면접 질문 총괄

클라 - 기초

1 바이트(1 Byte)는 몇비트인가?

역사적으로 하드웨어에 따라 다르며, 명확한 기준은 없지만 8비트가 일반적입니다.

short 2 / int 4 / long 4

ushort 2 / uint 4 / ulong 4

char 1 / uchar 1 / float 4 / double 8

1픽셀은 몇 바이트인가? 3가지 색을 표현하기 위해서는 몇 바이트가 필요한가?

1픽셀은 1바이트이며 3가지 색을 표현하기 위해서는 3바이트가 필요합니다.

Call By Ref의 의미는 무엇인가?

함수에 주소값을 전달하는 방식입니다.

이벤트 리스너란 무엇인가?

이벤트 핸들러를 추가할 수 있는 객체로 이벤트가 발생할때마다 해당 이벤트를 처리하는 역할을 수행합니다.

동적 바인딩 (=다이나믹 바인딩)이란 무엇인가?

런타임에 값에 따라 변수, 데이터 타입, 호출 함수가 결정됩니다.

[TODO] C# 리플렉션이란 무엇인가?

프로그램 실행 중 자신의 구조를 검사하고 수정할 수 있게 하는 기능

- 컴파일 시점이 아닌 런타임 시점에 타입의 메타데이터를 읽거나

인스턴스를 생성, 메소드 호출, 프로퍼티나 필드에 접근할 수 있게 함

- 타입 정보 조회 (T) 클래스나 GetType()
- 동적 인스턴스 생성
- 메소드 실행
- 필드와 프로퍼티에 접근 및 수정

리플렉션 장단점

- 성능 저하
- 보안 문제 (프라이빗 멤버에 접근 가능해서)
- 유지보수 어려움

따라서 신중한 사용이 필요함

[TODO] 퓨어 펑션이란 무엇인가?

순수함수로 [같은 입력]이 주어지면 같은 값을 반환하고, 사이트 이펙트가 없어야하는 함수입니다.

클라 - 객체지향

OOP란 무엇인가?

오브젝트 오리엔티드 프로그래밍의 약자로 [객체지향프로그래밍]을 의미합니다.

OOP의 5가지 특징은 무엇인가?

캡슐화, 정보은닉, 추상화, 상속, 다형성이 있습니다.

OOP의 다섯가지 특징을 설명하시오.

1. 캡슐화는 하나의 목적을 위해 [메소드]와 [데이터]를 하나로 묶는 것을 의미합니다.

재사용 용이하도록 하며, 객체의 세부 내용이 드러나지 않습니다.

2. 정보 은닉은 외부에는 필요한 정보만 공개하는 특징입니다.

3. 추상화는 우리가 상상한 물체를 객체화 합니다.

4. 상속은 이미 만들어진 클래스를 파생시켜 새 클래스를 만드는 기법입니다.

5. 다형성은 호출하는 객체에 따라 같은 함수라도 다른 동작을 할 수 있게 합니다.

[오버로딩] [오버라이딩]이 대표적입니다.

오버로딩과 오버라이딩에 대해 설명하시오.

오버로딩과 오버라이딩 모두 객체지향 프로그래밍의 다형성을 충족하기 위한 기법으로.

1. 오버로딩은 같은 이름을 쓰지만 매개변수의 유형과 갯수만 달라집니다.

2. 오버라이딩의 경우는 상위 클래스의 메소드를 하위 메소드에서 재정의합니다.

[TODO] 객체지향의 솔리드 원칙에 대해 설명하시오.

S - 한 클래스는 하나의 책임만

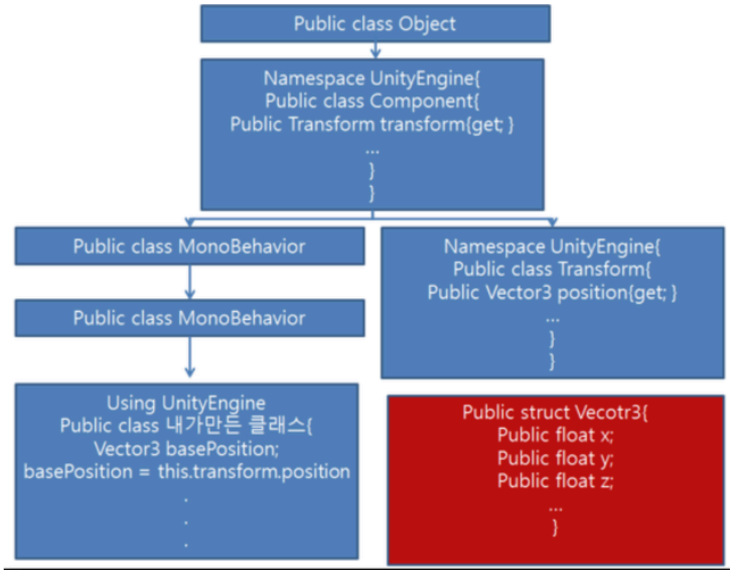
O - 폐쇄 원칙으로 기존 기능은 유지한채 기능을 확장할 수 있어야 합니다.

L - 프로그램의 정확성을 손상시키지 않고, 하위 타입의 인스턴스로 바꿀 수 있어야 합니다 (하위 객체는 상위 타입의 책임을 무시하거나 재정의 하지 않고 확장만을 수행합니다.)

I - 범용 인터페이스보단 인터페이스 여러개를 사용합니다.

D - 변동성이 적은 추상화에 의존해야 합니다. (구체화에 의존해선 안됩니다.)

클라 - 유니티


Public class Object
Namespace UnityEngine{
Public class Component{
Public Transform transform{get; }
...
}
}
Public class MonoBehavior
Public class MonoBehavior
Using UnityEngine
Public class 내가만든 클래스{
Vector3 basePosition;
basePosition = this.transform.position
.
.
.
Namespace UnityEngine{
Public class Transform{
Public Vector3 position{get; }
...
}
}
Public struct Vecotr3{
Public float x;
Public float y;
Public float z;
...
}

유니티의 자주 사용하는 기능들

- 3D / 2D 렌더링
- 물리 엔진 (리지드바디, 콜리더)
- 사용자 인터페이스 (UGUI)
- 애니메이션
- 스크립팅
- 에셋관리 (어드레서블)

Unity 메모리 관리법

가비지 컬렉터를 통해 메모리 관리를 합니다.

- 불필요한 알로케이션 피하기
- 오브젝트 풀링을 사용해 메모리 최적화
- 프로파일러로 메모리 누수 감지 및 해결

모바일 게임 개발 시 성능 최적화를 위한 접근 방식

- 저해상도 텍스쳐 사용
- 메쉬 최적화
- 오버 드로우 최소화
- 배치 처리 최적화
- 프로파일러 사용으로 성능 병목 점검

오버 드로우를 최소화 할 수 있는 방법은?

- 한 픽셀에 여러번 그리기 작업이 이루어질 때를 말함
- 오클루전 컬링 사용 (카메라의 보이지 않는 오브젝트의 렌더링을 방지)
- 오브젝트 레이어 사용 (카메라 컬링 마스크를 이용해 특정 레이어만 렌더링 되도록)
- 적절한 머테리얼과 셰이더 사용 (알파맵, 반투명 머테리얼 최소화)

- LOD 사용 (레벨 오브 디테일 사용) → 카메라에서 멀리 떨어진 오브젝트는 낮은 해상도 모델 사용

- 라이트 베이킹 사용 (오버드로우와 직접 연관은 없지만 렌더링 성능 전반에 긍정 영향)

그 외

프로파일링 도구 사용

텍스쳐와 메쉬 체적화

<details>
<summary>배치 처리를 최적화 할 수 있는 방법은?</summary>

배치 처리는 곧 렌더링 성능 최적화

- 배치는 GPU로 전송되는 드로우콜의 수를 말함

- 정적 배치 (스태틱 옵션 활성화)

- 동적 배치 (런타임에 변할 수 있는 오브젝트들을 자동 결합, 동일한 머테리얼 사용)

- GPU 인스턴싱 → 동일 메쉬와 머테리얼 사용 오브젝트를 효율적으로 렌더링 (동적 배치와 비슷하지만 더 많은 오브젝트에도 효과적, 머테리얼에서 Enable GPU Instancing을 체크)

- 머테리얼과 텍스쳐 최소화 (텍스쳐 아틀라스 사용)

- 적절한 LOD

- 오브젝트 컬링

- 셰이더 최적화
</details>

<details>
<summary>멀티 플랫폼 개발 경험과 극복</summary>

- 유연한 UI 설계를 했습니다

→ UGUI의 앵커 기능을 잘 사용해야함

- 플랫폼별 조건부 컴파일 사용

- 각 플랫폼에 맞는 테스팅 진행
</details>

<details>
<summary>코루틴에 대하여</summary>

- 코루틴은 유니티에서 비동기적으로 작업을 처리하는데 사용.

- 유니티는 싱글 스레드 기반

- 스타트 코루틴과 옐드 리턴을 사용해 코루틴 관리
</details>

<details>
<summary>C# 가비지 컬렉터와 유니티 GC의 차이</summary>

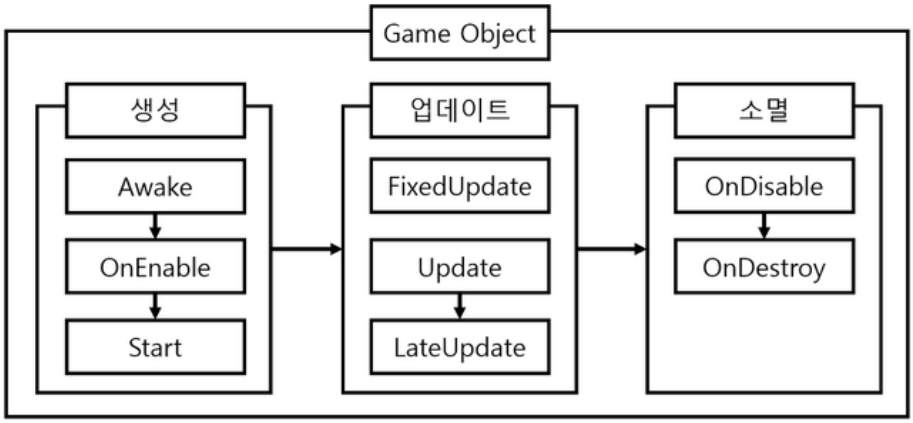
- C#은 닷넷 앱에서 자동 처리, 메모리 할당과 해제의 복잡성을 추상화, 다양한 타입의 앱에서 효율적으로 작동하도록 설계
</details>

- 유니티 쥐시는 게임 개발에 특화되어 있어, 게임의 프레임 속도와 성능을 유지하는 것이 중요, 유니티 쥐씨는 덜 빈번하게 실행되어 프레임 드랍을 최소화함

- 발생 시기 차이

- 시샵은 시스템 메모리 사용량과 CPU 로드에 따라 실행, 유니티는 개발자가 실행 시기를 더 세밀하게 제어 가능, 중요하지 않은 순간에 수동으로 가비지 컬렉션 트리거 가능
System.GC.Collect()

Start와 Awake 차이 (스타코1)



Awake는 생성 시점에서 한번 불려지고, Enable 상태에 따른 중복 호출이 되지 않습니다. Start는 OnEnable 이후에 불려지기 때문에 중복 호출이 될 수 있습니다.

생성 시점에서 단 한번만 불려져야 되는 경우는 Awake, UI 출현 연출 처럼 활성화 시점에서 불려져야되는 로직은 OnEnable에 넣어줍니다.

싱글턴 매니저에 대해서

시리얼라이제이션 (스타코1)

직렬화와 역직렬화는 바이트를 객체(데이터)간 변환하는 관계에서 사용되는 개념입니다.

객체->바이트는 시리얼라이제이션

바이트->객체는 디시리얼라이제이션이라 합니다.

유니티에서의 시리얼라이제이션은 [유니티에서 런타임 또는 에디터에서 사용 가능하게 코드를 직렬화 하는 것]으로

두가지 조건이 있습니다.

1) public

-

2) Serialized Field

- private 필드의 직렬화를 원할때 사용

- 구조체는 4.5 이상부터 가능

X) 안되는 것 (추상 클래스는 안됨)

- static, const, readonly

- 값이 정해져 있거나, 공용으로 사용 가능한 영역이거나, 수정이 불가하므로

delegate와 action, 이벤트 차이 (스타코1) // + Func

- [delegate]

- 메서드에 대한 “참조 형식” (특정 매개변수 목록과 반환 형식이 있는 함수의)

- 메서드를 다른 메서드에 인수로 전달하는데 사용

[action]

- 매개 변수가 1개이고, 값을 반환하지 않는 메서드 (void)를 캡슐화

- += 으로 등록 가능하며, -=을 미리 해주는 습관이 좋음

[func]

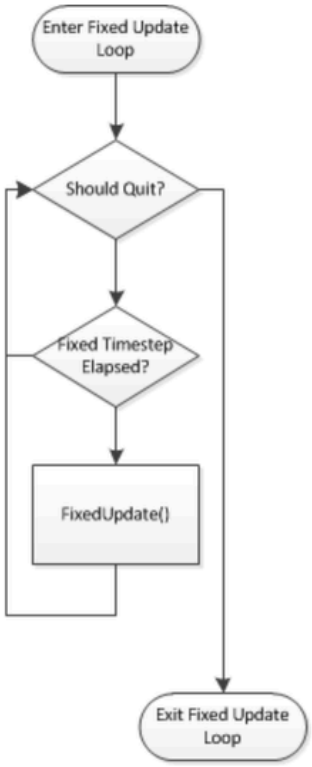
- 리턴 타입이 반드시 존재

FixedUpdate, LateUpdate, Update

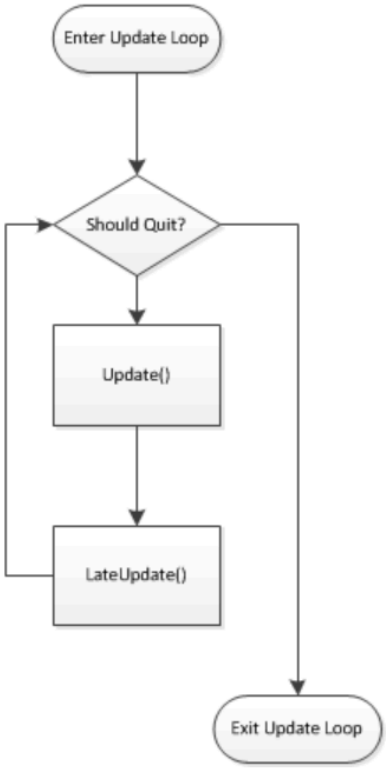
- Update()

  - Enable 상태일 때 [매 프레임]마다 호출

Fixed Update Loop



- FixedUpdate()
  - 설정된 값에 따라 일정 간격으로 호출
  - Rigidbody 같은 오브젝트 조정 시 사용
- Update는 불규칙한 호출로 물리엔진 충돌검사 등이 제대로 안될 수 있음



- LateUpdate()

  - 모든 Update 함수가 호출된 후 마지막에 호출

  - Ex) 오브젝트를 따라가게 설정한 카메라 같은 때 사용

UniTask

위의 예제에서 보았듯이 코루틴에서 UniTask 를 이용한 방식으로 구현하는 것도 크게 어렵지 않기 때문에 가급적 UniTask 로 대체하는 게 좋아 보입니다. 아래는 코루틴에서 사용하는 yield return... 을 대체하는 UniTask 함수들입니다.

yield return new WaitForSeconds/WaitForSecondsRealtime	await UniTask.Delay
yield return null	await UniTask.Yield await UniTask.NextFrame
yield return WaitForEndOfFrame	await UniTask.WaitForEndOfFrame
new WaitForFixedUpdate	await UniTask.WaitForFixedUpdate
yield return WaitUntil	await UniTask.WaitUntil

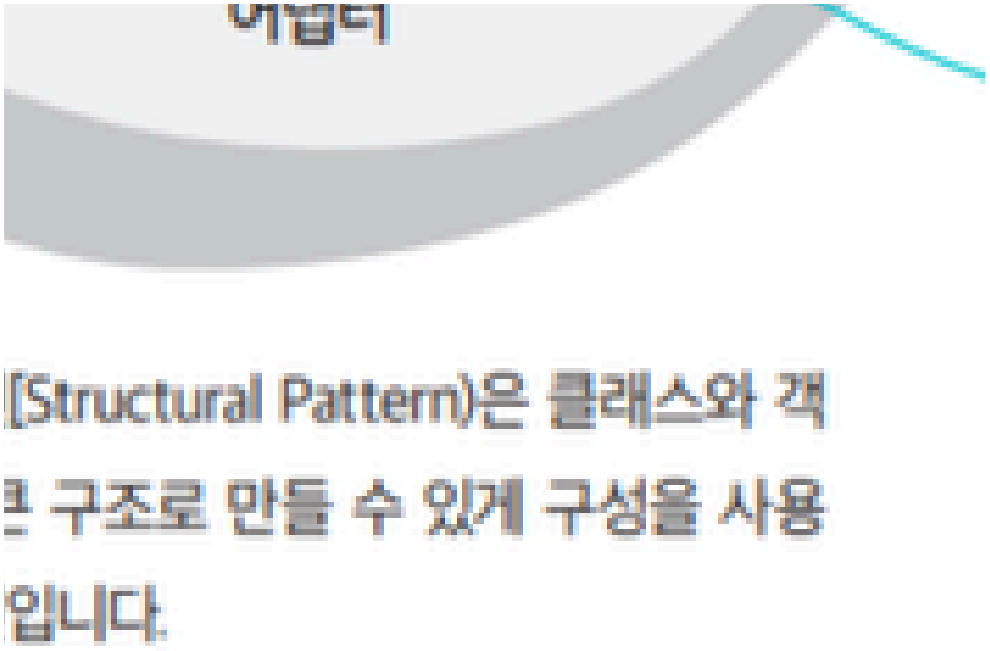
이 외에도 UniTask 는 더 많은 기능을 제공하고 있습니다.

메모리 사용량, 성능, try catch와 return 사용 가능

클라 - CBD

클라 - 디자인 패턴


템플릿 메
인터프리
구조
프록시
이터
퍼사드
컴포지트
브리지
이웨이트



[Design pattern] 많이 쓰는 14가지 핵심 GoF 디자인 패턴의 종류

디자인 패턴의 3가지 유형 패턴에 대해 설명하시오.

디자인 패턴은 크게 [생성] [구조] [행위] 패턴으로 나누어져 있습니다.

컨스트럭트, 스트럭쳐, 비헤이비어랄

생성 패턴의 대표적인 패턴을 설명하시오.

싱글턴, 팩토리, 추상 팩토리(앱스트렉 팩토리) 패턴이 있습니다.

1. 싱글턴은 생성패턴의 종류로 객체를 하나만 생성하고, 생성 객체는 어디서든 참조할 수 있습니다.
2. 팩토리 패턴은 객체 생성 처리를 서브 클래스로 분리해 처리합니다. (캡슐화 관련)
3. 추상 팩토리는 [구체적인 클래스]에 의존하지 않습니다. 서로 연관되거나 의존적 객체들이 조합을 만드는 [인터페이스]를 제공합니다.

[TODO] 추상팩토리 추가 공부하기

행위 패턴의 대표적인 패턴을 설명하시오.

[커맨드] [옵저버] [스테이트] 패턴이 있습니다.

1. [커맨드] 패턴은 Job 같이 실행될 기능을 캡슐화해 주어진 여러 기능을 수행할 수 있게 합니다. 재사용성이 용이합니다.
2. [옵저버] 패턴은 한 객체의 상태 변화시 다른 객체의 상태도 연동합니다.
3. [스테이트] 패턴은 상태에 따라 객체의 행위 내용을 변경합니다.

클라 - 자료구조

자료구조의 유형과 유형에 따른 대표 자료구조를 말해보시오

크게 단순, 선형, 비선형, 파일 자료구조로 되어있으며, 단순 자료구조에는 [정수, 실수, 문자, 문자열]이 대표적이며, 선형 자료구조는 [순차 리스트, 연결 리스트, 스택, 큐, 덱]이 있습니다.

비선형 자료구조는 [트리]와 [그래프]가 대표적입니다.

스택과 큐의 차이에 대해 말해보시오.

스택은 선입 후출, 큐는 선입 선출

트리란 무엇인가?

계층적 구조의 데이터를 표현하기 위한 자료구조입니다. 부모와 자식 노드로 구성되어 있습니다.

이진 트리란 무엇인가?

최대 2개의 자식노드가 존재하는 트리입니다.

순회의 3가지 방식에 대해 설명해보시오.

전위 순위, 중위 순위, 후위 순위가 있으며, 전위는 루트 먼저, 중위는 왼쪽 서브트리 먼저, 후위는 루트를 기준으로 왼쪽 오른쪽 먼저 검색합니다.

- 그래프 질문 보강하

자료구조의 파일 유형에 대해 설명 해보시오.

순차, 색인, 직접 파일로 구성되어 있습니다.

C# 자료구조에서 제네릭과 논제네릭은 무엇인가?

제네릭 사용 시 T형식을 사용할 수 있습니다. 제네릭으로 딕셔너리, 리스트, 큐, 스택이 대표적이며

비 제네릭으로는 딕셔너리에 대응되는 해시테이블, 리스트에 대응되는 어레이와 어레이리스트, 큐와 스택은 이름은 동일하지만 뒤에 타입이 명시되면 제네릭입니다.

박싱 언박싱에 대해 설명해보시오

밸류 타입은 스택에, 레퍼런스 타입은 힙 메모리에 쌓입니다.

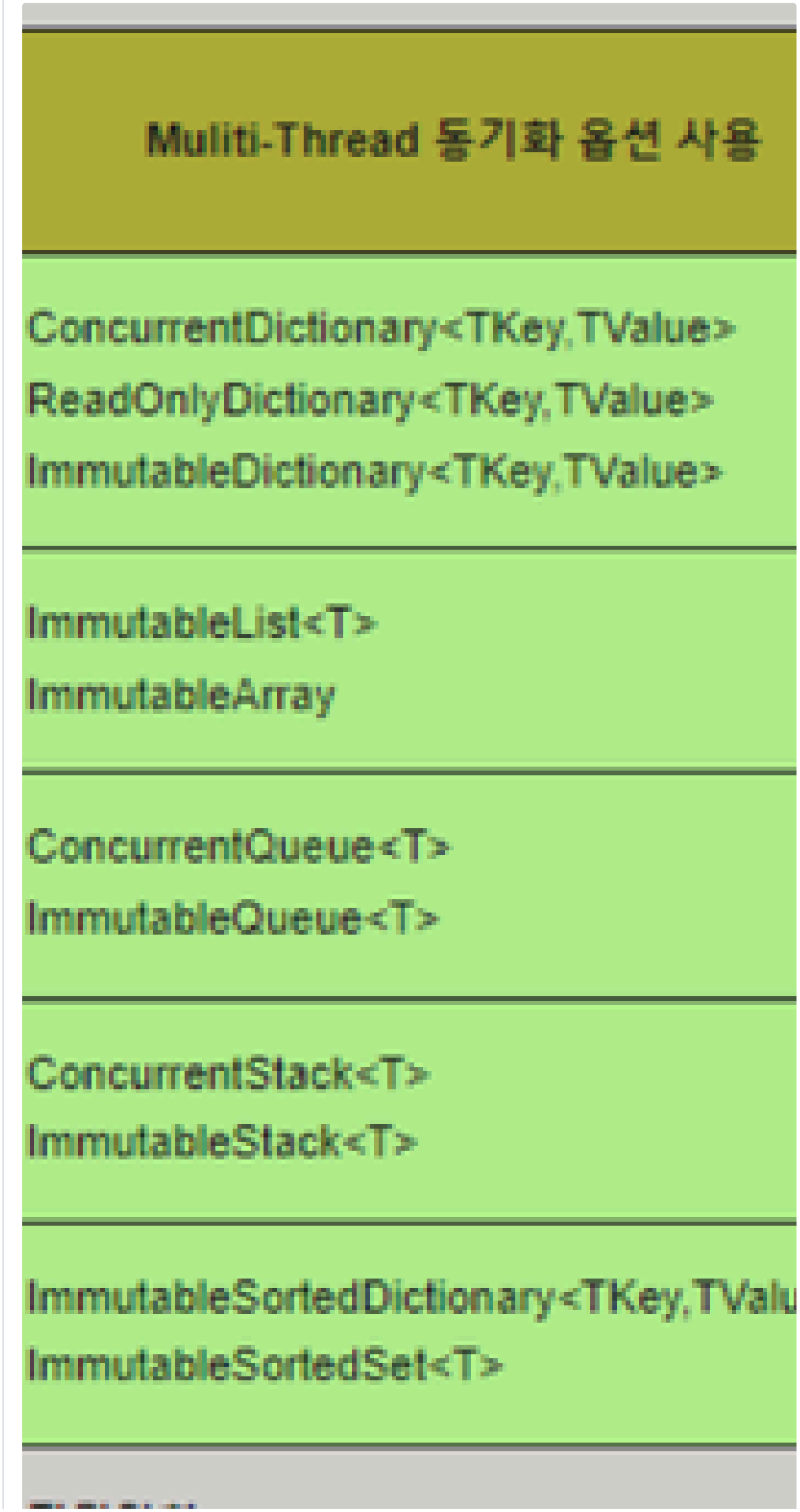
참조 타입은 값타입과 달리 선언해도 메모리가 생성되지 않습니다.

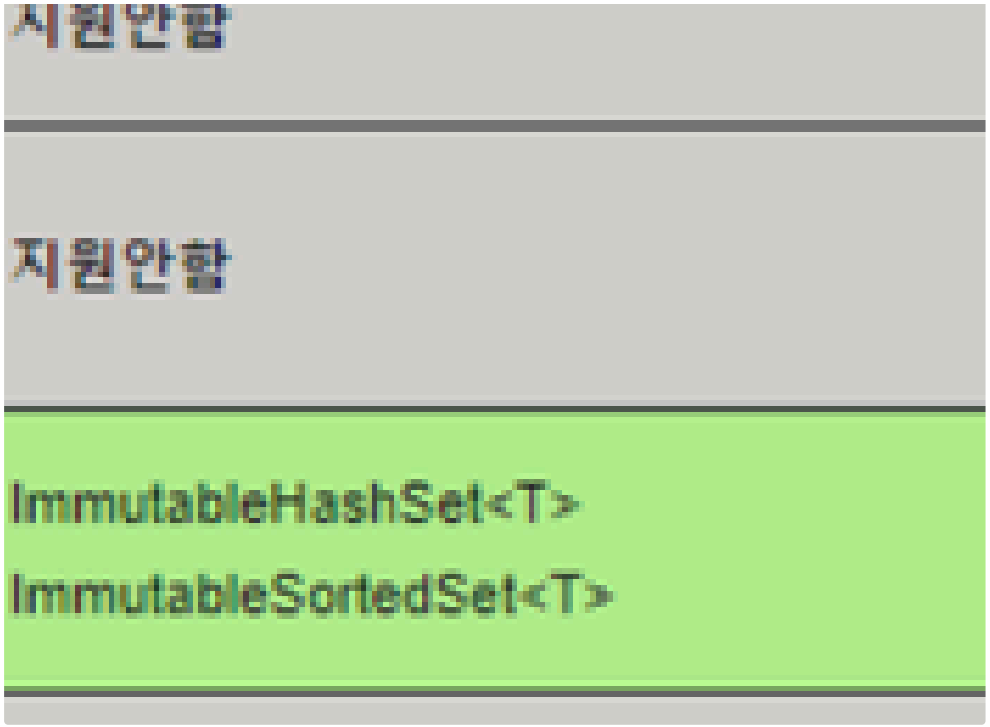
박싱은 밸류 타입을 레퍼 타입으로 변환할 때를 의미하며

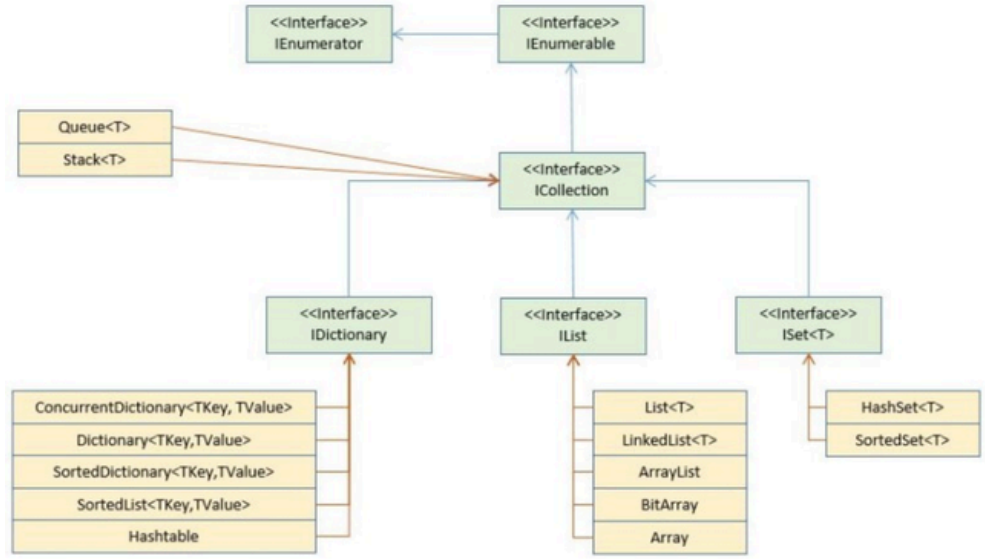
언박싱은 레퍼 타입을 값타입으로 변환할 때를 의미합니다.

이를 방지하기 위해 [제네릭]을 사용해 처리해야합니다.

또는 IS 캐스팅을 사용합니다.


Muliti-Thread 동기화 옵션 사용
ConcurrentDictionary<TKey,TValue>
ReadOnlyDictionary<TKey,TValue>
ImmutableDictionary<TKey,TValue>
ImmutableList<T>
ImmutableArray
ConcurrentQueue<T>
ImmutableQueue<T>
ConcurrentStack<T>
ImmutableStack<T>
ImmutableSortedDictionary<TKey,TValu
ImmutableSortedSet<T>





클라 - 자료구조 - 빅오 표기법

빅오의 일반적인 순서 대해 설명하시오.

O1 → logn → n → nlongn → n2(제곱) → 2n 순입니다.

속도별 대표적인 자료구조나 알고리즘을 설명하시오.

O1)

- 스택에서의 푸쉬, 팝

- 리스트의 인덱스 검색

- 딕셔너리 (C#)

logn) - 이진 트리

n) - for문

nlongn) - 퀵정렬, 병합정렬, 힙정렬 (퀵, 머지, 힙 소트)

n제곱) - 이중for문, 삽입정렬, 버블소트, 선택정렬

o2제곱 - 피보나치 수열

클라 - 알고리즘

정렬의 종류와 특징을 설명하시오

버블정렬, 선택정렬, 삽입정렬, 퀵정렬, 병합 정렬이 있습니다.

1. 버블 정렬은 [인접 데이터를 비교하면서 정렬] 합니다. 빅오는 n제곱 입니다.

2 .선택 정렬은 최소값을 찾아 순서대로 정렬하며, 빅오는 n제곱입니다.

3. 삽입 정렬은 앞 데이터와 비교해 적은 값을 찾을때까지 뒤로 밀어냅니다. 빅오는 n제곱 입니다.

4. 퀵 정렬은 [기준점을 기준으로 작으면 좌, 크면 우]로 정렬합니다.

- 재귀적 구현이 가능하며, n log n 속도가 나옵니다.

5. 병합 정렬 (머지 소트)는 [분할 정복 알고리즘]으로 리스트를 절반으로 잘라 비슷한 크기의 두 부분으로 나눈 후 재귀적으로 합병해 정렬하는 알고리즘 입니다. n log n 속도가 나옵니다.

n log n의 속도가 나오는 정렬은 무엇인가?

퀵 정렬과 병합 정렬 입니다.

분할 정복 알고리즘이 적용되는 대표적인 정렬은 무엇인가?

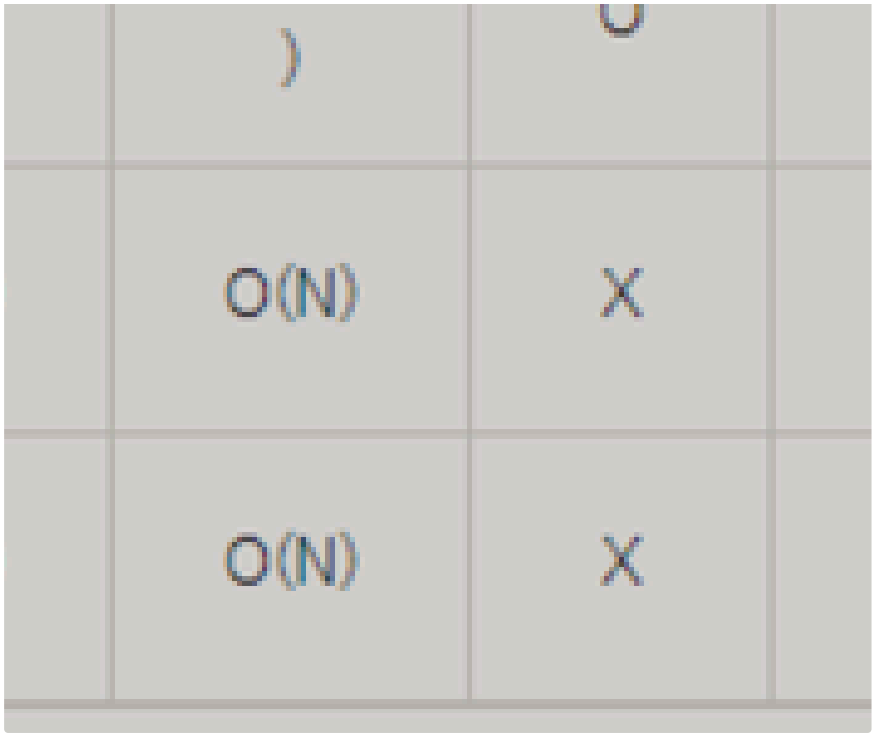
병합 정렬(머지 소트)입니다.

탐색 알고리즘의 종류와 특징을 설명하시오.

이진 탐색과 순차 탐색이 있습니다. 이진 탐색은 자료를 반으로 나눠 있을만한 곳을 탐색하고, 순차 탐색은 앞에서 하나하나 찾습니다.

Algorithm - 알고리즘 핵심정리

	최악	in-place	s
2)	O(N^2)	O	
2)	O(N^2)	O	
2)	O(N^2)	O	
gN	O(NlogN)	X	
gN	O(N^2)	O	
gN	O(NlogN		



클라 - 알고리즘

그래프의 탐색 두 방식에 대해 설명하시오.

[BFS]와 [DFS]가 있습니다.

1. BFS는 너비 우선 탐색으로 [정점과 같은 형제노드부터 탐색]합니다.

한발 한발 난이도대로 클리어하는 느낌으로 큐를 사용합니다. (가장 오래된 애가 먼저 나옴)

2. DFS는 깊이 우선 탐색으로 [정점의 자식부터 탐색]합니다. 재귀적으로 인접부터 확인하면서 방문하지 않는 경우를 찾습니다. 보스부터 잡고 나오는 느낌입니다.

최단 경로 알고리즘의 종류를 설명하시오.

최단 경로 알고리즘으로는 숏 패스, 단일 출발, 전체 쌍, 다익스트라, 최소신장트리 알고리즘이 있습니다.

다익스트라 알고리즘에 대해 설명하시오.

최단 경로 알고리즘 중 [단일 출발]에 해당하며, 가중치가 있으면 다익스트라로. BFS와 비슷하지만 다릅니다. 에이스타 알고리즘과 연관되어 있습니다.

다이나믹 프로그래밍을 활용한 대표적인 최단 경로 탐색 알고리즘 입니다.

그리디 알고리즘이 적용되는 대표적인 알고리즘입니다.

클라 - 문제 해결 전략

문제 해결 전략의 대표적인 4가지 유형에 대해 설명하시오.

재귀호출, 동적계획, 분할정복, 탐욕법이 있습니다.

재귀 호출이란 무엇인가?

함수 안에서 함수를 계속 호출합니다.

동적 계획에 대해 설명하시오.

다이나믹 프로그래밍으로 하나의 문제 해결을 위해 작은 문제로 나누어 전체 문제를 해결합니다.

메모제이션 기법을 사용해 한번 연산된 값은 다시 연산하지 않습니다. (중복이 발생함)

분할 정복에 대해 설명하시오.

작은 문제로 나누어 하위 문제를 해결한 후 다시 병합해 상위 문제의 답을 얻습니다.

다이나믹 프로그래밍과의 차이점은 [나눠진 부분문제에 중복이 없습니다]

하향식 접근으로 메모제이션을 사용하지 않습니다.

대표적으로 머지 소팅이 있습니다.

탐욕법에 대해 설명하시오.

그리디로 [최적의 해에 가까운 값을 구하기 위해 매 순간에 최선을 선택합니다]

[TODO] 각 문제해결 전략의 장단점 정리하기

면접 질문 총괄

-