데이터 클래스 개념

이름 : 조대훈

목차

1. 데이터 클래스 종류

1.1 NomalType-Baesd Data Class

- 필드 멤버가 일반 형식, 사용자 정의 열거형/구조체/클래스 형식으로
  이루어진 클래스입니다.

- 데이터 특징 + Data' 이름을 갖게됩니다.

1.2 Array-Based Data Calss

- '배열' 형식을 갖는 클래스 멤버가 존재하는 데이터 클래스를 말합니다.

- '반복되는 데이터 이름 + Group' 이름을 갖게됩니다.

1.3 HashTable-Based Data Calss

- 'Key-Value' 형식을 갖는 클래스 멤버가 존재하는 데이터 클래스를 말합니다.

- '반복되는 데이터 이름 + Repository'이름을 갖게됩니다.

- '중복 HashTable-Based Data Class'의 경우, 접미어로 + 'NestedRepository'을 갖게됩니다.

2 데이터 클래스 초기화 방식

① 동적 데이터

- 데이터 초기화 시, '개발자 지정 Default 값' 또는 '사용자 설정 값'을 할당해야 합니다.

② 불변 데이터

- 데이터 초기화 시, '개발자 지정 Default 값'을 할당해야 합니다.

※ 데이터 클래스의 초기화

- '초기화'는 앞서 작성한 '데이터 분류'와 관련이 깊습니다.

- 데이터가 저장된 위치, 호출 시점 등에 따라서 추가적인 클래스를 필요로합니다.

- 초기화 관련 내용은 차후, 다른 문서에서 작성할 계획입니다.

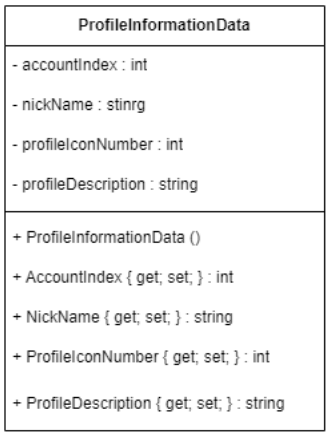
3. 데이터 클래스 책임

- 필드 멤버로 갖는 데이터에 대한 Get, Set 역할을 수행하는 메소드가 필요합니다.
- 각 Data 클래스에서 사용한 자료구조의 특징에 맞는 추가 메소드를 작성하여 사용합니다.
- 각 Data 의 성격과 특징에 알맞게 필요한 추가 메소드를 작성하여 사용합니다.

3.1 NomalType-Based Data Class

- 필드 멤버에 대한 Get 과 Set 프로퍼티만 제공하면 됩니다.

3.1.1 UML



3.1.2 코드

ProfileInformationData.cs

Assembly-CSharp

Portfolio.ProfileInformationData

```
namespace Portfolio
{
    참조 11개
    public class ProfileInformationData
    {
        private int accountIndex;
        private string nickName;
        private int profileIconNumber;
        private string profileDescription;

        참조 0개
        public ProfileInformationData(int accountIndex, string nickName,
                                      int profileIconNumber, string profileDescription)
        {
            this.accountIndex = accountIndex;
            this.nickName = nickName;
            this.profileIconNumber = profileIconNumber;
            this.profileDescription = profileDescription;
        }

        참조 0개
        public int AccountIndex { get => accountIndex; set => accountIndex = value; }
        참조 0개
        public string NickName { get => nickName; set => nickName = value; }
        참조 0개
        public int ProfileIconNumber { get => profileIconNumber; set => profileIconNumber = value; }
        참조 0개
        public string ProfileDescription { get => profileDescription; set => profileDescription = value; }
    }
}
```

3.2 Array-Based Data Class

- 필드 멤버를 직접적으로 접근하여 사용하면 잘못된 사용으로 연결될 수도 있기에 필드 멤버에 대한 프로퍼티를 사용하지 않습니다.

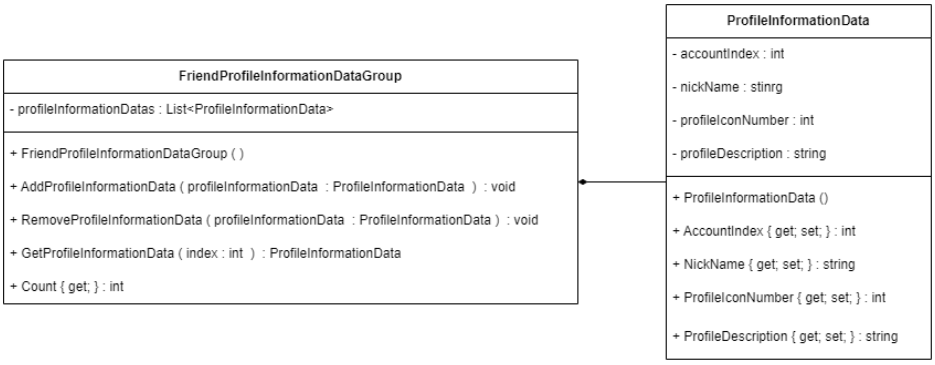
- **'순환'이 자주** 필요한 데이터, HashTable-Based Data 로는 감당하기 힘든 메모리를 사용하는 데이터의 경우에 사용됩니다.

- 필드 멤버에 대한 Get 역할을 하는 'Get 관련 메소드'가 필요합니다.

- 필드 멤버에 대한 Set 역할을 하는 'Add', 'Remove 메소드'가 필요합니다.

- Array 의 특징에 따라서, 데이터의 길이를 반환하는 'Count 메소드'가 자주 사용됩니다.

3.2.1 UML



3.2.2 코드

FriendProfileInformationDataGroup.cs

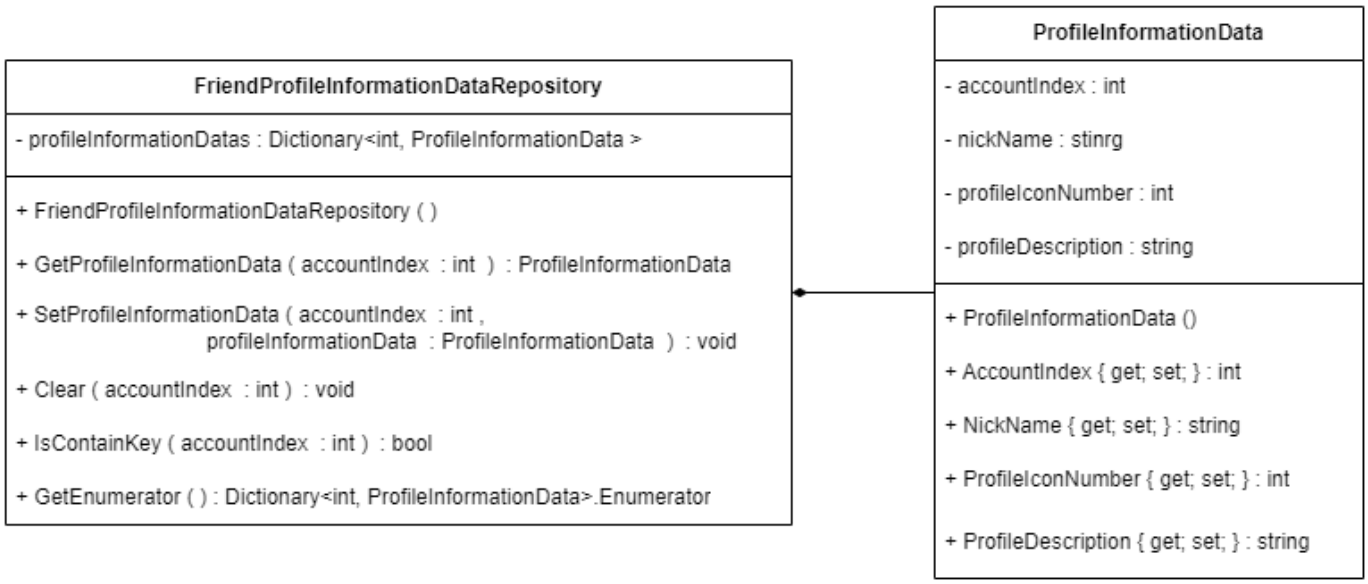
Assembly-CSharp — Portfolio.FriendProfileInformationDataGroup

```
using System.Collections.Generic;

namespace Portfolio
{
    참조 1개
    public class FriendProfileInformationDataGroup
    {
        private List<ProfileInformationData> profileInformationDatas;

        참조 0개
        public FriendProfileInformationDataGroup()
        {
            this.profileInformationDatas = new List<ProfileInformationData>();
        }

        참조 0개
        public ProfileInformationData GetProfileInformationData(int index)
        {
            if(this.profileInformationDatas.Count > index)
                return this.profileInformationDatas[index];
            else
                return null;
        }

        참조 0개
        public void AddProfileInformationData(ProfileInformationData profileInformationData)
        {
            this.profileInformationDatas.Add(profileInformationData);
        }
        참조 0개
        public void RemoveProfileInformationData(ProfileInformationData profileInformationData)
        {
            this.profileInformationDatas.Remove(profileInformationData);
        }

        참조 0개
        public int Count { get => this.profileInformationDatas.Count; }
    }
}
```

3.3 HashTable-Based Data Calss

- 필드 멤버를 직접적으로 접근하여 사용하면 잘못된 사용으로 연결될 수도 있기에 필드 멤버에 대한 프로퍼티를 사용하지 않습니다.
- **'순환'이 거의 없는** 데이터, 특정 Key 값에 대응되는 Value 값이 즉각적으로 반환해야 되는 데이터의 경우에 사용됩니다.
- 필드 멤버에 대한 Get 역할을 하는 'GetValue 메소드'가 필요합니다.
- 필드 멤버에 대한 Set 역할을 하는 'Set', 'Clear 메소드'가 필요합니다.
- HashTable 의 특징에 따라서, 특정 Key 값의 사용여부를 반환하는 'IsContainKey 메소드'가 자주 사용됩니다.
- HashTable 의 특징에 따라서, HashTable 순환을 위한 'GetAllKeys 메소드'가 가끔 필요합니다.

3.3.1 UML



3.3.2 코드

FriendProfileInformationRepository.cs

Assembly-CSharp | Portfolio.FriendProfileInformationRepository | SetProfileInformatio

```
using System.Collections.Generic;

namespace Portfolio
{
    참조 1개
    public class FriendProfileInformationRepository
    {
        private Dictionary<int, ProfileInformationData> profileInformationDatas;

        참조 0개
        public FriendProfileInformationRepository()
        {
            this.profileInformationDatas = new Dictionary<int, ProfileInformationData>();
        }

        참조 0개
        public ProfileInformationData GetProfileInformationData(int accountIndex)
        {
            if (this.profileInformationDatas.ContainsKey(accountIndex))
                return this.profileInformationDatas[accountIndex];
            else
                return null;
        }

        참조 0개
        public void SetProfileInformationData(int accountIndex, ProfileInformationData profileInformationData)
        {
            if (this.profileInformationDatas.ContainsKey(accountIndex))
                this.profileInformationDatas[accountIndex] = profileInformationData;
            else
                this.profileInformationDatas.Add(accountIndex, profileInformationData);
        }

        참조 0개
        public void Clear(int accountIndex)
        {
            if (this.profileInformationDatas.ContainsKey(accountIndex))
                this.profileInformationDatas.Remove(accountIndex);
            else
                return;
        }

        참조 0개
        public bool IsContainKey(int accountIndex)
        {
            if (this.profileInformationDatas.ContainsKey(accountIndex))
                return true;
            else
                return false;
        }

        참조 0개
        public Dictionary<int, ProfileInformationData>.Enumerator GetEnumerator()
        {
            return this.profileInformationDatas.GetEnumerator();
        }
    }
}
```