1. 스프링의 핵심
   1. 좋은 객체 지향 애플리케이션을 개발할 수 있게 도와주는 프레임워크!!
2. 객체 지향이란?
   1. 절차 지향 프로그래밍 ->
      1. 함수와 제어문을 통한, 코드의 논리적 흐름, 즉 프로그램을 명령어의 목록으로 보는 시각
   2. 객체 지향 프로그래밍 ->
      1. 프로그래밍을 여러 개의 독립된 단위인, ‘객체’의 모임으로 보고, 이들간의 메시지 전달을 통해, 문제를 해결
      2. 객체지향 프로그래밍은, 프로그램을 유연하고 변경하기 용이하게 만들기 때문에, 대규모 소프트웨어 개발에 많이 사용됨
3. 객체 지향의 특징
   1. ***다형성***
      1. 세상을 역할과, 구현으로 구분
         1. 클라이언트는, 대상의 내부 상세 구현을 모르더라도, 역할만 알면, 상호작용이 가능하기 때문에, 구현이 바뀌더라도, 새로운 구현이 역할을 충실히 따른다면, 클라이언트를 변경하지 않아도 된다.
         2. 인터페이스가 동일하다면, 구현체만 변경하면 되기 때문에, 상세 구현이 변경되어도 클라이언트 코드의 변경이 일어나지 않는다.
         3. 예시: 운전자 – 자동차 (운전자는 자동차라는 역할에 의존한다)
         4. 그림-1 참고
   2. SOLID
      1. SRP
         1. 단일 책임 원칙
            1. 하나의 클래스는 하나의 책임만 가져야 한다.
            2. 코드의 변경의 이유는 하나여야 한다.
      2. OCP
         1. 개방-폐쇄 원칙
            1. 확장에는 열려있고, 변경에는 닫혀있어야 한다.
            2. ***객체를 생성하고, 연관관계를 맺어주고, 조립해주는 설정자가 필요(스프링)***
            3. 예시

MySQL을 사용하던 중, Oracle을 사용하도록 변경하더라도, 클라이언트 코드의 변경이 일어나지 않는다.

* + 1. LSP
       1. 리스코프 치환 원칙
          1. 자식 타입의 객체는 부모 타입의 각체를 대체할 수 있어야 한다.
          2. 예시

액셀을 눌렀을 때는 앞으로 가야 하는데, 이걸 뒤로 가도록 구현한다면, LSP 원칙을 깨는 것

* + 1. ISP
       1. 인터페이스 분리 원칙
          1. 특정 클라이언트를 위한 인터페이스 여러 개가, 범용 인터페이스 하나보다 낫다.
    2. DIP
       1. 의존관계 역전 원칙
          1. 추상화에 의존해야 하며, 구체화에 의존하면 안된다.
          2. 클라이언트 코드는 역할에 의존해야하며, 이는 곧 인터페이스에 의존하다록 하고, 실제 구현 코드에 의존해서는 안된다.(구현 코드에 대하여 알면 안된다)

1. 스프링
   1. DI : 의존성 주입
   2. DI 컨테이너 제공(자바의 객체들을 생성하고, 이 내부에서 의존관계를 연결)
   3. 위의 객체지향 원칙을 지키기 위한 중복 코드를 프레임워크한 것이 스프링!
2. 실습
   1. 강조 포인트
      1. DI 컨테이너 없이 개발해보기(DIP를 의존할 수밖에 없음)
      2. DI 컨테이너가 추가된 이후 영역 분리 확실히 하기(실행 영역 vs 구성 영역)
         1. 구현체가 변경된 경우에도, 클라이언트 코드에 변경이 없음을 보여주기
         2. 그림-2
      3. IoC
         1. 프로그램은, 인터페이스에만 의존하고, 각 인터페이스에 대한 주입은 컨테이너가 수행한다.
         2. 이때, 프로그렘의 제어 흐름을 직접 제어하는 것이 아니라, 외부에서 관리하는 것(프레임워크가 수행)을 제어의 역전이라고 한다.
      4. DI
         1. 의존 객체를 외부에서 주입해준다.
         2. 정적인 클래스간의 의존관계는, 실행 시점에 결정되는 동적인 객체의 의존관계와는 다르게 봐야 한다.
         3. DI를 사용하면, 정적인 클래스 의존관계는 변경하지 않고, 동적인 객체 인스턴스 간의 의존관계를 쉽게 변경할 수 있다.
      5. IoC 컨테이너
         1. AppConfig 처럼, 의존 관계를 연결해 주는 곳.
         2. DI 컨테이너라고 한다.

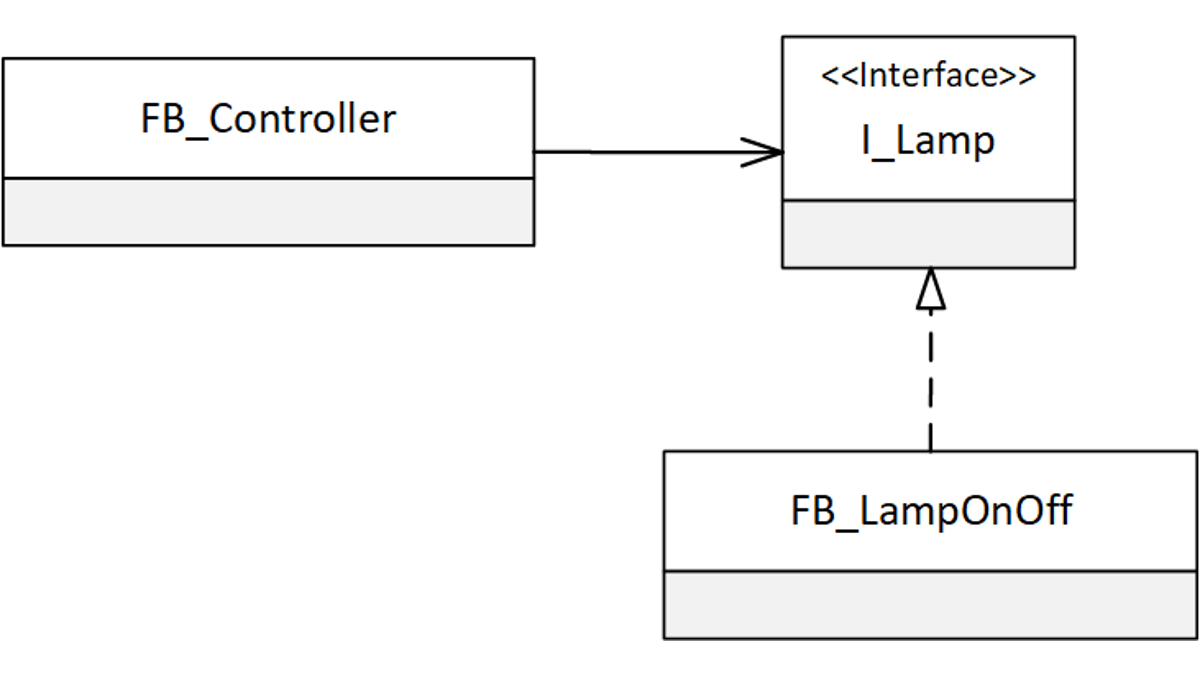


그림 1

A diagram of a company

Description automatically generated

그림 2