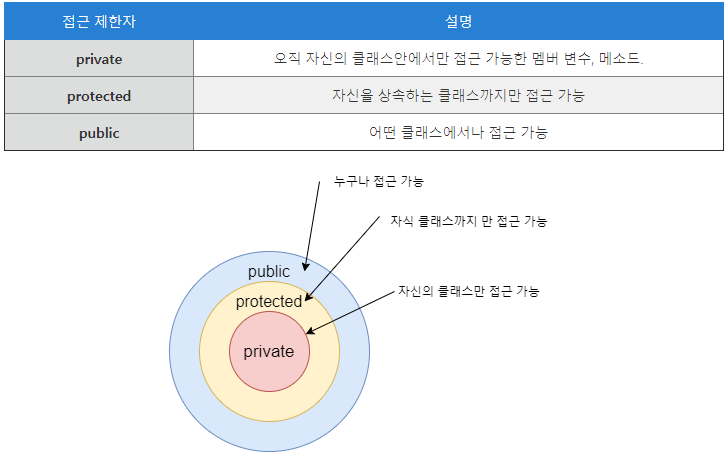
**Class Inheritance**

1. **클래스 상속**- 객체지향 프로그래밍에서는 부모 클래스의 멤버 변수와 메소드를 자식 클래스가 재사용  
   - 멤버 변수나 메소드에 접근 제한자를 더하여 아무나 접근할 수 없도록 할 수 있다.  
   - ‘ : ‘ 콜론을 사용하여 상속을 받을 수 있다.
2. **접근제어 지시자**
3. **캡슐화(Encapsulation)**- 멤버 변수를 직접 변경할 수 없도록 캡슐처럼 껍데기를 둘러싸는 과정  
   - 예시코드  
   - B는 A를 상속받고, a, b는 직접 설정할 수 없게 private로 접근을 제한  
   - 메인 함수에서 b는 setAB를 호출해 a의 값을 100, b의 값을 200으로 바꾸려고 한다.  
   - 하지만 A에서 a의 최대값은 50, b의 최대값은 100으로 제한  
   - B가 a, b를 변경하려면 setA와 setB를 호출하여 인자를 전달.  
   - 이렇게 멤버의 값을 함부로 변경할 수 없도록 멤버함수로 껍질을 입히는 작업을 캡슐화텍스트이(가) 표시된 사진

   자동 생성된 설명
4. **다형성(Polymorphism)**- 부모클래스를 통해 만들어진 객체가 자신을 상속받는 여러 클래스 객체로 모양을 띄는 것  
   - Virtual 키워드  
   : 자신을 상속받는 자식 클래스의 객체가 자신의 메소드를 사용하는데 오버라이딩 했다면  
   자식 클래스의 객체 메소드를 호출하라고 지정하는 방법은 메소드 앞에 virtual 키워드 사용  
   자신의 메소드는 가상으로 만들어져 있으니 자식의 메소드를 호출하라는 의미  
   - 메소드의 virtual이 지정되면 이후 자식은 자동으로 virtual 키워드가 적용된다.  
   - 하지만 명시적으로 virtual을 지정해주는 것이 관례 -> 코드의 가독성과 이해  
   - 다형성이 필요한 경우  
   : 함수에서 해당 클래스를 상속하는 모든 객체를 받는 경우