codingOn

디자인 패턴

SW 디자인 패턴

- 소프트웨어 개발에 있어 자주 발생하는 문제들에 대한 일반적인 해결책
- 특정 문맥(소스코드 구조)에서 발생하는 설계 문제를 해결
- 설계 과정에 대한 가이드 또는 지침

- 즉, 자주 발생하는 문제들에 대해 미리 특정한 패턴을 만들어둔 것
- 주로 클래스 정의, 클래스 설계, 상속 관계 등에 있어서 특정한 소스 코드 작성 방식을 말함

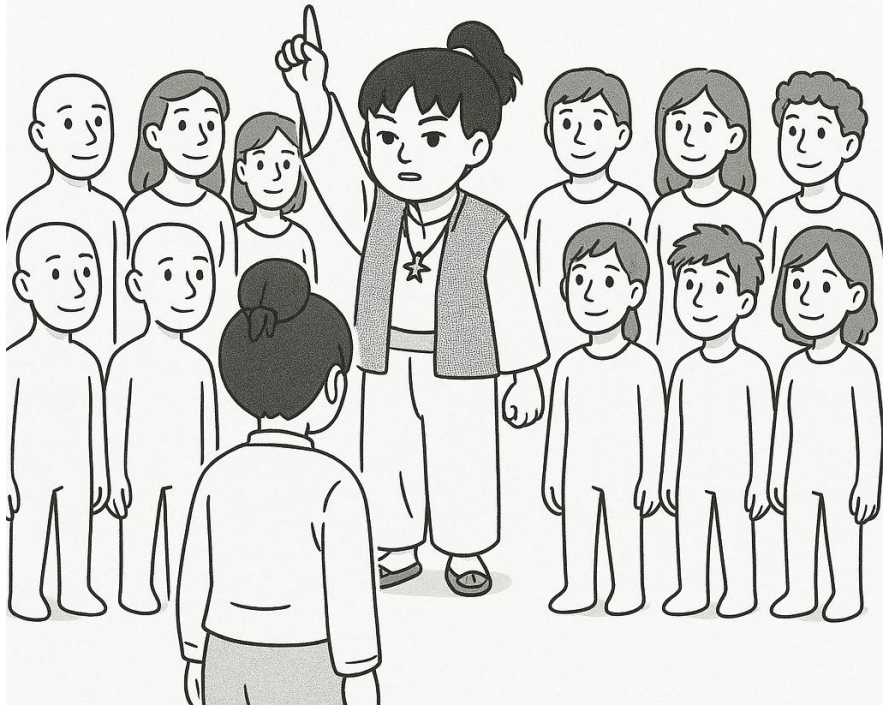
codingOn

SW 디자인 패턴

- 다양한 상황에 대한 해결 방법(디자인 패턴)이 있음
  - Singleton: 한 클래스의 인스턴스가 오직 하나만 존재하도록 보장
  - Prototype: 기존 객체를 복제하여 새로운 객체를 생성
  - Adapter: 호환되지 않는 인터페이스를 가진 클래스들이 함께 작동할 수 있도록 함
  - Bridge: 추상화(Abstract)와 구현을 분리하여, 독립적으로 변형될 수 있게 함
  - Interpreter: 주어진 언어의 문법에 대한 표현을 정의하고 문장을 해석
  - State: 객체의 내부 상태에 따라 객체의 행동을 변경
  - Strategy: 알고리즘을 객체의 행동으로 캡슐화하여 동일한 기능을 하는 다른 알고리즘으로 교체할 수 있게 함
  - Iterator: 컬렉션 내의 요소를 순차적으로 접근하는 방법을 제공
  - ...

Singleton 패턴

- 프로그램 실행 중 **단 하나의 인스턴스**만 생성되도록 보장하고, 그 인스턴스에 어디서든 접근할 수 있게 만드는 디자인 패턴.



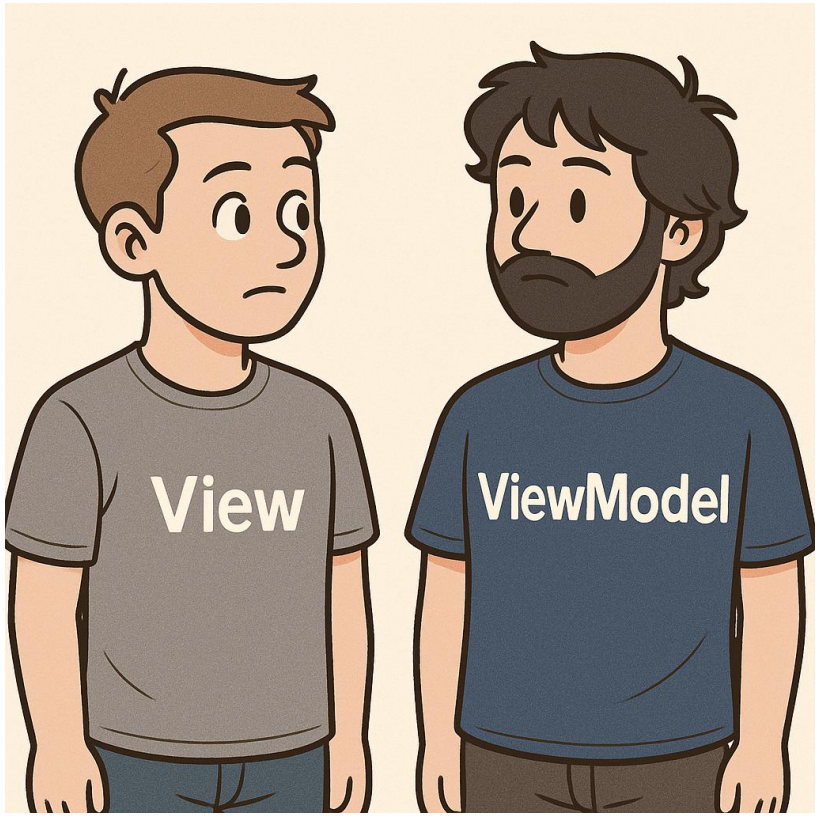
특징

1. 외부에서 객체를 새로 생성하지 못하도록 막음
   - private 생성자
2. 클래스 수준에서 단 하나의 인스턴스를 저장
   - static 필드
3. 이 인스턴스를 어디서든 접근할 수 있게 해줌
   - static 메서드

Ex. 로그 기록기, DB 연결 클래스, 설정 관리자 등등

WPF MVVM 패턴

- WPF 출시와 함께 Microsoft에서 개발한 디자인 패턴
- Model + View + View Model 로 구성됨
- 사용자 인터페이스에서 발생하는 이벤트 중심의 프로그래밍에서 탈피하기 위한 목적에서 고안 됨 (View가 너무 많은 기능을 담당하는 것을 방지)
- 개발 속도 향상 보다 유지보수를 쉽게 하기 위함



WPF MVVM 패턴

- Model (프로그래머)
  - 데이터와 비즈니스 로직을 담당.
  - DB, 네트워크 요청 또는 파일 시스템과 같은 데이터 소스와 상호작용
- View (디자이너)
  - 사용자 인터페이스를 담당하여 사용자 입력처리 및 화면 갱신을 처리
  - XAML 같은 마크업 언어를 사용하여 디자인
- ViewModel (프로그래머)
  - View와 Model 사이에서 중재자 역할을 수행
  - View에서 발생하는 이벤트 감지, 이벤트에 맞는 Model의 로직 수행
  - View에 표시할 데이터를 가공

codingOn

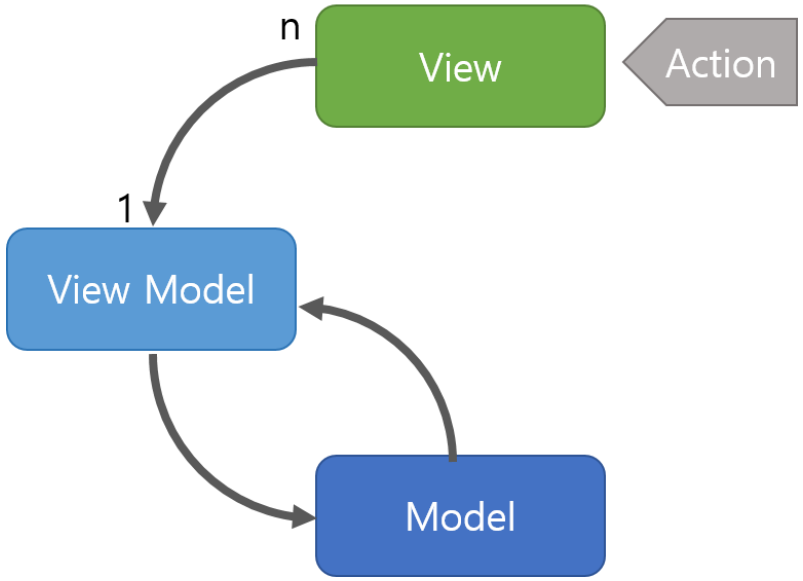
WPF MVVM 패턴

특징

- 역할 분리 명확함.
  - View는 **UI**만, ViewModel은 **로직 처리**만, Model은 **데이터 관리**만
- 데이터 바인딩 중심으로 동작.
  - ViewModel의 속성과 View의 UI 요소를 자동으로 연결
  - 값이 바뀌면 UI에 자동 반영
- 명령(Command) 패턴으로 사용자 입력 처리
  - 버튼 클릭 등 이벤트를 Icommand로 처리
- 테스트, 유지보수 용이 및 확장성 증가.

MVVM 패턴

- MVVM 패턴 동작 시나리오
  1. 사용자의 Action은 View를 통해 들어옴
  2. View에 Action이 들어오면, Command 패턴으로 View Model에 Action을 전달
     - Command 패턴은 사용자 요청을 매개변수로 직접 전달하는 것이 아니라 각종 관련 정보들을 포함한 객체로 한 번 감싸서 전달하는 것
  3. View Model은 Model에게 데이터를 요청
  4. Model은 View Model에게 요청받은 데이터를 응답
  5. View Model은 응답 받은 데이터를 가공하여 저장
  6. View는 View Model과 Data Binding하여 화면을 그림



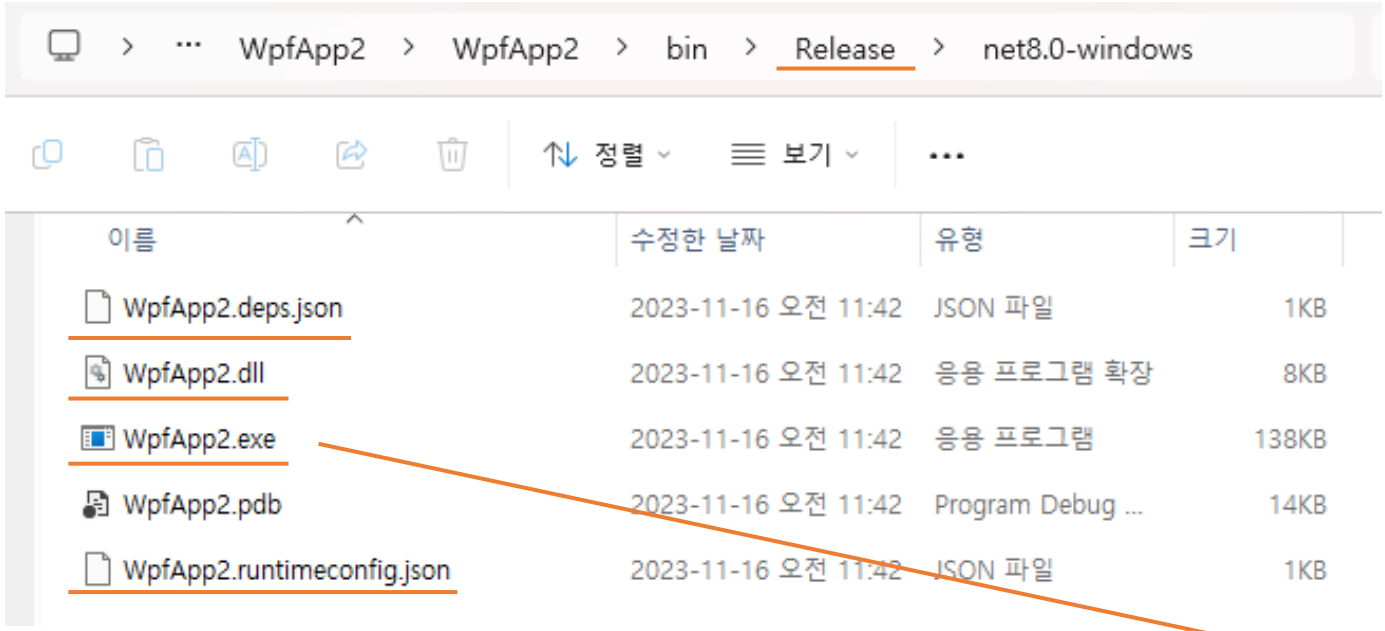
릴리즈

릴리즈 빌드

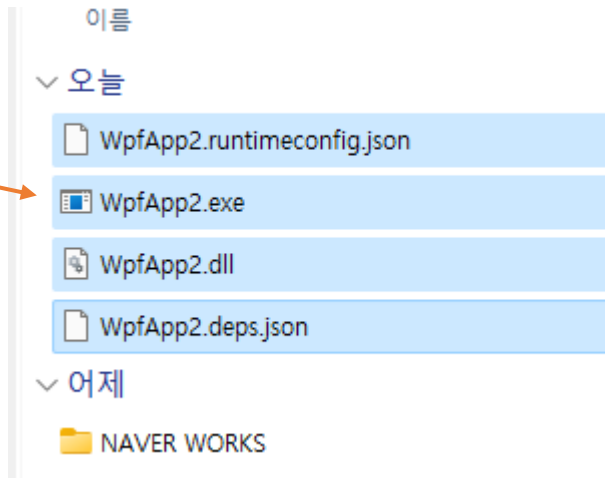
- 코드 실행 결과는 동일

Debug	Release
• 코드 최적화 없음 • 코드 실행 속도 느림 • 메모리 사용량이 많음 • 실행 파일에 디버깅에 필요한 정보 포함 • 컴파일 속도 빠름	• 코드 최적화 진행 • 코드 실행 속도 빠름 • 메모리 사용량이 적음 • 디버깅에 필요한 정보가 거의 포함되지 않음 • 최적화 때문에 컴파일 속도 느림

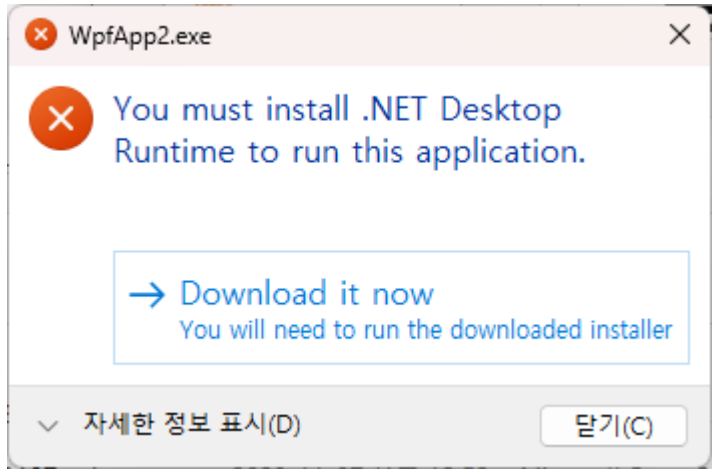
릴리즈 빌드



다른 위치에 복사 및 실행



릴리즈 빌드



닷넷이 설치되어 있지 않은 PC에서는
닷넷 런타임 설치가 필요함

https://dotnet.microsoft.com/en-us/download/dotnet/8.0

.NET Desktop Runtime 8.0.0

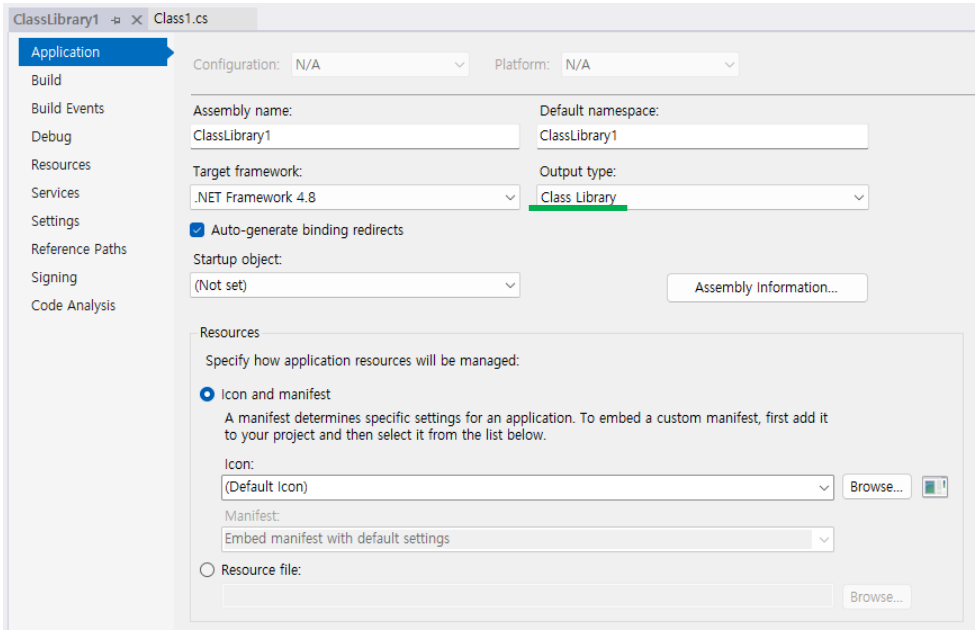
The .NET Desktop Runtime enables you to run existing Windows desktop applications. **This release includes the .NET Runtime; you don't need to install it separately.**

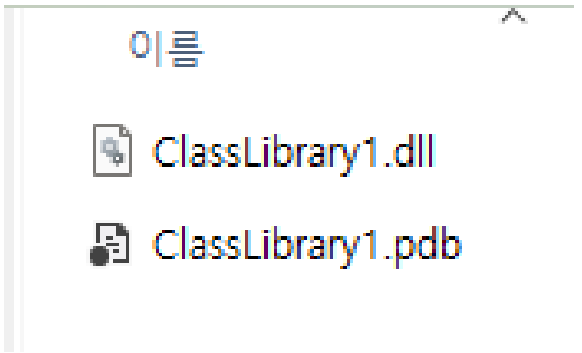
OS	Installers	Binaries
Windows	Arm64 | x64 | x86 | winget instructions	

.NET Runtime 8.0.0

dll 만들기

- 프로젝트 속성 (우클릭 맨 아래) > Output type: Class Library 로 변경
- 프로젝트 빌드 후 프로젝트 폴더/bin/(빌드 모드)/(프로젝트 이름).dll 생성
- Debug 모드로 만든 dll은 Debug 모드에서만 사용 가능 (반대도 마찬가지)





dll 가져오기

- 프로젝트 우클릭 > Add > Project Reference > Browse > dll 파일 선택
- 프로젝트/Depedencies/Assemblies에 namespace가 추가된 것을 확인
- using으로 namespace를 가져오거나 전체 이름을 지정하여 사용