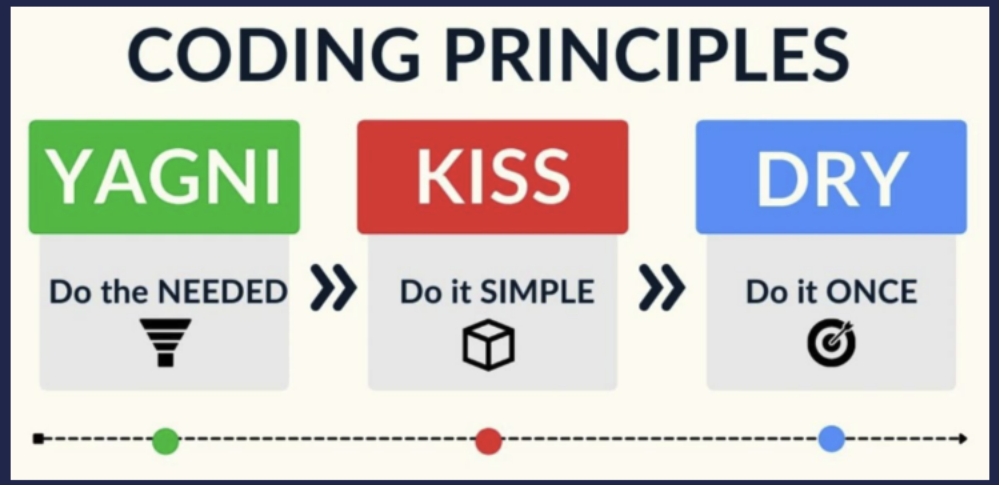
클린코드 시리즈

김하연 튜터
챌린지반 특강 7강

안 좋은 코드란? 4탄

16. 추측성 일반화 (Speculative Generality)

- 불필요하게 복잡하고 확장 가능한 코드
- You Aren't Gonna Need It (YAGNI) 원칙
- 현재 필요하지 않는 기능은 추가하지 말 것



Example 1

Example 2

예제 코드 1

```
// 안 좋은 코드 예시 1
// 거의 사용되지 않는 메서드를 포함한 클래스
3 references
public class DataProcessor
{
    1 reference
    public void ProcessData(Data data) { /* 데이터 처리 로직 */ }
    0 references
    public void ProcessFutureData(Data data) { /* 현재 사용되지 않음 */ }
}
```

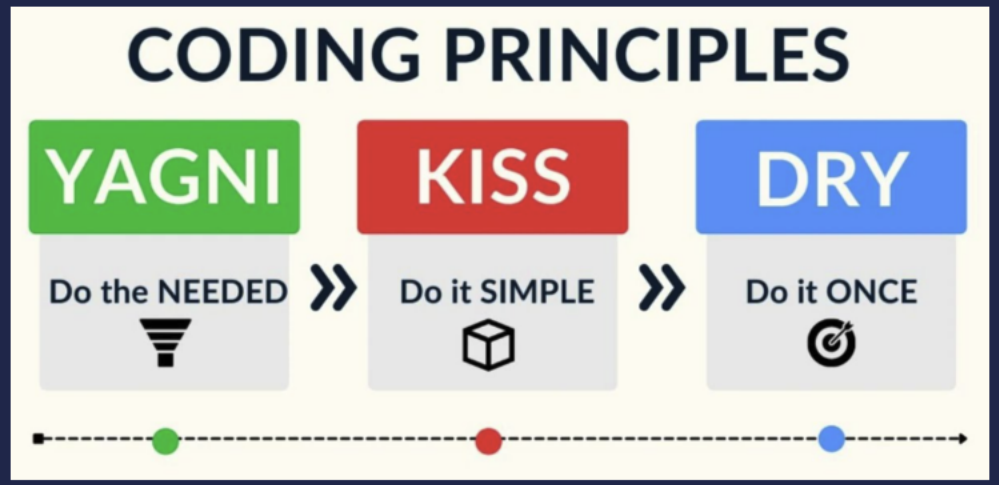
안 좋은 예시

```
// 좋은 코드 예시 1
// 필요한 기능만을 가진 클래스
3 references
public class DataProcessor
{
    1 reference
    public void ProcessData(Data data) { /* 데이터 처리 로직 */ }
}
```

좋은 예시

16. 추측성 일반화 (Speculative Generality)

- 불필요하게 복잡하고 확장 가능한 코드
- You Aren't Gonna Need It (YAGNI) 원칙
- 현재 필요하지 않는 기능은 추가하지 말 것



Example 1

Example 2

예제 코드 2

```
// 안 좋은 코드 예시 2
// 불필요한 위임
1 reference
public class DataHandler
{
    1 reference
    private DataProcessor processor = new DataProcessor();
    0 references
    public void HandleData(Data data) { processor.ProcessData(data); }
}
```

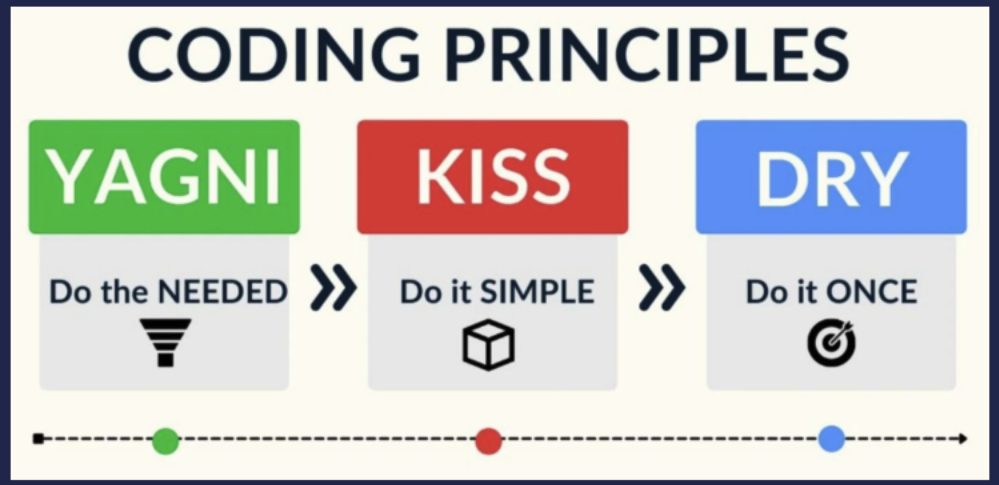
안 좋은 예시

```
// 좋은 코드 예시 2
// DataProcessor의 기능을 직접 사용
1 reference
public class DataHandler
{
    0 references
    public void HandleData(Data data)
    {
        // 데이터 처리 로직을 여기에 직접 구현
    }
}
```

좋은 예시

16. 추측성 일반화 (Speculative Generality)

- 불필요하게 복잡하고 확장 가능한 코드
- You Aren't Gonna Need It (YAGNI) 원칙
- 현재 필요하지 않는 기능은 추가하지 말 것



Example 1

Example 2

클린코드 시리즈

김하연 튜터
챌린지반 특강 7강

안 좋은 코드란? 4탄

17. 긴 매개변수 목록 (Long Parameter List)

- 너무 많은 매개변수는 문제가 있는 것
- 해결법
  - 질의 함수로 바꾸기
  - 객체를 통째로 넘기기
  - 플래그 인수는 제거하기
  - 여러 함수를 클래스로 묶기 (공통되는 매개변수를 클래스 변수로 관리)

Example 1

Example 2

예제 코드 1

```
// 안 좋은 코드 예시 1
5 references
public class Order
{
    3 references
    public decimal OrderTotal { get; set; }
    2 references
    public decimal TaxRate { get; set; }

    1 reference
    public decimal CalculateTotalCost(decimal orderTotal, decimal taxRate)
    {
        return orderTotal + (orderTotal * taxRate);
    }
}

// 사용 예시
Order order = new Order();
decimal totalCost = order.CalculateTotalCost(order.OrderTotal, order.TaxRate);
```

안 좋은 예시

```
// 좋은 코드 예시 1 (질의 함수로 바꿈)
5 references
public class Order
{
    3 references
    public decimal OrderTotal { get; set; }
    2 references
    public decimal TaxRate { get; set; }

    2 references
    public decimal CalculateTotalCost()
    {
        return OrderTotal + (OrderTotal * TaxRate);
    }
}

// 사용 예시
Order order = new Order();
decimal totalCost = order.CalculateTotalCost();
```

좋은 예시

17. 긴 매개변수 목록 (Long Parameter List)

- 너무 많은 매개변수는 문제가 있는 것
- 해결법
  - 질의 함수로 바꾸기
  - 객체를 통째로 넘기기
  - 플래그 인수는 제거하기
  - 여러 함수를 클래스로 묶기 (공통되는 매개변수를 클래스 변수로 관리)

Example 1

Example 2

예제 코드 2

```
// 안 좋은 코드 예시 2
public void RenderDocument(string document, bool isPDF)
{
    if (isPDF)
    {
        // PDF로 렌더링
    }
    else
    {
        // 다른 형식으로 렌더링
    }
}
```

안 좋은 예시

```
// 좋은 코드 예시 2 (플래그 인수 제거하기)
public void RenderDocumentAsPDF(string document)
{
    // PDF로 렌더링
}

public void RenderDocumentAsOtherFormat(string document)
{
    // 다른 형식으로 렌더링
}
```

좋은 예시

17. 긴 매개변수 목록 (Long Parameter List)

- 너무 많은 매개변수는 문제가 있는 것
- 해결법
  - 질의 함수로 바꾸기
  - 객체를 통째로 넘기기
  - 플래그 인수는 제거하기
  - 여러 함수를 클래스로 묶기 (공통되는 매개변수를 클래스 변수로 관리)

Example 1

Example 2

클린코드 시리즈

김하연 튜터
챌린지반 특강 7강

안 좋은 코드란? 4탄

18. 거대한 클래스 (Large Class)

- **한 클래스가 너무 많은 책임을 가지면, 코드를 복잡하게 만든다.**
- 중복된 코드가 생겨나고, 클래스 내의 필드와 메소드 간의 연관성이 낮아서, 응집력도 떨어진다. (객체 지향 프로그래밍에서 응집력도 중요한 요소 중 하나)

Example 1

Example 2

예제 코드 1

```
// 안 좋은 코드 예시 1
1 reference
public class Employee
{
    0 references
    public string Name { get; set; }
    0 references
    public string Department { get; set; }
    0 references
    public string Address { get; set; }
    0 references
    public string Phone { get; set; }
    0 references
    public string Email { get; set; }
    // 기타 많은 필드...

    0 references
    public void CalculatePay() { /* 급여 계산 로직 */ }
    0 references
    public void GenerateReport() { /* 보고서 생성 로직 */ }
    // 기타 많은 메소드...
}
```

안 좋은 예시

```
// 좋은 코드 예시 1
1 reference
public class Employee
{
    0 references
    public string Name { get; set; }
    0 references
    public ContactInfo ContactInfo { get; set; }

    0 references
    public void CalculatePay() { /* 급여 계산 로직 */ }
}

1 reference
public class ContactInfo
{
    0 references
    public string Department { get; set; }
    0 references
    public string Address { get; set; }
    0 references
    public string Phone { get; set; }
    0 references
    public string Email { get; set; }
}
```

좋은 예시

18. 거대한 클래스 (Large Class)

- **한 클래스가 너무 많은 책임을 가지면, 코드를 복잡하게 만든다.**
- 중복된 코드가 생겨나고, 클래스 내의 필드와 메소드 간의 연관성이 낮아서, 응집력도 떨어진다. (객체 지향 프로그래밍에서 응집력도 중요한 요소 중 하나)

Example 1

Example 2

예제 코드 2

```
// 안 좋은 코드 예시 2
1 reference
public class ReportGenerator
{
    0 references
    public void GenerateSalesReport(SalesData data)
    {
        // 데이터 검증 로직
        // 보고서 생성 로직
    }

    0 references
    public void GenerateEmployeeReport(EmployeeData data)
    {
        // 데이터 검증 로직 (중복)
        // 보고서 생성 로직
    }
}
```

안 좋은 예시

```
// 좋은 코드 예시 2
1 reference
public class ReportGenerator
{
    0 references
    public void GenerateSalesReport(SalesData data)
    {
        ValidateData(data);
        // 보고서 생성 로직
    }

    0 references
    public void GenerateEmployeeReport(EmployeeData data)
    {
        ValidateData(data);
        // 보고서 생성 로직
    }

    2 references
    private void ValidateData(object data)
    {
        // 데이터 검증 공통 로직
    }
}
```

좋은 예시

18. 거대한 클래스 (Large Class)

- **한 클래스가 너무 많은 책임을 가지면, 코드를 복잡하게 만든다.**
- 중복된 코드가 생겨나고, 클래스 내의 필드와 메소드 간의 연관성이 낮아서, 응집력도 떨어진다. (객체 지향 프로그래밍에서 응집력도 중요한 요소 중 하나)

Example 1

Example 2

클린코드 시리즈

김하연 튜터
챌린지반 특강 7강

안 좋은 코드란? 4탄

19. 메시지 체인 (Message Chain)

- **클래스도 프라이버시가 있다.**
- 클라이언트 코드에서 객체의 내부 구조에 대해서 지나치게 많이 아는 것은 좋은 것이 아님.
- 클래스끼리 강한 결합이 있는 것은 피하자.

Example

예제 코드

```
// 안 좋은 코드 예시
5 references
public class Customer
{
    1 reference
    public Account Account { get; set; }
}

3 references
public class Account
{
    1 reference
    public BillingPlan BillingPlan { get; set; }
}

2 references
public class BillingPlan
{
    2 references
    public decimal Rate { get; set; }
}

// 사용 예시
Customer customer = new Customer();
decimal rate = customer.Account.BillingPlan.Rate;
```

안 좋은 예시

```
// 좋은 코드 예시
5 references
public class Customer
{
    1 reference
    private Account account;

    1 reference
    public decimal GetBillingRate()
    {
        return account.GetBillingRate();
    }
}

3 references
public class Account
{
    1 reference
    private BillingPlan billingPlan;

    1 reference
    public decimal GetBillingRate()
    {
        return billingPlan.Rate;
    }
}

// 사용 예시
Customer customer = new Customer();
decimal rate = customer.GetBillingRate();
```

좋은 예시

19. 메시지 체인 (Message Chain)

- **클래스도 프라이버시가 있다.**
- 클라이언트 코드에서 객체의 내부 구조에 대해서 지나치게 많이 아는 것은 좋은 것이 아님.
- 클래스끼리 강한 결합이 있는 것은 피하자.

Example

클린코드 시리즈

김하연 튜터
챌린지반 특강 7강

안 좋은 코드란? 4탄

20. 서로 다른 인터페이스의 대안 클래스들

- 동일한 구조의 클래스는 왠만하면 하나의 인터페이스 아래 통합하여 유연한 코드를 만들어나가자.
- 슈퍼 클래스 추출하기

Example 1

Example 2

예제 코드 1

```
// 안 좋은 코드 예시 1
1 reference
public class CsvReader
{
    0 references
    public string ReadCsvFile() { /* CSV 파일 읽기 로직 */ }
}

1 reference
public class XmlReader
{
    0 references
    public string ReadXmlFile() { /* XML 파일 읽기 로직 */ }
}
```

안 좋은 예시

```
// 좋은 코드 예시 1
2 references
public interface IFileReader
{
    2 references
    string ReadFile();
}

1 reference
public class CsvReader : IFileReader
{
    1 reference
    public string ReadFile() { /* CSV 파일 읽기 로직 */ }
}

1 reference
public class XmlReader : IFileReader
{
    1 reference
    public string ReadFile() { /* XML 파일 읽기 로직 */ }
}
```

좋은 예시

20. 서로 다른 인터페이스의 대안 클래스들

- 동일한 구조의 클래스는 왠만하면 하나의 인터페이스 아래 통합하여 유연한 코드를 만들어나가자.
- 슈퍼 클래스 추출하기

Example 1

Example 2

예제 코드 2

```
// 안 좋은 코드 예시 2
1 reference
public class PdfExporter
{
    0 references
    public void Export(string content) { /* PDF 내보내기 로직 */ }
    // 기타 공통 메소드들...
}

1 reference
public class WordExporter
{
    0 references
    public void Export(string content) { /* Word 내보내기 로직 */ }
    // 기타 공통 메소드들...
}
```

안 좋은 예시

```
// 좋은 코드 예시 2
2 references
public abstract class FileExporter
{
    2 references
    public abstract void Export(string content);
    // 기타 공통 메소드들...
}

1 reference
public class PdfExporter : FileExporter
{
    1 reference
    public override void Export(string content) { /* PDF 내보내기 로직 */ }
}

1 reference
public class WordExporter : FileExporter
{
    1 reference
    public override void Export(string content) { /* Word 내보내기 로직 */ }
}
```

좋은 예시

20. 서로 다른 인터페이스의 대안 클래스들

- 동일한 구조의 클래스는 왠만하면 하나의 인터페이스 아래 통합하여 유연한 코드를 만들어나가자.
- 슈퍼 클래스 추출하기

Example 1

Example 2

클린코드 시리즈

김하연 튜터
챌린지반 특강 7강

안 좋은 코드란? 4탄

클린코드 20가지 원칙

- 기이한 이름
- 중복 코드
- 긴 함수
- 전역 변수의 남용
- 주석의 남용
- 가변 데이터
- 뒤엉킨 변경
- 기본형 집착
- 반복되는 Switch문
- 성의 없는 요소
- 기능 편애
- 데이터 뭉치
- 반복문
- 임시 필드
- 상속 포기하기
- 추측성 일반화
- 긴 매개변수 목록
- 거대한 클래스
- 메시지 체인
- 서로 다른 인터페이스의 대안 클래스들

클린코드 시리즈

김하연 튜터
챌린지반 특강 7강

안 좋은 코드란? 4탄