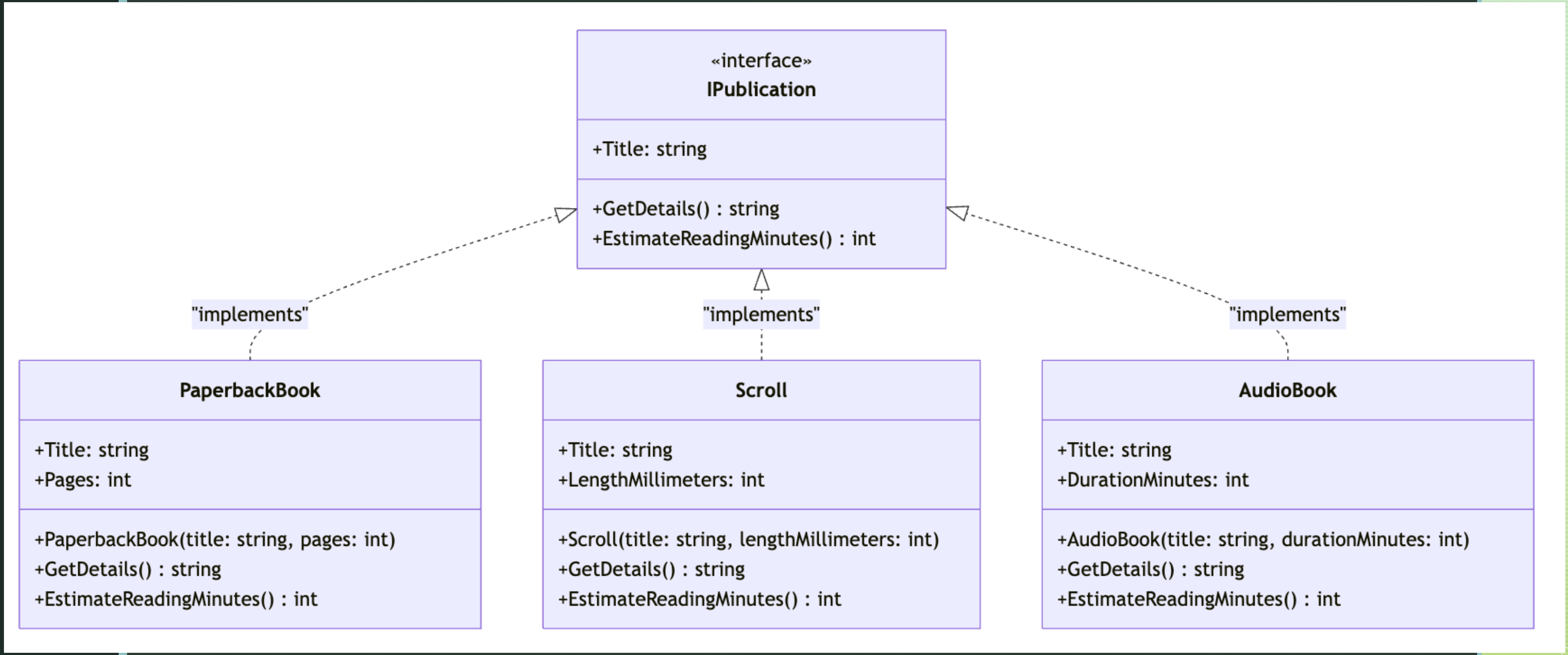
Architecture

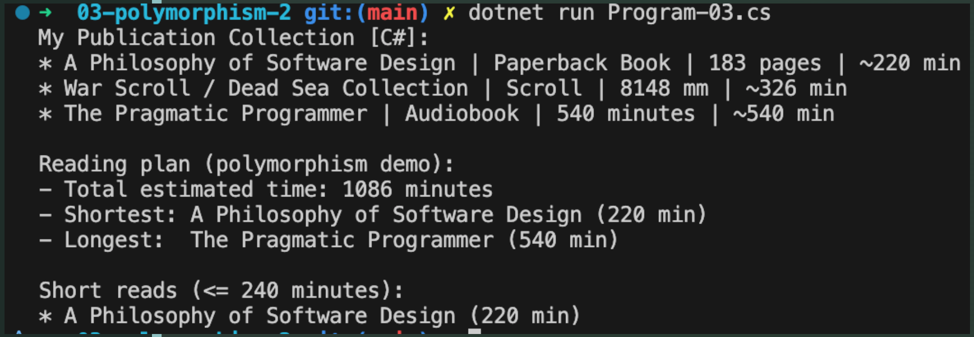
- Evolves the base class by introducing an **abstract method**.
- Subclasses are now **required to implement behavior** specific to their type.
- Polymorphism is exercised by calling the abstract method through a **base-class reference**.
- Shared data remains in the base class; behavior diverges in subclasses.

OO Concepts Illustrated

- Polymorphism
- Abstract methods
- Method overriding
- Dynamic dispatch

Polymorphism 2
Program-03.cs





```
● → 03-polymorphism-2 git:(main) ✗ dotnet run Program-03.cs
  My Publication Collection [C#]:
  * A Philosophy of Software Design | Paperback Book | 183 pages | ~220 min
  * War Scroll / Dead Sea Collection | Scroll | 8148 mm | ~326 min
  * The Pragmatic Programmer | Audiobook | 540 minutes | ~540 min

  Reading plan (polymorphism demo):
  - Total estimated time: 1086 minutes
  - Shortest: A Philosophy of Software Design (220 min)
  - Longest:  The Pragmatic Programmer (540 min)

  Short reads (<= 240 minutes):
  * A Philosophy of Software Design (220 min)
```

Polymorphism 2
Program-03.cs

Architecture

- Introduces an **interface (IPublication)** to define behavioral contracts.
- Abstract base class is no longer responsible for polymorphic behavior.
- Concrete classes implement the interface directly.
- Base class (if present) is now focused on **shared state only**.

OO Concepts Illustrated

- Interfaces
- Interface-based polymorphism
- Separation of concerns

Interfaces

- Define what a class can do
- Do not contain implementation details
- Allow multiple implementations

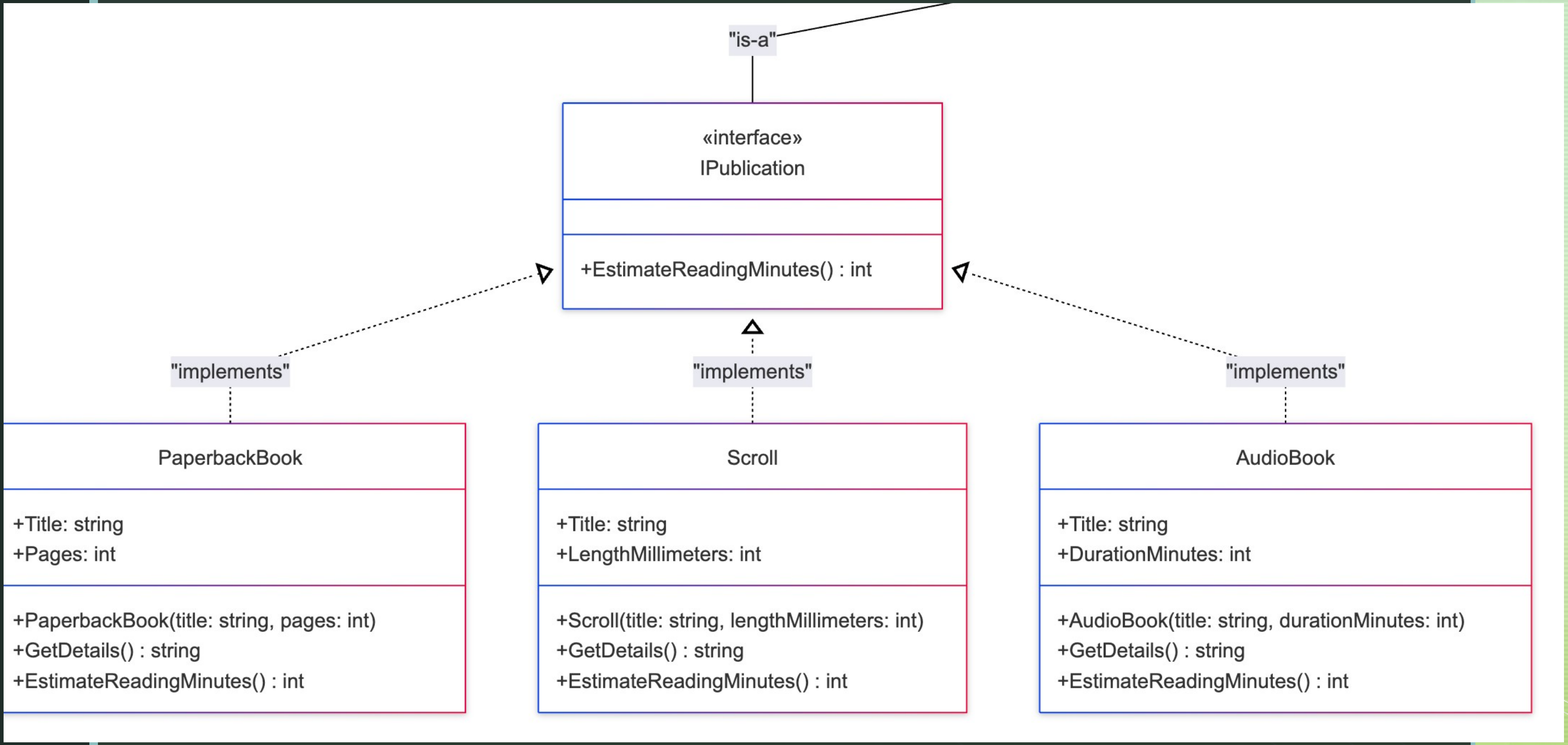
Kinds of Types

- Primitive types: built-in values
- Concrete types: instantiable classes
- Abstract types: interfaces and abstract classes

Type Examples

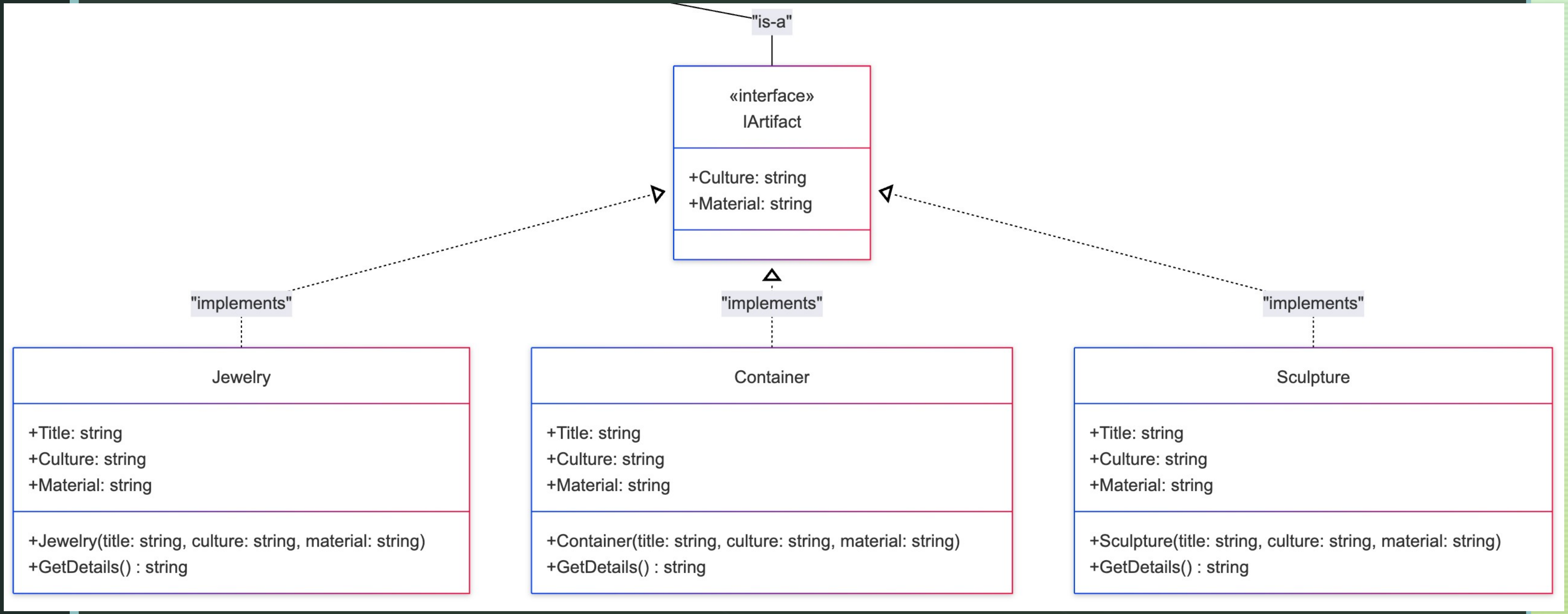
```
int count;              // primitive

class User { }           // concrete

interface IRepository { } // abstract

abstract class UserBase { }  // abstract with
implementation
```

Polymorphism 3
Program-04.cs

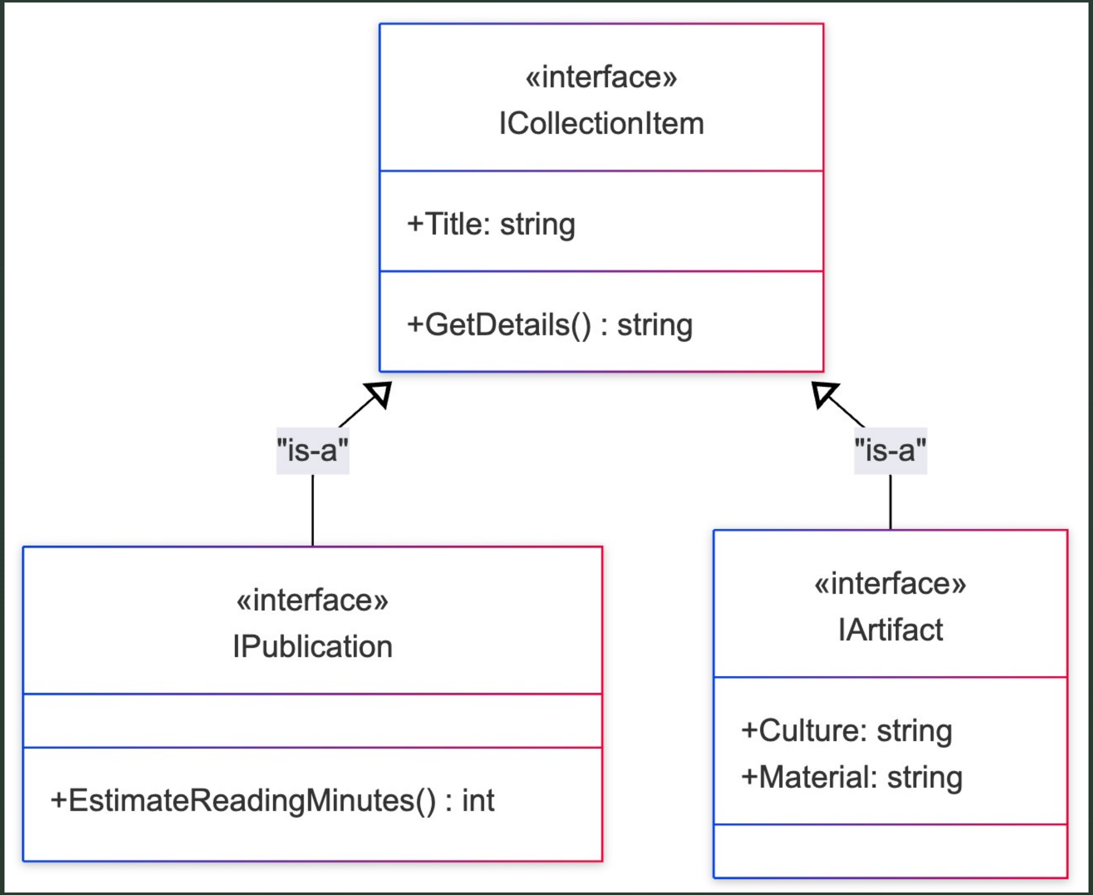


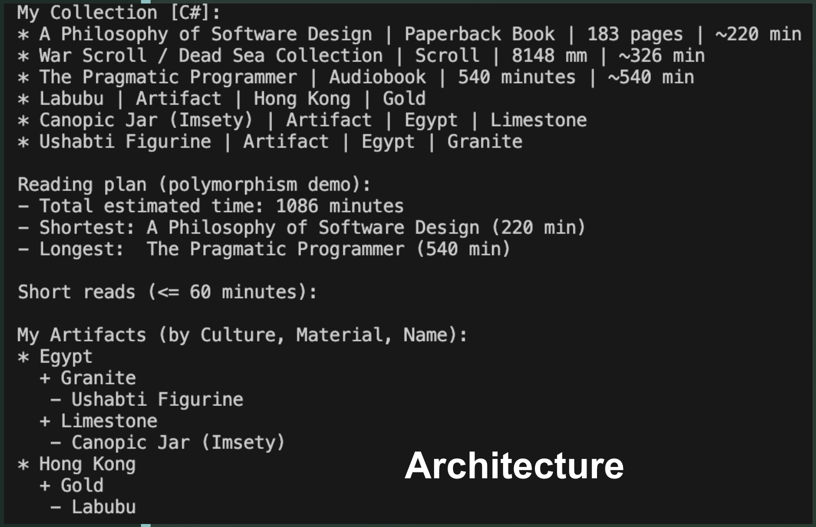
"is-a"
«interface»
IPublication
+EstimateReadingMinutes() : int
"implements"
"implements"
"implements"
PaperbackBook
+Title: string
+Pages: int
+PaperbackBook(title: string, pages: int)
+GetDetails() : string
+EstimateReadingMinutes() : int
Scroll
+Title: string
+LengthMillimeters: int
+Scroll(title: string, lengthMillimeters: int)
+GetDetails() : string
+EstimateReadingMinutes() : int
AudioBook
+Title: string
+DurationMinutes: int
+AudioBook(title: string, durationMinutes: int)
+GetDetails() : string
+EstimateReadingMinutes() : int

Polymorphism 3
Program-04.cs



Polymorphism 3
Program-04.cs





```
My Collection [C#]:
* A Philosophy of Software Design | Paperback Book | 183 pages | ~220 min
* War Scroll / Dead Sea Collection | Scroll | 8148 mm | ~326 min
* The Pragmatic Programmer | Audiobook | 540 minutes | ~540 min
* Labubu | Artifact | Hong Kong | Gold
* Canopic Jar (Imsety) | Artifact | Egypt | Limestone
* Ushabti Figurine | Artifact | Egypt | Granite

Reading plan (polymorphism demo):
- Total estimated time: 1086 minutes
- Shortest: A Philosophy of Software Design (220 min)
- Longest:  The Pragmatic Programmer (540 min)

Short reads (<= 60 minutes):

My Artifacts (by Culture, Material, Name):
* Egypt
  + Granite
   - Ushabti Figurine
  + Limestone
   - Canopic Jar (Imsety)
* Hong Kong
  + Gold
   - Labubu
```

Polymorphism 3
Program-04.cs

Architecture

- Introduces a **top-level interface** (ICollectionItem) representing a broad capability.
- Specialized interfaces (IPublication) **extend the top-level interface**.
- Concrete classes implement the **most specific interface** appropriate to them.
- Client code works against **multiple abstraction levels**.

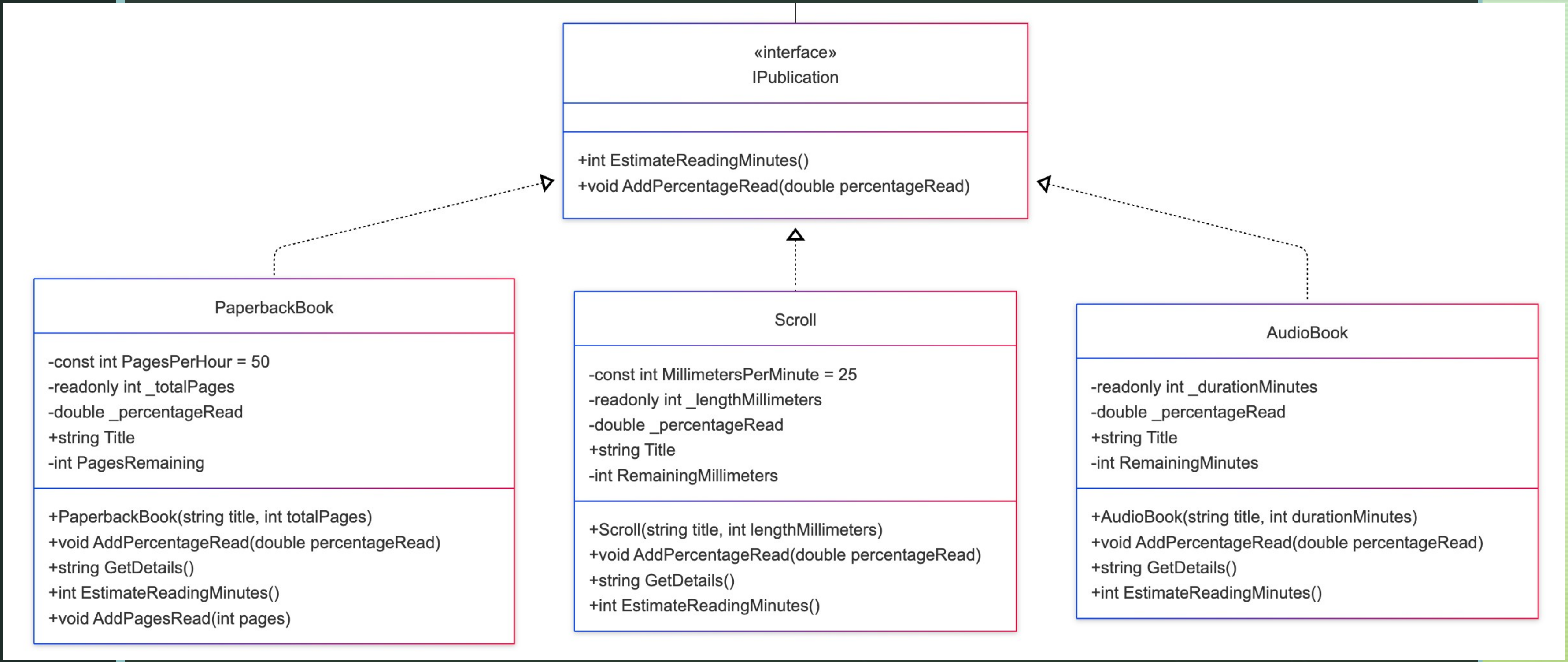
OO Concepts Illustrated

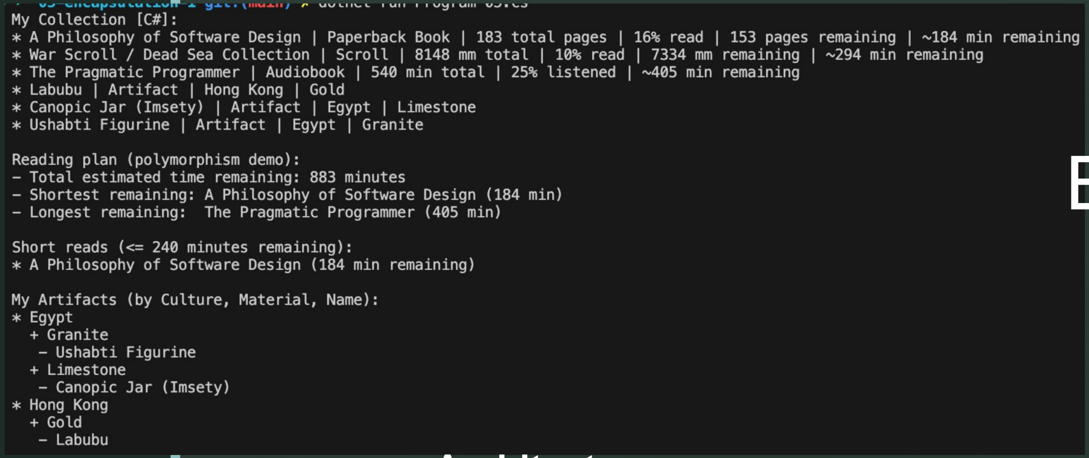
- Interface inheritance
- Liskov Substitution Principle (LSP)
- Deep polymorphism

Encapsulation

- A class controls its own data
- State is accessed through methods
- Protects internal consistency

Encapsulation 1
Program-05.cs


«interface»
IPublication
+int EstimateReadingMinutes()
+void AddPercentageRead(double percentageRead)
PaperbackBook
-const int PagesPerHour = 50
-readonly int _totalPages
-double _percentageRead
+string Title
-int PagesRemaining
+PaperbackBook(string title, int totalPages)
+void AddPercentageRead(double percentageRead)
+string GetDetails()
+int EstimateReadingMinutes()
+void AddPagesRead(int pages)
Scroll
-const int MillimetersPerMinute = 25
-readonly int _lengthMillimeters
-double _percentageRead
+string Title
-int RemainingMillimeters
+Scroll(string title, int lengthMillimeters)
+void AddPercentageRead(double percentageRead)
+string GetDetails()
+int EstimateReadingMinutes()
AudioBook
-readonly int _durationMinutes
-double _percentageRead
+string Title
-int RemainingMinutes
+AudioBook(string title, int durationMinutes)
+void AddPercentageRead(double percentageRead)
+string GetDetails()
+int EstimateReadingMinutes()



```
➜  05-encapsulation-1 git:(main) ✗ dotnet run Program-05.cs
My Collection [C#]:
* A Philosophy of Software Design | Paperback Book | 183 total pages | 16% read | 153 pages remaining | ~184 min remaining
* War Scroll / Dead Sea Collection | Scroll | 8148 mm total | 10% read | 7334 mm remaining | ~294 min remaining
* The Pragmatic Programmer | Audiobook | 540 min total | 25% listened | ~405 min remaining
* Labubu | Artifact | Hong Kong | Gold
* Canopic Jar (Imsety) | Artifact | Egypt | Limestone
* Ushabti Figurine | Artifact | Egypt | Granite

Reading plan (polymorphism demo):
- Total estimated time remaining: 883 minutes
- Shortest remaining: A Philosophy of Software Design (184 min)
- Longest remaining:  The Pragmatic Programmer (405 min)

Short reads (<= 240 minutes remaining):
* A Philosophy of Software Design (184 min remaining)

My Artifacts (by Culture, Material, Name):
* Egypt
  + Granite
   - Ushabti Figurine
  + Limestone
   - Canopic Jar (Imsety)
* Hong Kong
  + Gold
   - Labubu
```

Encapsulation 1 Program-05.cs

Architecture

- Introduces **private fields** to encapsulate internal state.
- State changes are controlled via **methods and properties**.
- Demonstrates **casting between interfaces and concrete types** where appropriate.
- Maintains polymorphism while protecting object invariants.

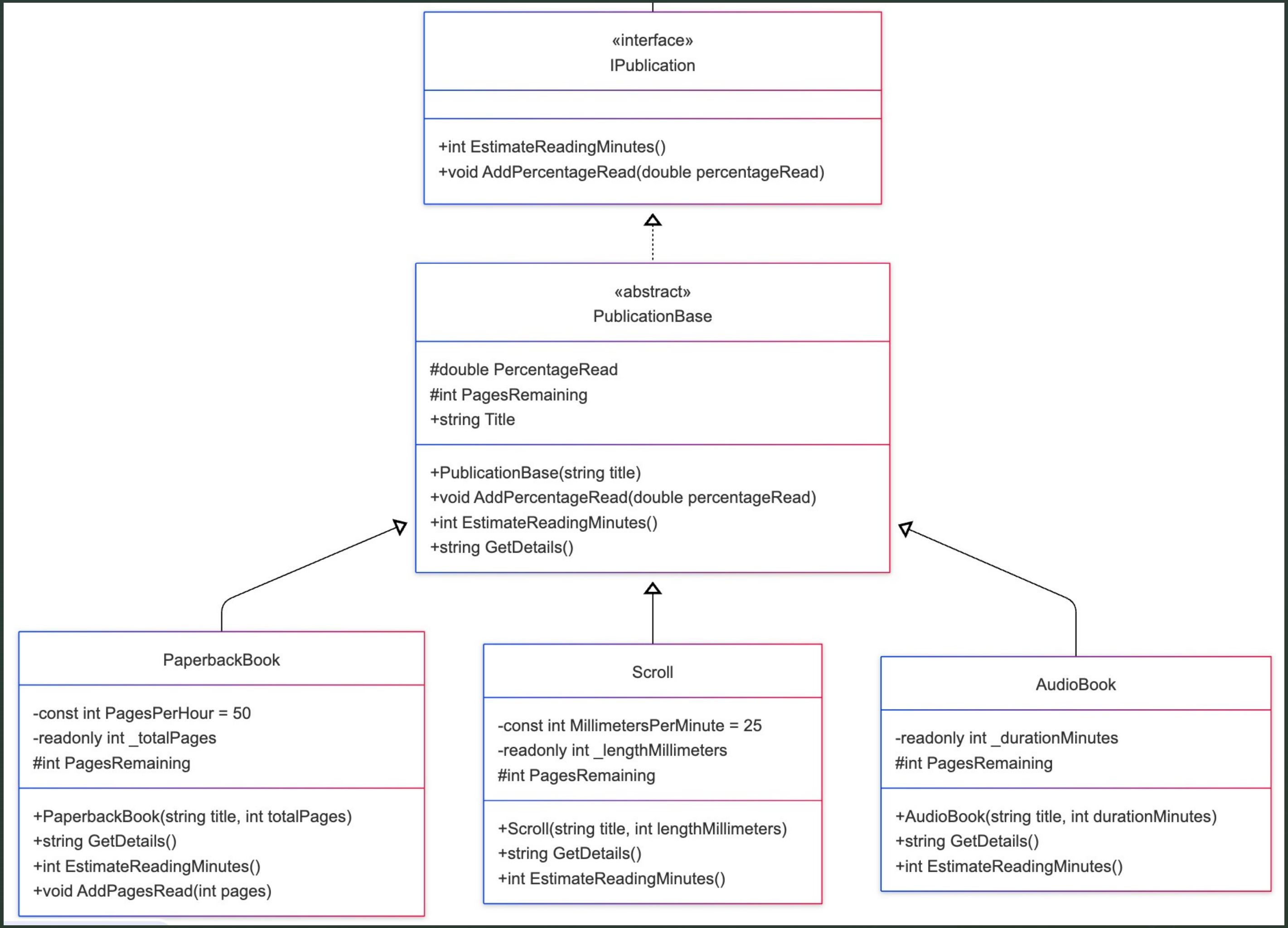
OO Concepts Illustrated

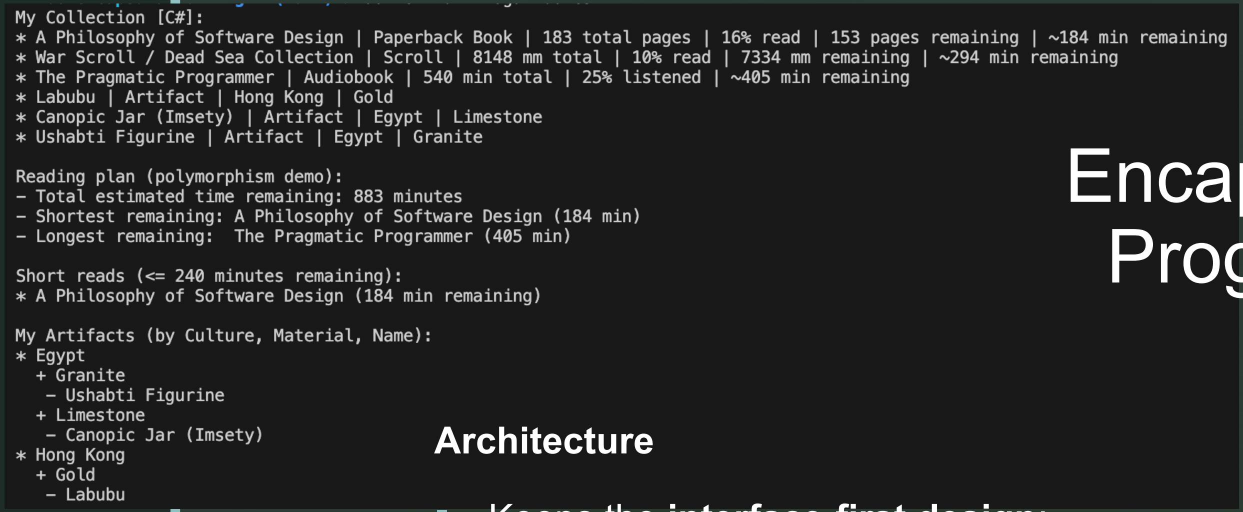
- Encapsulation
- Information hiding
- Controlled access to state
- Safe downcasting

Access Modifiers

- private: accessible only inside the class
- public: accessible everywhere
- protected: accessible to subclasses

Encapsulation 2
Program-06.cs


«interface»
IPublication
+int EstimateReadingMinutes()
+void AddPercentageRead(double percentageRead)
«abstract»
PublicationBase
#double PercentageRead
#int PagesRemaining
+string Title
+PublicationBase(string title)
+void AddPercentageRead(double percentageRead)
+int EstimateReadingMinutes()
+string GetDetails()
PaperbackBook
-const int PagesPerHour = 50
-readonly int _totalPages
#int PagesRemaining
+PaperbackBook(string title, int totalPages)
+string GetDetails()
+int EstimateReadingMinutes()
+void AddPagesRead(int pages)
Scroll
-const int MillimetersPerMinute = 25
-readonly int _lengthMillimeters
#int PagesRemaining
+Scroll(string title, int lengthMillimeters)
+string GetDetails()
+int EstimateReadingMinutes()
AudioBook
-readonly int _durationMinutes
#int PagesRemaining
+AudioBook(string title, int durationMinutes)
+string GetDetails()
+int EstimateReadingMinutes()



```
My Collection [C#]:
* A Philosophy of Software Design | Paperback Book | 183 total pages | 16% read | 153 pages remaining | ~184 min remaining
* War Scroll / Dead Sea Collection | Scroll | 8148 mm total | 10% read | 7334 mm remaining | ~294 min remaining
* The Pragmatic Programmer | Audiobook | 540 min total | 25% listened | ~405 min remaining
* Labubu | Artifact | Hong Kong | Gold
* Canopic Jar (Imsety) | Artifact | Egypt | Limestone
* Ushabti Figurine | Artifact | Egypt | Granite

Reading plan (polymorphism demo):
- Total estimated time remaining: 883 minutes
- Shortest remaining: A Philosophy of Software Design (184 min)
- Longest remaining:  The Pragmatic Programmer (405 min)

Short reads (<= 240 minutes remaining):
* A Philosophy of Software Design (184 min remaining)

My Artifacts (by Culture, Material, Name):
* Egypt
  + Granite
   - Ushabti Figurine
  + Limestone
   - Canopic Jar (Imsety)
* Hong Kong
  + Gold
   - Labubu
```

Encapsulation 2
Program-06.cs

Architecture

- Keeps the **interface-first design**:
  - ICollectionItem remains the base interface for anything in the collection.
  - IPublication : ICollectionItem still defines “publication” capabilities (e.g., estimated time remaining + progress updates).
  - IArtifact : ICollectionItem continues to represent non-reading/listening collection items.
- Introduces a new **abstract class beneath the interface**:
  - PublicationBase : IPublication acts as a shared implementation layer.
  - Holds common state for publication progress using **protected fields** (teaching-focused encapsulation).
- Centralizes shared progress behavior:
  - A single implementation of progress mutation logic (e.g., “add % read/listened”) now lives in the abstract base.
  - Concrete publication types override only what varies (e.g., how “remaining” is computed and how details are presented).

Encapsulation 2
Program-06.cs

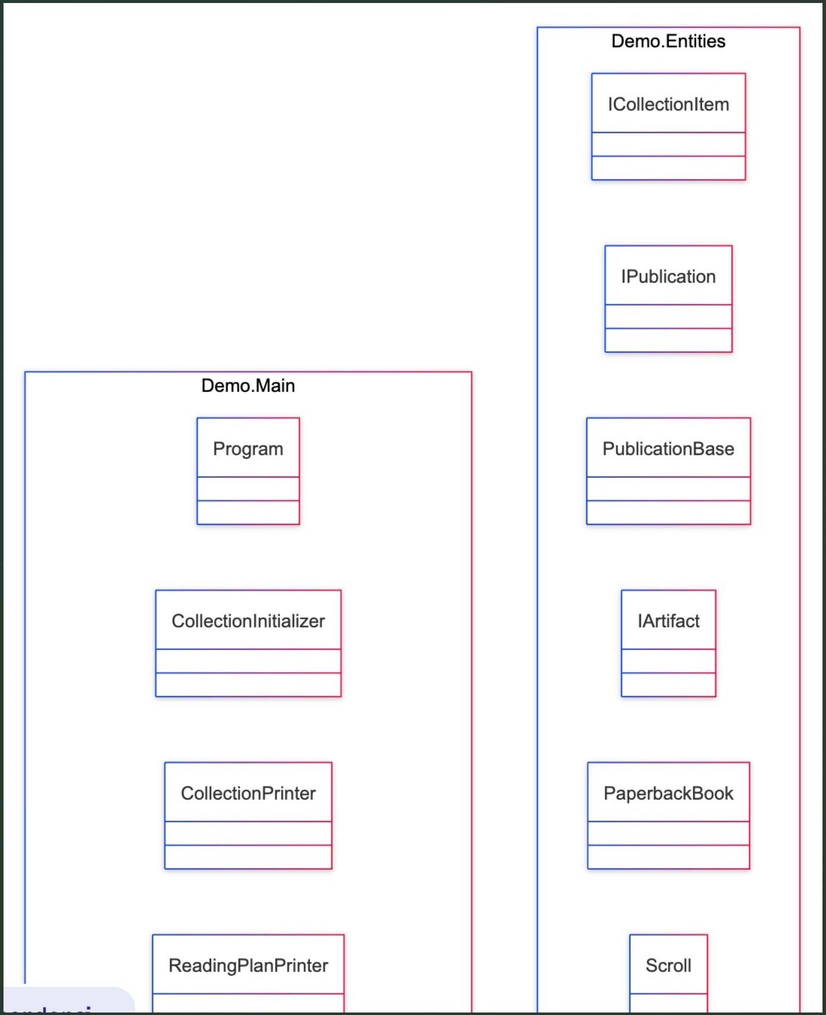
OO Concepts Illustrated

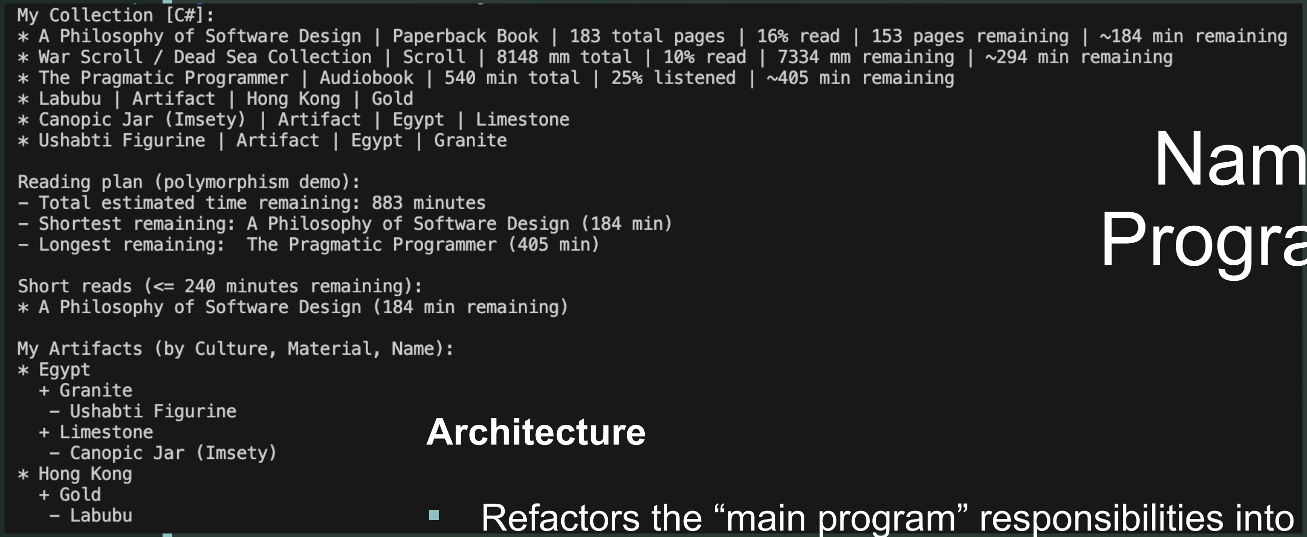
- Encapsulation via protected (and what it exposes vs private)
- Interfaces + abstract classes together (“contract + partial implementation”)
- Reuse without duplicating state/logic
- Template-style variation: base handles common mechanics, subclasses specialize specifics

Namespaces and Organization

- Group related classes
- Prevent naming conflicts
- Improve readability

Namespaces Program-07.cs





```
My Collection [C#]:
* A Philosophy of Software Design | Paperback Book | 183 total pages | 16% read | 153 pages remaining | ~184 min remaining
* War Scroll / Dead Sea Collection | Scroll | 8148 mm total | 10% read | 7334 mm remaining | ~294 min remaining
* The Pragmatic Programmer | Audiobook | 540 min total | 25% listened | ~405 min remaining
* Labubu | Artifact | Hong Kong | Gold
* Canopic Jar (Imsety) | Artifact | Egypt | Limestone
* Ushabti Figurine | Artifact | Egypt | Granite

Reading plan (polymorphism demo):
- Total estimated time remaining: 883 minutes
- Shortest remaining: A Philosophy of Software Design (184 min)
- Longest remaining:  The Pragmatic Programmer (405 min)

Short reads (<= 240 minutes remaining):
* A Philosophy of Software Design (184 min remaining)

My Artifacts (by Culture, Material, Name):
* Egypt
  + Granite
   - Ushabti Figurine
  + Limestone
   - Canopic Jar (Imsety)
* Hong Kong
  + Gold
   - Labubu
```

Namespaces Program-07.cs

Architecture

- Refactors the "main program" responsibilities into **focused helper classes**:
  - CollectionInitializer (builds the collection)
  - CollectionPrinter (prints all items)
  - ReadingPlanPrinter (prints reading plan view)
  - ShortReadsPrinter (filters and prints short reads)
  - ArtifactsByCultureMaterialNamePrinter (grouping/sorting/reporting)
- Program.Main() becomes a small coordinator:
  - Instantiates helpers
  - Orchestrates the workflow
  - No longer contains all the logic itself

Namespaces
Program-07.cs

OO Concepts Illustrated

- Avoiding a **God class**
- Single Responsibility Principle (SRP)
- Basic “composition of services” (Main composes small collaborators)
- “Namespace/storytelling” benefits:
  - Code layout communicates intent (main entry + helpers + entities)

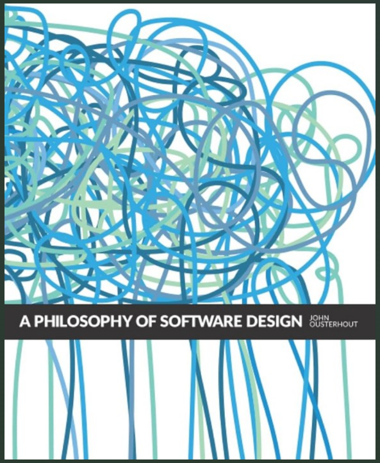
What You Should Take Away

- OO concepts give us shared language
- Each concept solves a specific problem
- The following lecture goes into diagramming with Mermaid.
- Following the diagramming lecture, we go on to how to make good OO design decisions.

Assigned Readings

A Philosophy of Software Design - Ousterhout

- Preface
- Chapter 1: Introduction
- Chapter 2: The Nature of Complexity



Lecture Recording Jan 14 2026

https://www.loom.com/share/b54e52c10e0a4d11909f2adbefee5c14