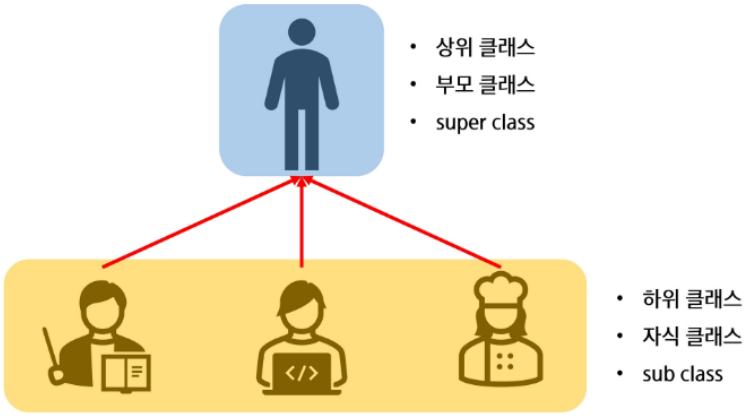
Java (상속 & 다형성)

날짜	@2024년 7월 15일
상태	진행 중

상속

어떤 클래스의 특성을 그대로 갖는 새로운 클래스를 정의한 것




```
public class Person {
    String name;
    int age;

    public void eat() {
        System.out.prinln("음식을 먹는다.");
    }
}
```

상속 X

```
public class Student {
    String name;
    int age;
    String major;
```

```
    public void eat() {
        System.out.println("음식을 먹는다.");
    }

    public void study() {
        System.out.println("공부를 한다.");
    }
}
```

상속 적용

```
public class Student extends Person {
    String major;

    public void study() {
        System.out.println("공부를 한다.");
    }
}
```

상속 (Inheritance)

1. **확장성**, **재사용성**
    - 부모의 생성자와 초기화 블록은 상속 X
2. 클래스 선언 시 `extends` 키워드 명시
    - 자바는 다중 상속 허용 X (단일 상속만 지원)
3. 관계
    - 부모 (상위, `Super`) 클래스
    - 자식 (하위, `Sub`) 클래스
4. 자식 클래스는 부모 클래스의 **멤버 변수**, **메소드**를 자신의 것처럼 사용 가능
    - 접근 제한자에 따라 사용 여부가 달라짐
5. `Object` 클래스는 모든 클래스의 **조상 클래스**
    - 별도의 extends 선언이 없는 클래스는 `extends Object` 생략된 상태
6. `super` 키워드

```
public class Person {
    String name;
    int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void eat() {
        System.out.println("음식을 먹는다.");
    }
}
```

- `super`를 통해 조상 클래스의 **생성자**, **메서드** 호출

```
public class Student extends Person {
    String major;

    public Student(String name, int age, String major) {
        super(name, age);
        this.major = major;
    }

    public void study() {
          super.eat();
          System.out.println("공부를 한다.");
      }
}
```

7. 오버라이딩 (재정의, `overriding`)
    - 상위 클래스에 선언된 메서드를 자식 클래스에서 **재정의**하는 것
    - 메서드의 이름, 반환형, 매개변수(타입, 개수, 순서)가 모두 동일해야 함
    - 하위 클래스의 접근 제어자 범위가 상위 클래스보다 크거나 같아야 함
    - 조상보다 더 큰 예외를 던질 수 없음

- 메서드 오버로딩(`overloading`)과 다름
  - 오버로딩 (다중 정의): 이름이 같고 매개변수가 다른 메서드를 여러 개 정의하는 것

```
public class Student extends Person {
    String major;

    public Student(String name, int age, String major) {
        super(name, age);
        this.major = major;
    }

    public void study() {
        super.eat();
        System.out.println("공부를 한다.");
    }

    public void eat() {
        System.out.println("지식을 먹는다.");
    }
}
```