기술 주의자 v 논리 주의자

김상현

Software Architect

001

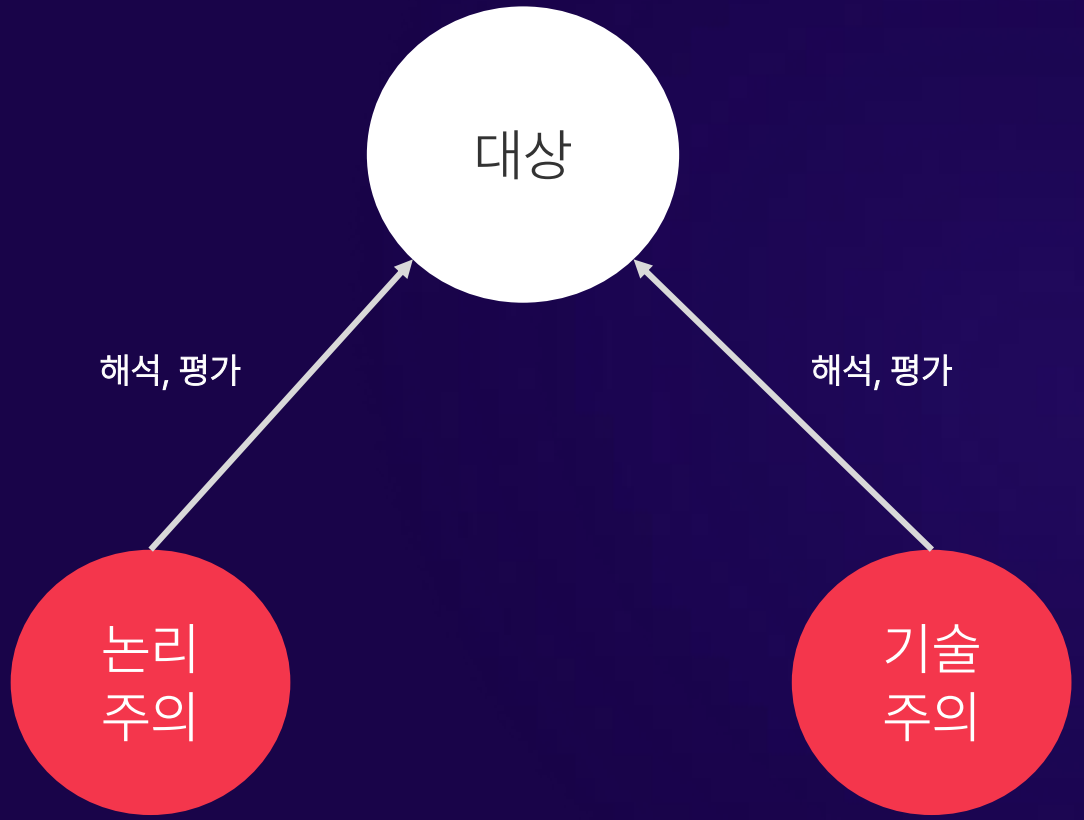
기술 주의 v 논리 주의

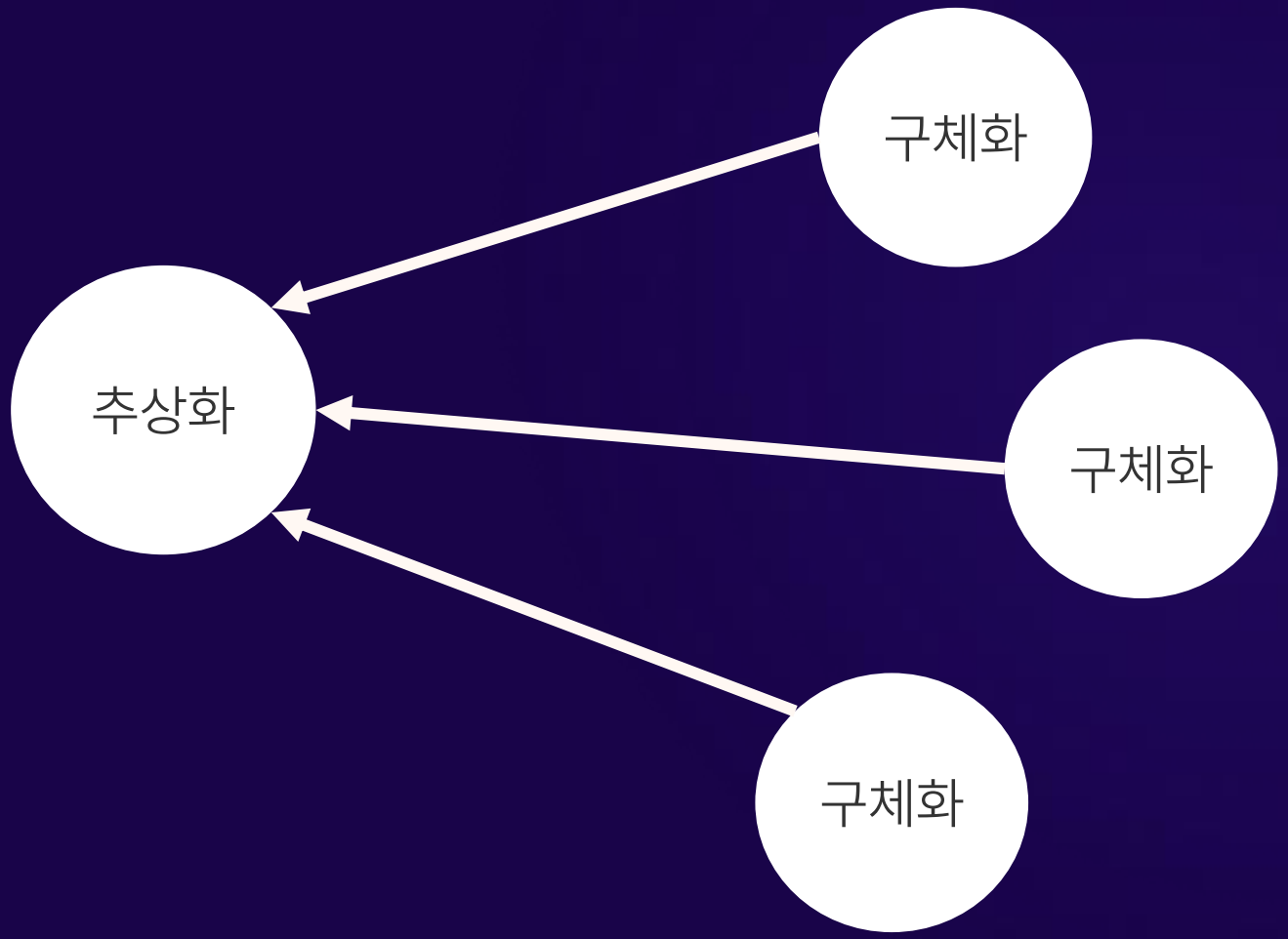
기술 주의

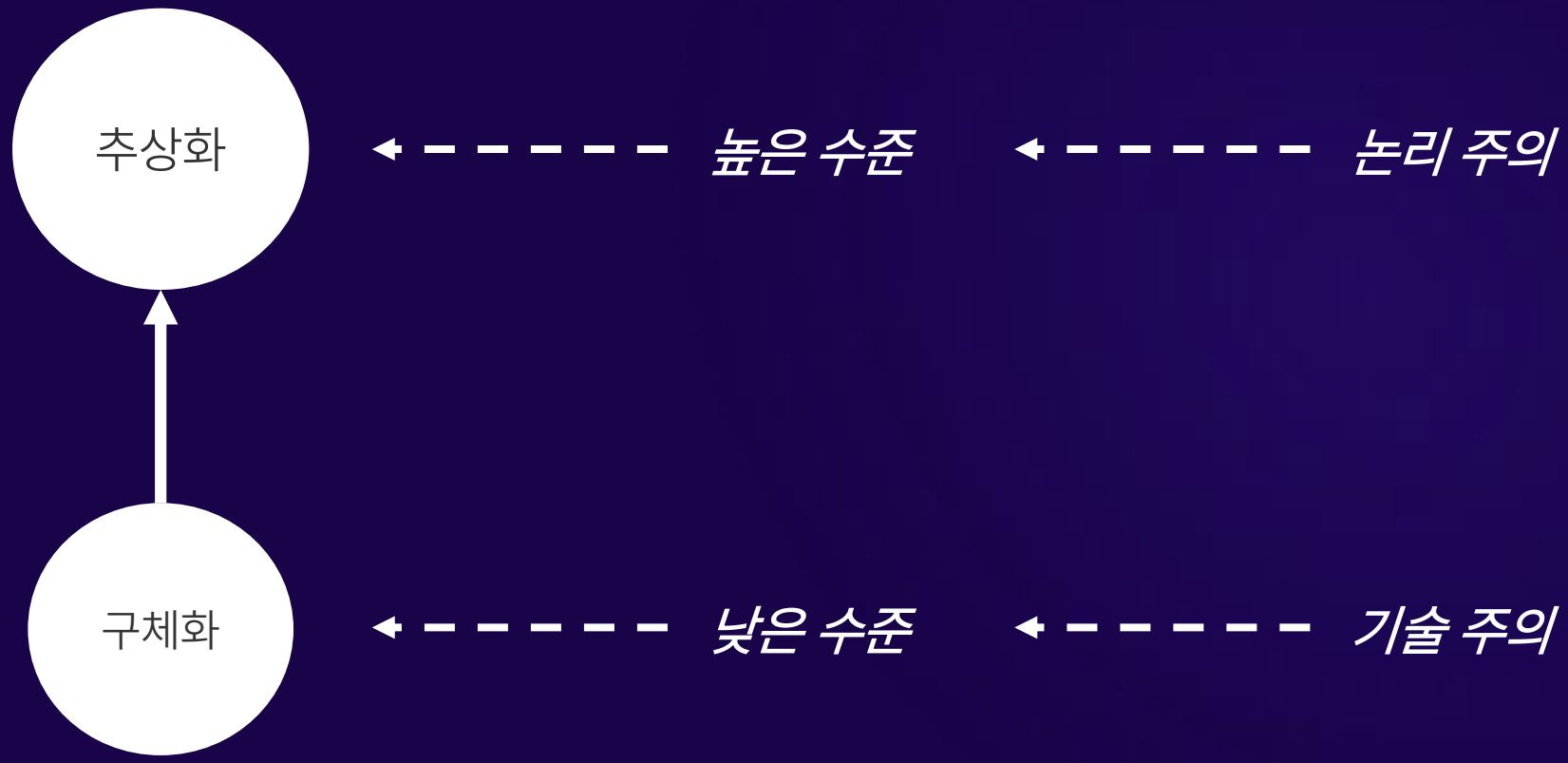
대상을 물리적인 기술에 기준하여 해석하고 평가하는 사고 방식

논리 주의

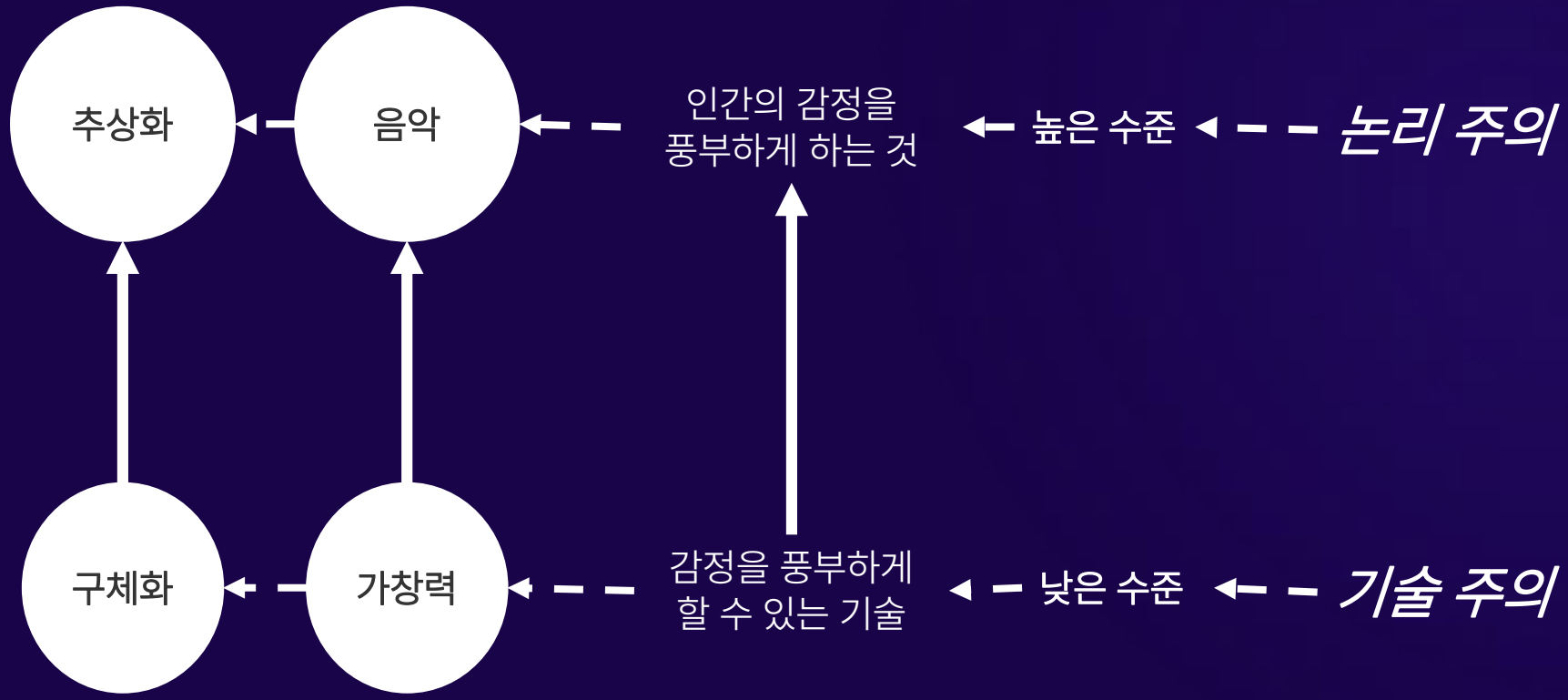
대상을 추상적인 논리에 기준하여 해석하고 평가하는 사고 방식


대상
해석, 평가
해석, 평가
논리
주의
기술
주의


구체화
추상화
구체화
구체화


추상화
높은 수준
논리 주의
구체화
낮은 수준
기술 주의

대상: 누가 더 뛰어난 가수인가?



Command Pattern

개체 자체가 명령이다

```
Class SaveCommand
{
        void Execute()
        {
                ...
        }
}
```

```
Class Save
{
        void Run()
        {
                ...
        }
}
```

연속 통합 (CI)

작업자들의 결과물을 매우 자주 통합하자는 것

빌드 자동화 사용

Github Actions

빌드 자동화 사용 안함

모든 팀원이 Task를 하루 안에
끝낼 수 있는 양으로 쪼개어 매일 병합.

002

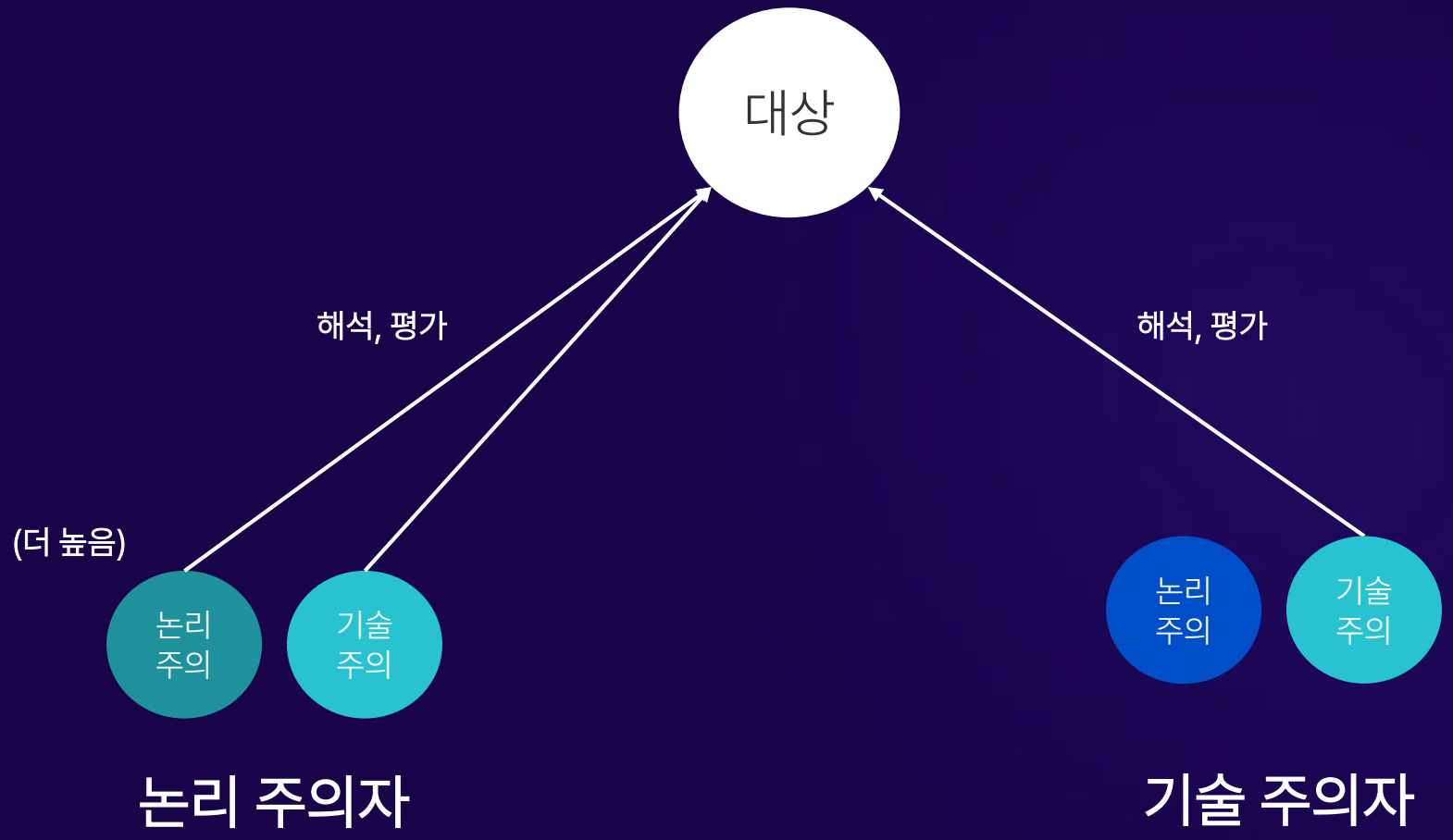
기술 주의자 v 논리 주의자

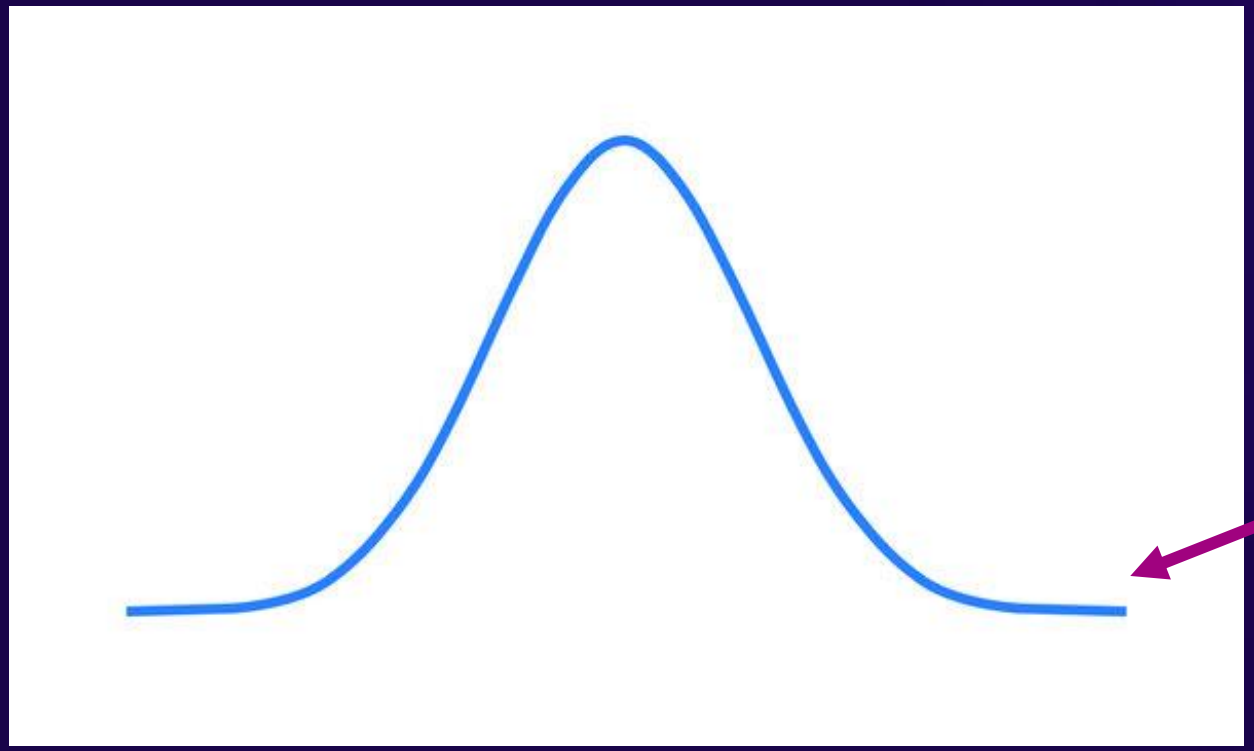
논리 주의자

대상을 기술 주의, 논리 주의에 기준하여 해석하고 평가하는 자

기술 주의자

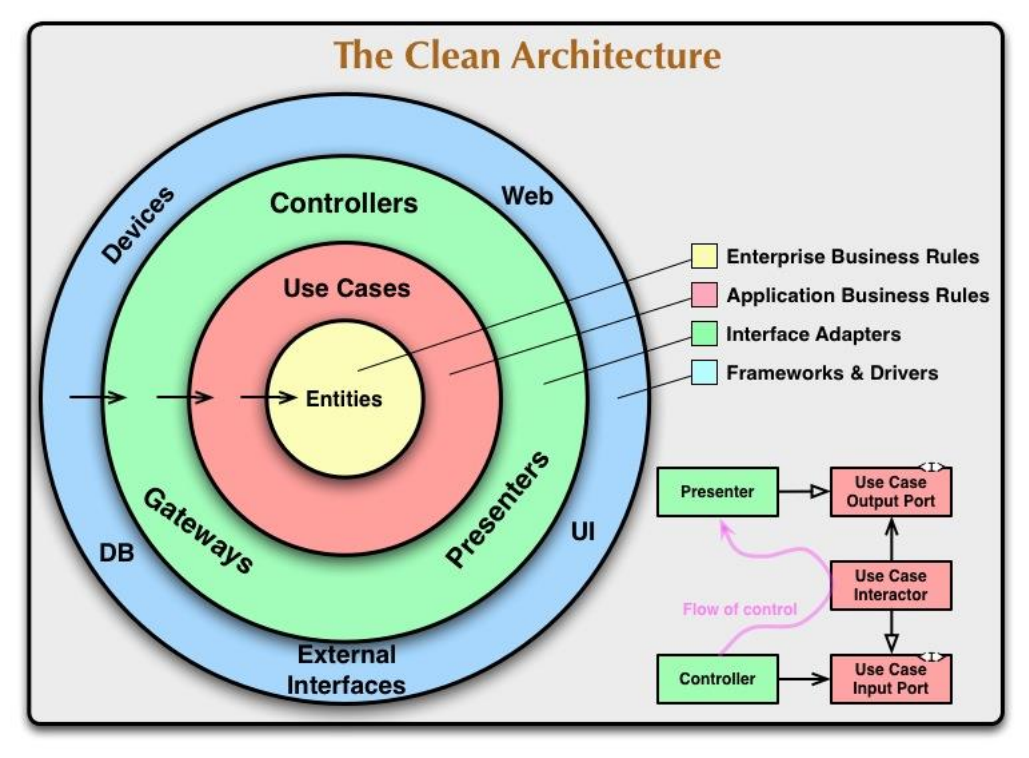
대상을 기술 주의에만 기준하여 해석하고 평가하는 자

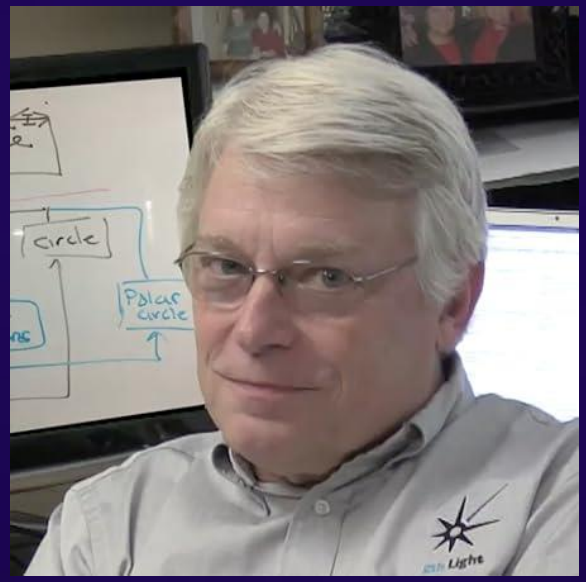

대상
해석, 평가
해석, 평가
(더 높음)
논리
주의
기술
주의
논리
주의
기술
주의
논리 주의자
기술 주의자



논리 주의자들

Clean Architecture





Robert C. Martin

003

계산하는 방법 v 계산된 결과

논리 주의

분할 v 비분할

기술기반 분할 v 도메인 기반 분할

강한 결합 v 낮은 결합

연속 통합 v 드문 통합

계산하는 방법 v 계산된 결과

... +@

계산하는 방법

소비 측에 계산하는 방법을 제공함

계산된 결과

소비 측에 계산된 결과 값을 제공함

도메인에 따라 유리한 걸 선택해야 한다.

계산하는 방법

- 공간 down, 시간 up
- 결과 업데이트 관리 필요 없음
- 많은 조회에 불리함
- ...

계산된 결과

- 공간 up, 시간 down
- 결과 업데이트 관리 필요함
- 많은 조회에 유리함
- ...

```
6 references
public class User
{
    3 references
    public string Name { get; }

    3 references
    public DateOnly BirthDate { get; private set; }

    0 references
    public User(string name, DateOnly birthDate)
    {
        this.Name = name;
        this.BirthDate = birthDate;
    }

    0 references
    public void UpdateBirthDate(DateOnly birthDate)
    {
        this.BirthDate = birthDate;
    }

    0 references
    public uint GetAge()
    {
        var today = DateOnly.FromDateTime(DateTime.Today);
        return (uint)(today.Year - this.BirthDate.Year);
    }
}
```

필드 — 계산된 결과

함수 — 계산하는 방법

Age 제공: 계산하는 방법

```
6 references
public class User
{
    3 references
    public string Name { get; }

    3 references
    public DateOnly BirthDate { get; private set; }

    0 references
    public User(string name, DateOnly birthDate)
    {
        this.Name = name;
        this.BirthDate = birthDate;
    }

    0 references
    public void UpdateBirthDate(DateOnly birthDate)
    {
        this.BirthDate = birthDate;
    }

    0 references
    public uint GetAge()
    {
        var today = DateOnly.FromDateTime(DateTime.Today);
        return (uint)(today.Year - this.BirthDate.Year);
    }
}
```

Age 제공: 계산된 결과

```
1 reference
public class User
{
    1 reference
    public string Name { get; }

    2 references
    public uint Age { get; private set; }

    2 references
    public DateOnly BirthDate { get; private set; }

    0 references
    public User(string name, DateOnly birthDate)
    {
        this.Name = name;
        this.BirthDate = birthDate;
        this.Age = CalculateAge(birthDate);
    }

    0 references
    public void UpdateBirthDate(DateOnly birthDate)
    {
        this.BirthDate = birthDate;
        this.Age = CalculateAge(birthDate);
    }

    2 references
    private static uint CalculateAge(DateOnly birthDate)
    {
        var today = DateOnly.FromDateTime(DateTime.Today);
        return (uint)(today.Year - birthDate.Year);
    }
}
```

배열

계산된 결과

LINQ

계산하는 방법

```
0 references
public static void 계산된_결과()
{
    int[] values = [1, 2, 3, 4, 5];

    int value = values[0];
}

0 references
public static void 계산하는_방법()
{
    IEnumerable<int> values = Enumerable.Range(1, 5);

    int value = values.ElementAt(0);
}
```

```
0 references
59  public void WriteUsers()
60  {
61      // 1. 계산하는 방법으로
62      IEnumerable<User> users = this._userRepository.GetUsers();
63
64      foreach (User user in users)
65      {
66          Console.WriteLine($"{user.Name} is {user.Age} years old.");
67      }
68  }
69
0 references
70  public void WriteUsers2()
71  {
72      // 2. 계산된 결과로
73      User[] users = this._userRepository.GetUsers().ToArray();
74
75      foreach (User user in users)
76      {
77          Console.WriteLine($"{user.Name} is {user.Age} years old.");
78      }
79  }
```

ReactiveX (Rx)

Observer Pattern + **Linq** + Scheduler

계산된 결과 -> 계산하는 방법

성능 최적화의 기본

계산하는 방법을

계산된 결과으로

CDN, 스냅샷, Materialized View ...

개체 지향 프로그래밍

계산된 결과

함수 지향 프로그래밍

계산하는 방법

김상현

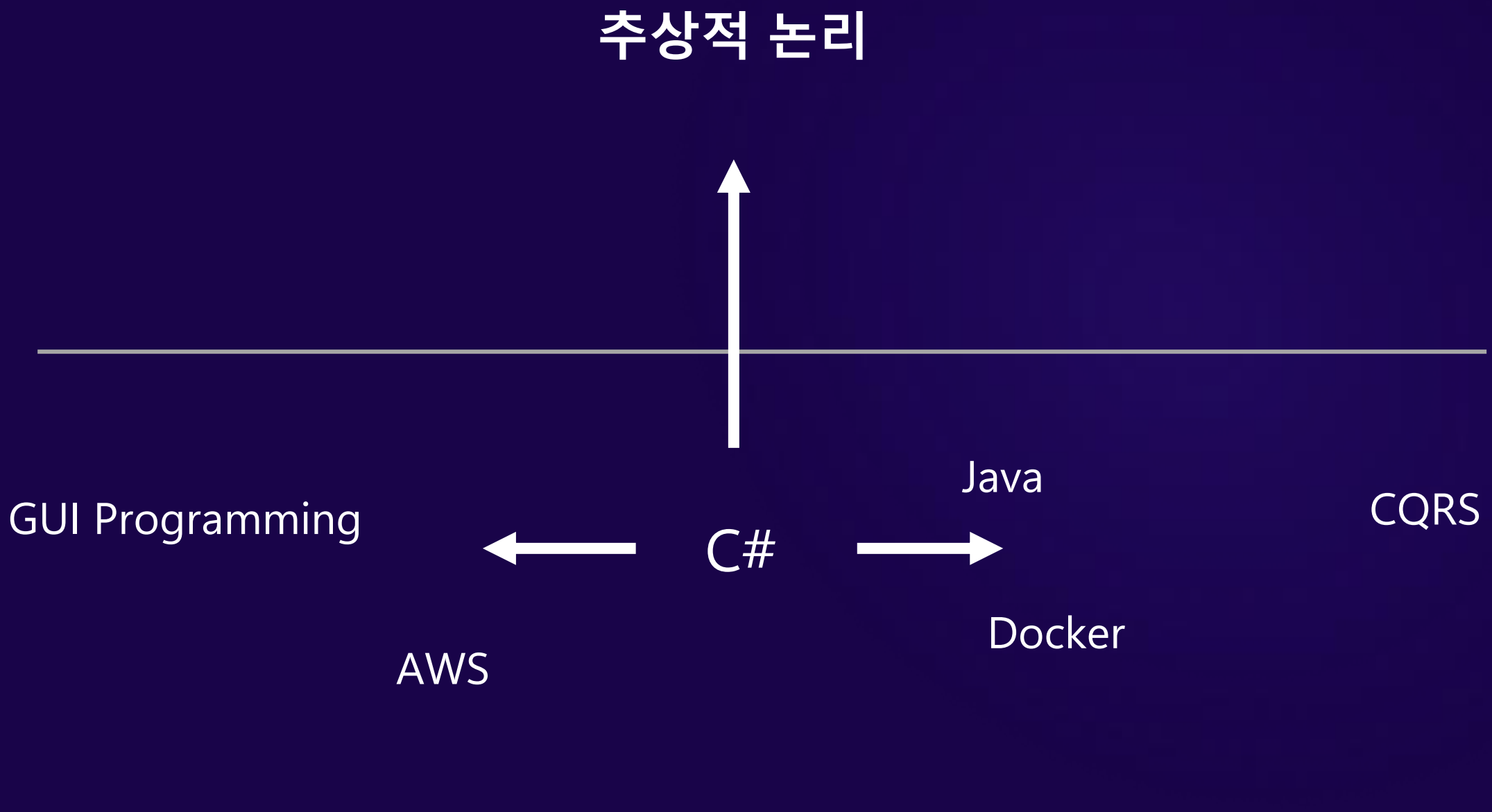
Software Architect

Microsoft 2023 MVP

논리적 프로그래밍

현실적 엔지니어링

LinkedIn


추상적 논리
GUI Programming
AWS
C#
Java
Docker
CQRS

감사합니다

기술 주의자 v 논리 주의자

김상현

Software Architect