상속

객체지향 프로그래밍의 이점

- 모듈화
  - 큰 규모의 프로그램을 `객체`라는 작은 단위로 구분하여 관리할 수 있다.
    - 유지보수가 용이하다
      - 프로그램의 수정사항 있는 경우 필요한 객체만 수정하면 된다
    - 분업이 용이하다
      - 개발자의 역할을 객체 단위로 나눌 수 있다
- 재사용성
  - 한 번 정의한 `class` 로부터 다양한 속성을 가진 객체를 만들어낼 수 있다.
  - `상속` 을 통해 기존에 정의한 `class` 로부터 새로운 `객체`를 정의할 수 있다.

UML 다이어그램

- 객체의 정의와 객체간의 관계를 그림으로 나타낸 것

상속

- 기존 정의된 객체의 `상태` 와 `행위` 를 물려받은 확장된 객체를 정의하는 것
- `extends` 키워드를 사용한다
- 물려준 객체 : 부모 , 물려 받은 객체 : 자식

```
class Human {
        int age;
        String name;

        void work() {
                System.out.println("할 일을 합니다.");
```

```
        }
}

class Student extends Human{
        String major;

        @Override
        void work() {
                System.out.println("열심히 공부를 합니다.");
        }
}

class Developer extends Human {
        String companyName;

        @Override
        void work() {
                System.out.println("열심히 개발을 합니다.");
        }
}
```

오버라이딩

- 부모의 메소드를 자식이 재정의 하는 것
- `@Override` 어노테이션은 해당 메서드가 부모로 부터 물려받은 것임을 명시해준다

해시(Hash)

- 임의의 길이를 갖는 임의의 데이터를 고정된 길이의 데이터로 매핑하는 단방향 함수
- 다차원의 데이터를 하나의 값으로 나타낸것
- 복잡한 데이터를 효율적으로 비교하기 위해 고안되었음

다차원 데이터는 Hash Function을 거쳐 하나의 값으로 나타내며 이를 Hash Code라고 한다.