상속

상속

- 부모 클래스의 **멤버**를 자식 클래스에게 물려줄 수 있다.
- 부모클래스를 상위 클래스(슈퍼 클래스)라고 부르고 자식 클래스를 하위 클래스(서브 클래스) 또는 파생 클래스 라고 부른다.
- 상속으로 **중복되는 코드**를 줄일 수 있다.
- 부모 클래스를 수정하면 **모든 자식 클래스들도 수정**되는 효과!
- 모든 클래스는 기본적으로 **Object 클래스를 상속**받고있다!

코드로 알아보는 상속

```java
class 할아버지 {
    String house = "이층집";
    public void singWell() { System.out.println("노래를 잘 한다!"); }
}

class 아버지 extends 할아버지 {
    String car = "자동차";
    public void drawWell() { System.out.println("그림을 잘 그린다!"); }
}

class 나 extends 아버지 {
    String pc = "컴퓨터";
    public void programmingWell() { System.out.println("프로그래밍을 잘 한다!"); }
}

public class Main {

    public static void main(String[] args) {
        나 나 = new 나();
        System.out.println(나.house);
        System.out.println(나.car);
        System.out.println(나.pc);
        나.singWell();
        나.drawWell();
        나.programmingWell();
```

```
Console
<terminated> Main (1) [Java Application] C:₩Program Files₩Java₩jre1.8.0_301₩bin₩javaw.exe (2021. 12. 20. 오전 10:05:01)
이층집
자동차
컴퓨터
노래를 잘 한다!
그림을 잘 그린다!
프로그래밍을 잘 한다!
```

자바에서 상속의 특징

- extends 키워드 뒤에 부모 클래스를 기술 `class Child extends Parent`
- 여러 개의 부모 클래스를 상속 받을 수 없다.
- 부모 클래스에서 private 접근 제한을 갖는 멤버변수와 메소드는 상속 대상에서 제외된다.
- 부모클래스와 자식클래스가 다른 패키지에 존재한다면 default 접근제한자를 갖는 멤버변수와 메소드도 상속 대상에서 제외

부모 생성자 호출 (super)

- 부모 객체가 먼저 생성된 후 자식 객체가 생성됨.
- 자식 생성자의 맨첫줄에 super()가 생략 되어있음.
- super(매개값, …) 으로 명시적으로 호출 가능하며, 일치하는 부모 생성자가 없을 경우 컴파일 에러
- 부모 클래스에 기본 생성자가 없을 경우 반드시 자식 생성자의 첫줄에 부모 생성자를 호출 해야한다!

```
class People {
    public String name;
    public String ssn;

    public People(String name, String ssn) {
        this.name = name;
        this.ssn = ssn;
    }
}

class Student extends People {
    public int studentNo;
    public Student(String name, String ssn, int studentNo) {
        super(name, ssn);
        this.studentNo = studentNo;
    }
}
```

메서드 재정의 (Overriding)

- 부모-자식 클래스가 100% 맞게 설계되었다면 가장 이상적이지만, **일부 메서드는 자식 클래스에서 적합하지 않을 수 있다!**
- **자식클래스에서 다시 재정의** 하는것이 오버라이딩(Overriding)

❖ **메소드 재정의 방법**

- 부모의 메서드와 동일한 시그니처(리턴타입, 메소드명, 매개변수)
- 접근 제한을 **더 강하게 정의 X** (public -> protected, private 등)
- 새로운 예외(Exception)를 throws할 수 없다.

메서드 재정의 (Overriding)

- @Override **어노테이션**이 생성됨.
  (생략 가능하지만, 보통 그대로 명시!)

- super.**부모메서드()**로
  부모 메서드 호출 가능!

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

    public void jump() {
        System.out.println("일반인의 점프!");
    }

    public void eat() {
        System.out.println("먹기!");
    }
}

class SpiderMan extends Person {
    boolean isSpider;

    @Override
    public void jump() {
        if(isSpider) {
            System.out.println("스파이더맨의 강력한 점프!");
        } else {
            super.jump();
        }
    }

}
```

final

- 해당 선언이 최종 상태이고 수정될 수 없음을 뜻하는 final
- 클래스, 멤버변수, 메소드를 선언할 때 사용가능
- final클래스: 상속할 수 없는 클래스(예: String)
- final메소드: 자식 클래스에서 재정의 할 수 없는 메소드
- final변수: 한번 초기화되면 변경할 수 없는 변수(즉, 상수)

protected 접근제한자

- default 접근제한자와 유사하지만 protected는 다른패키지 이더라도 **상속 관계일 경우에는 접근허용**

- new 연산자를 통해 생성자를 직접 호출 할 수는 없지만 **super()는 가능**

```
class A {
    protected String member1;
    protected A() {}
    protected void method() {}
}

class B extends A {
    public B() {
        super();
        this.member1 = "value";
        this.me
    }
}
```

member1 : String - A
method() : void - A