1회차 과제 제출[디자인패턴, 프로그래밍 패러다임]

▼ 목차

디자인 패턴

1. 싱글톤 패턴(singleton pattern)

하나의 클래스에 오직 하나의 인스턴스만 가지는 패턴이다.

하나의 인스턴스를 여러 모듈이 공유해서 사용한다.

하나의 클래스를 기반으로 하나의 인스턴스를 만들어서 이를 기반으로 로직을 구현하는데 쓰인다.

장점 : 인스턴스 생성 비용이 적음.

단점 : 단위별 테스트가 불가함, 의존성이 높음⇒**의존성 주입**

ex) 데이터베이스 연결 모듈

- 의존성 주입이란?

모듈간의 의존성 정도를 낮추기 위한 방법.

의존성이란 종속성이라고도 하며, B가 A에 의존성이 있다는 것은 A의 변경사항에 대해 B에도 동일하게 변경된다는 것을 의미한다.

- 의존성 주입의 장점

1. 애플리케이션 의존성 방형이 일관
2. 애플리케이션 추론의 간편화
3. 모듈간의 관계 명확성 증가

- 의존성 주입 하는 방법

1. 생성자를 사용해서 주입(권장)

```
@Service
public class BugService {

  private Login login;

  public BugService(Login login){
    this.login = login;
```

```
  }

}
```

2. Field 변수를 통한 주입

```
@Service
public class BugService {

  @Autowired
  private BugRepository bugRepository;

}
```

3. setter를 통한 주입

```
@Service
public class BugService {

  private BugRepository bugRepository;

  @Autowired
  public void setBugRepository(BugRepository bugRepository) {
    this.bugRepository = bugRepository;
  }
}
```

2. 팩토리 패턴(factory pattern)

팩토리 패턴이란?

- 객체를 사용하는 코드에서 객체 생성 부분을 떼어내 추상화한 패턴

- 상위 클래스가 주요 뼈대를 결정하고 하위 클래스에서 객체 생성에 관한 구체적인 내용을 결정하는 패턴이다.
- 상위클래스와 하위클래스가 분리되어 있다.

비유 : 커피 기계⇒ 상위클래스

아메리카노 제조법, 라떼 제조법 ⇒ 하위클래스

팩토리 패턴의 장점

1. 결합력이 느슨하고 유연함.
2. 유지보수의 편리성

```
const num = new Object(42);
const str = new Objest('abc');
num.constructor.name;
str.constructor.name;
```

전달하는 타입에 따라서 Object의 타입이 달라지는 것을 볼 수 있다.

⇒ 하위 클래스의 모습에 따라서 상위클래스가 변동된다.

3. 전략 패턴(strategy pattern)

📎 https://victorydntmd.tistory.com/292

객체들이 할 수 있는 행위 각각에 대해 전략 클래스를 생성하고, 유사한 행위들을 캡슐화하는 인터페이스를 정의하여,

객체의 행위를 동적으로 바꾸고 싶은 경우 직접 행위를 수정하지 않고 전략을 바꿔주기만 함으로써 행위를 유연하게 확장하는 방법이다.

객체가 할 수 있는 행위들을 각각의 전략으로 만들어 놓고 동적으로 행위의 수정이 필요한 경우 전략을 바꾸는 것만으로 행위를 수정이 가능하도록 만든 패턴이다.

두 가지 이상의 전략로직을 만들고 필요에 따라 적합한 로직을 사용하는 방법

자바로 전략패턴 구현(교재내용)

js로 전략패턴 구현(교재내용)

4. 옵저버 패턴

옵저버 패턴(observer pattern)이란 객체의 상태 변화를 관찰하는 관찰자들, 즉 옵저버들의 목록을 객체에 등록하여 상태변화가 있을 때 마다 메서드를 통해 객체가 직접 목록의 각 옵저버에게 통지하도록 하는 디자인 패턴이다.

주로 분산 이벤트 핸들링 시스템을 구현하는 데 사용된다. 발행/구독 모델로 알려져 있기도 한다.

어떤 객체의 상태가 변할 때 그와 연관된 객체들에게 알림을 보내는 디자인패턴.

[예시]

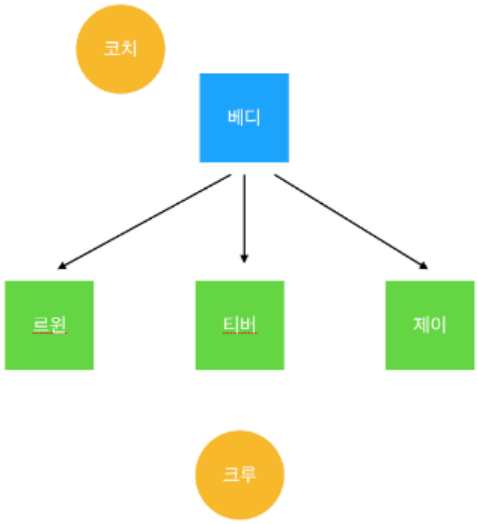
학생들은 코치가 하는 일들을 모두 알림 받아야 한다.

코치가 “밥을 먹는다”면 모든 크루들은 코치가 밥을 먹었다는 것을 알아야 한다.

베디라는 객체는 코치이다.(코치 인터페이스)

코치 인터페이스의 기능은 크루들을 등록한다, 크루들을 등록 해제한다, 크루들에게 행동을 알린다.

3가지 기능이다.



자바에서의 옵저버 패턴

js에서의 프록시 객체를 이용한 옵저버 패턴

js에서의 옵저버 패턴은 프록시 객체를 통해서 구현할 수도 있다.

프록시 객체란

프록시 객체는 어떠한 대상의 기본적인 동작(속성 접근, 할당, 순회, 열거, 함수호출 등)의 작업을 가로챌 수 있는 객체를 뜻하며, 자바스크립트에서 프록시 객체는 두 개의 매개변수를 가진다.

객체의 여러 기본 동작을 가로채서 특별한 다른 동작을 할 수 있게 한다.

객체는 js의 모든 자료형이 대상이 될 수 있다.

target : 프로시할 대상

handler : 프록시 객체의 target동작을 가로채서 정의할 동작들이 정해져 있는 함수

✅ 자바에서의 상속과 구현의 차이

상속 : 추가, 확장 가능

구현 : 재정의 가능

프록시 객체를 구현한 코드

```
const handler = {
  get : function(target, name){
    return name ==='name' ? `${target.a} ${target.b}` : target[name]
  }
}

const p = new Proxy({a:'kundol', b : 'is amunu zangin'}, handler)
console.log(p.name)    // kundol is amunu zangin
```

5. 프록시 패턴과 프록시 서버

프록시 패턴이란?

대상 객체에 접근하기 전 그 접근에 대한 흐름을 가로채 대상 객체 앞단의 인터페이스 역할을 하는 디자인 패턴이다.

하나의 로직이 작동하는 일련의 과정에서 한 단계만 변경하고 싶은 경우 프록시 패턴을 사용하여 해당 부분의 작동을 변경하여 사용함.

프록시 패턴의 사용성 : 보안, 데이터 검증, 캐싱, 로깅

js의 프록시 객체 자세한 설명

프록시 서버

프록시 서버로 쓰이는 대표적인 서버로는 Nginx와 CloudFlare가 있다.

Q. 프록시 서버를 둠으로써 얻을 수 있는 기술적 이점은? cdn, ddos 공격방어, https구축

cdn : 웹 사이트 콘텐츠를 지리적으로 사용자와 가까운 CDN 서버에 저장하여 컴퓨터에 훨씬 빨리 도달할 수 있도록 한다.

6. 이터레이터 패턴(iterator pattern)

이터레이터 패턴이란?

이터레이터 패턴은 이터레이터를 사용하여 컬렉션의 요소들에 접근하는 디자인 패턴이다.

자료형의 구조가 뭐던지 간에 이터레이터라는 하나의 인터페이스를 탈 수 있다.

ex)for a of b 이터레이터 프로토콜

7. 노출모듈 패턴(revealing module pattern)

노출모듈 패턴이란?

노출모듈 패턴이란 즉시실행 함수를 통해 private, public과 같은 접근제어자를 두는 패턴이다.

js의 경우에는 java와 달리 접근제어자가 존재하지 않기 때문에 노출 모듈을 사용하여 접근제어자를 둘 수 있다.

8. MVC 패턴(model view controller)

mvc패턴이란 model view controller로 이루어진 디자인 패턴이다.

9. MVP 패턴(model view presenter)

mvc패턴이란 model view controller로 이루어진 디자인 패턴이다.

8. MVVM 패턴(model view vie model)

mvc패턴이란 model view controller로 이루어진 디자인 패턴이다.

프로그래밍 패러다임(programming paradigm)

패러다임이란?

한 시대 사람들의 견해나 사고를 근본적으로 규정하고 있는 **인식의 체계**

프로그래밍 패러다임이란?

프로그래머에게 프로그래밍의 관점을 갖게 해주는 역할을 하는 **개발 방법론**

프로그래밍을 거시적인 관점에서 봤을 때 개발 방법론을 통해 전체적인 흐름을 읽을 수 있음

내가 추구하는 개발 방법론은?

프로그래밍 패러다임의 구분

선언형 ⇒ 함수형

명령형 ⇒ 객체지향

⇒ 절차지향

선언형 프로그래밍(declarative programming)

“프로그램은 함수로 이루어진 것이다.”

‘무엇을 풀어내는가’가 중요한 방법론.

함수형 프로그래밍(functional programming)

함수형 프로그래밍은 선언형 프로그래밍에 속한다.

순수함수들이 블럭을 쌓아 로직을 구현하고 고차함수를 통해 재사용성을 높인 프로그래밍 패러다임이다.

```
const mungChi = [1,2,3,4,5,11,12];
  .reduce((max,num) ⇒ num >max ? num : max,0);
  console.log(ret);    //12
```

명령형 프로그래밍

객체지향프로그래밍

객체지향프로그래밍(OOP, object- Oriented Programming)은 객체의 집합으로 프로그램의 상호작용을 표현하며 데이터를 객체로 취금하여 객체 내부에 선언된 메서드를 활용하는 방식

객체지향프로그래밍 특징

- 추상화
- 캡슐화
- 상속성
- 다형성

오버로딩(overloading)

같은 이름을 가진 메서드를 여러개 두는 것.

매개변수를 다양하게 변환하여 둘 수 있으며 컴파일 중에 발생하는 정적 다형성.

오버라이딩(overriding)

오버라이딩은 일반적으로 메서드오버라이딩을 가르킨다.

상위 클래스로부터 상속받은 메서드를 하위클래스가 재정의하는 것을 의미한다.

런타임 중에 발생하는 동적 다향성.

절차형프로그래밍

로직이 수행되어야 할 연속적인 계산 과정으로 이루어져 있습니다.

하나의 일련의 과정으로 코드를 구현하기 때문에 가독성이 좋고 실행이 빠르다.