Chapter 07

상속

07-1. 상속

혼자 공부하는 자바 (신용권 저)

한빛미디어 Hanbit Media, Inc.

시작하기 전에

[핵심 키워드] : 상속, 메소드 재정의, final 클래스, final 메소드

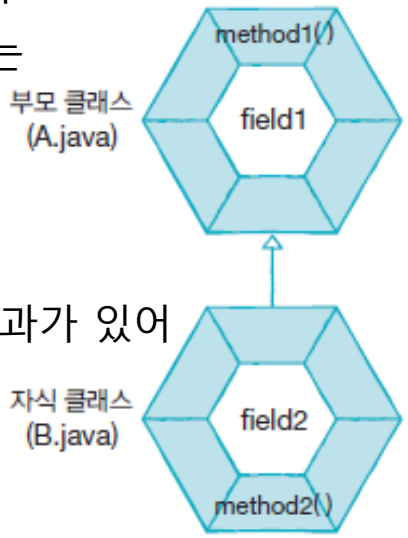
[핵심 포인트]

객체 지향 프로그램에서 부모 클래스의 멤버를 자식 클래스에게 물려줄 수 있다.

❖ 상속

- 이미 개발된 클래스를 재사용하여 새로운 클래스를 만들기에 중복되는 코드를 줄임

- 부모 클래스의 한번의 수정으로 모든 자식 클래스까지 수정되는 효과가 있어 유지보수 시간이 줄어듬




```
public class A {
  int field1;
  void method1() { … }
}
```

```
A를 상속
public class B extends A {
  String field2;
  void method2() { … }
}
```

클래스 상속

❖ 클래스 상속

- 자식 클래스 선언 시 부모 클래스 선택
- extends 뒤에 부모 클래스 기술

```
class 자식클래스 extends 부모클래스 {
  //필드
  //생성자
  //메소드
}
```

```
class SportsCar extends Car {
}
```

- 여러 개의 부모 클래스 상속할 수 없음
- 부모 클래스에서 private 접근 제한 갖는 필드와 메소드는 상속 대상에서 제외
- 부모와 자식 클래스가 다른 패키지에 존재할 경우 default 접근 제한된 필드와 메소드 역시 제외

예제 1 (상속 접근제한자)

CellPhone.java

```java
package sec01.exam01;

public class CellPhone {
    //필드
    String model;
    String color;

    //생성자

    //메소드
    void powerOn() { System.out.println("전원을 켭니다."); }
    void powerOff() { System.out.println("전원을 끕니다."); }
    void bell() { System.out.println("벨이 울립니다."); }
    void sendVoice(String message) { System.out.println("자기: " + message); }
    void receiveVoice(String message) { System.out.println("상대방: " + message); }
    void hangUp() { System.out.println("전화를 끊습니다."); }
}
```

예제 1 (상속 접근제한자)

DmbCellPhone.java

```java
package sec01.exam01;

public class DmbCellPhone extends CellPhone {
    //필드
    int channel;

    //생성자
    DmbCellPhone(String model, String color, int channel) {
        this.model = model;
        this.color = color;
        this.channel = channel;
    }

    //메소드
    void turnOnDmb() {
        System.out.println("채널 " + channel + "번 DMB 방송 수신을 시작합니다.");
    }
    void changeChannelDmb(int channel) {
        this.channel = channel;
        System.out.println("채널 " + channel + "번으로 바꿉니다.");
    }
    void turnOffDmb() {
        System.out.println("DMB 방송 수신을 멈춥니다.");
    }
}
```

예제 1 (상속 접근제한자)

DmbCellPhoneExample.java

```java
package sec01.exam01;

public class DmbCellPhoneExample {
    public static void main(String[] args) {
        //DmbCellPhone 객체 생성
        DmbCellPhone dmbCellPhone = new DmbCellPhone("자바폰", "검정", 10);

        //CellPhone으로부터 상속 받은 필드
        System.out.println("모델: " + dmbCellPhone.model);
        System.out.println("색상: " + dmbCellPhone.color);

        //DmbCellPhone의 필드
        System.out.println("채널: " + dmbCellPhone.channel);

        //CellPhone으로부터 상속 받은 메소드 호출
        dmbCellPhone.powerOn();
        dmbCellPhone.bell();
        dmbCellPhone.sendVoice("여보세요");
        dmbCellPhone.receiveVoice("안녕하세요! 저는 홍길동인데요");
        dmbCellPhone.sendVoice("아~ 예 반갑습니다.");
        dmbCellPhone.hangUp();

        //DmbCellPhone의 메소드 호출
        dmbCellPhone.turnOnDmb();
        dmbCellPhone.changeChannelDmb(12);
        dmbCellPhone.turnOffDmb();
    }
}
```

부모 생성자 호출

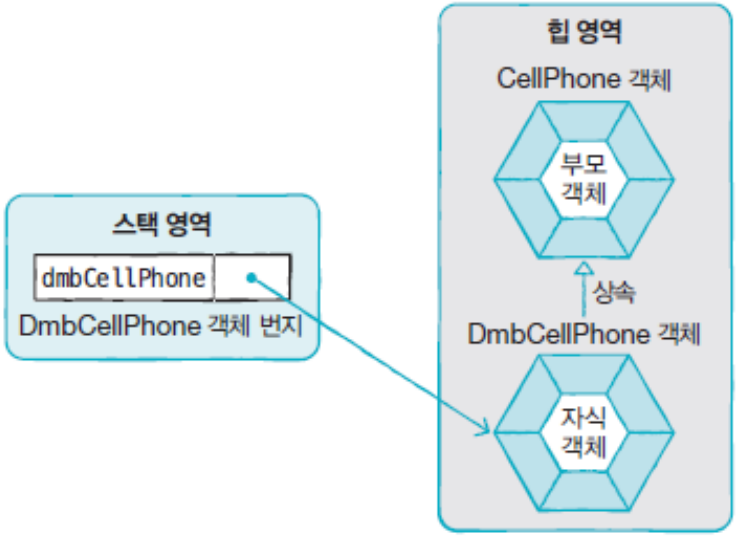
❖ 자식 객체 생성할 때 부모 객체가 먼저 생성되고 그 다음 자식 객체가 생성됨

```
DmbCellPhone dmbCellPhone = new DmbCellPhone();
```

- 자식 생성자의 맨 첫 줄에서 부모 생성자가 호출됨

```
public DmbCellPhone() {
  super();
}
```

```
public CellPhone() {
}
```



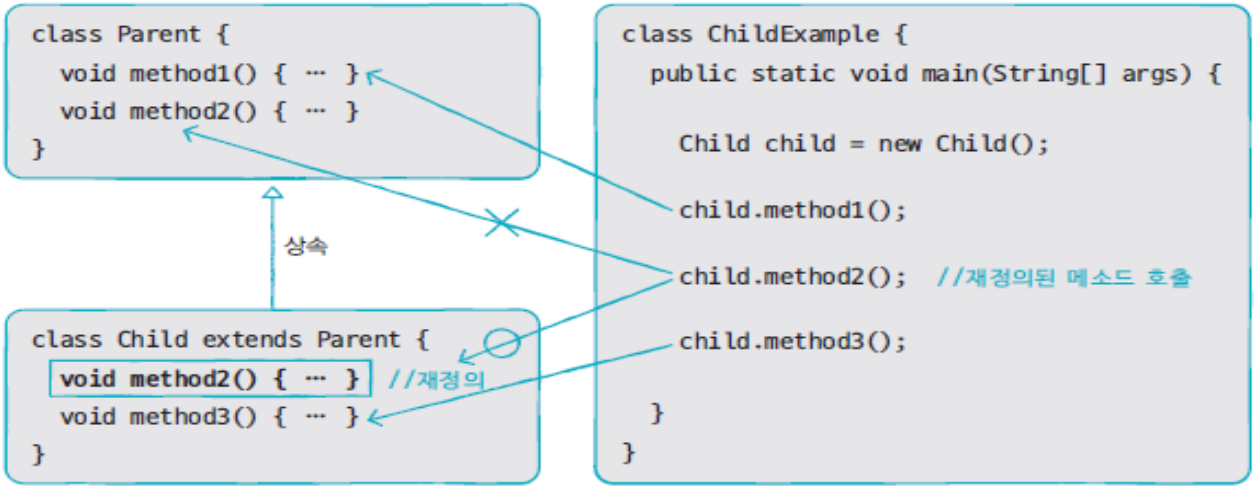
- 명시적으로 부모 생성자 호출하려는 경우

```
자식클래스( 매개변수선언, … ) {
  super( 매개값, … );
  …
}
```

메소드 재정의

❖ 메소드 재정의 (오버라이딩 / Overriding)

- 부모 클래스의 메소드가 자식 클래스에서 사용하기에 부적합할 경우 자식 클래스에서 수정하여 사용
- 메소드 재정의 방법
  - 부모 메소드와 동일한 시그니처 가져야 함
  - 접근 제한 더 강하게 재정의할 수 없음
  - 새로운 예외를 throws 할 수 없음
- 메소드가 재정의될 경우 부모 객체 메소드가 숨겨지며,
  자식 객체에서 메소드 호출하면 재정의된 자식 메소드가 호출됨



예제 2 (오버라이딩 / Overriding)

Calculator.java

```java
package sec01.exam03;

public class Calculator {
    double areaCircle(double r) {
        System.out.println("Calculator 객체의 areaCircle() 실행");
        return 3.14159 * r * r;
    }
}
```

Computer.java

```java
package sec01.exam03;

public class Computer extends Calculator {
    @Override
    double areaCircle(double r) {
        System.out.println("Computer 객체의 areaCircle() 실행");
        return Math.PI * r * r;
    }
}
```

ComputerExample.java

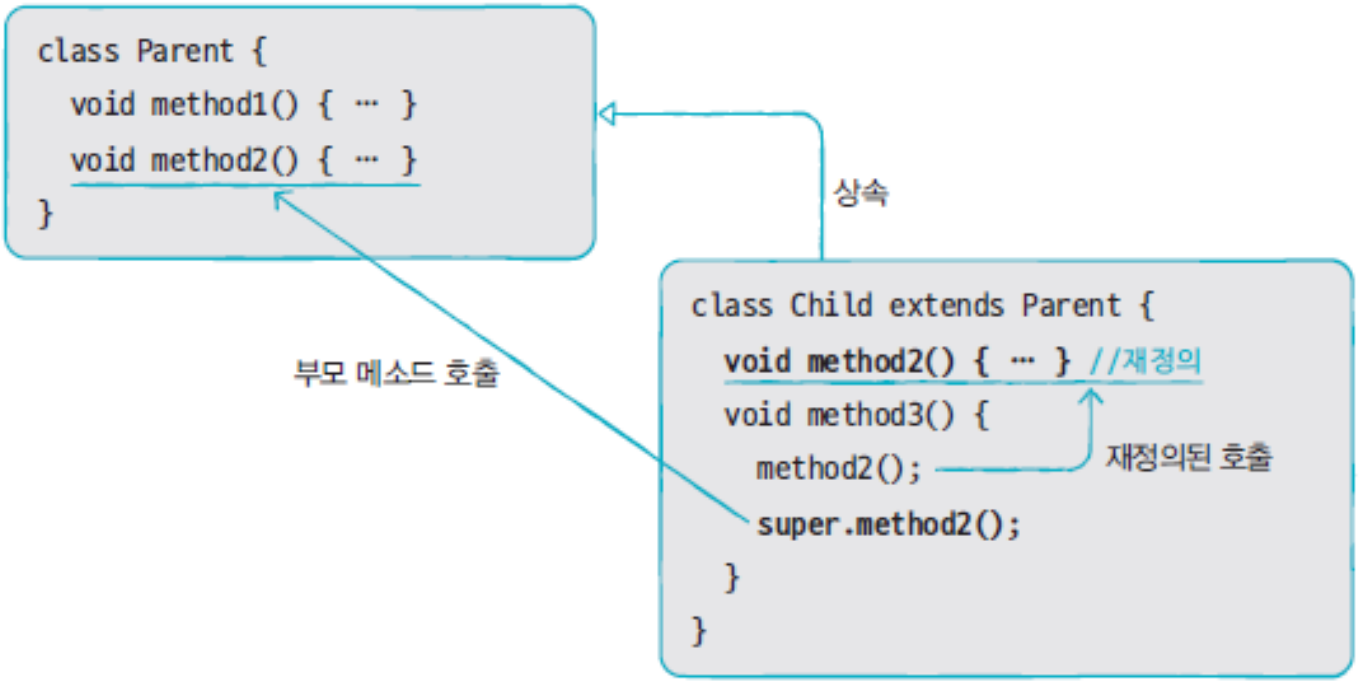
```java
package sec01.exam03;

public class ComputerExample {
    public static void main(String[] args) {
        int r = 10;
        Calculator calculator = new Calculator();
        System.out.println("원면적 : " + calculator.areaCircle(r));
        System.out.println();
        Computer computer = new Computer();
        System.out.println("원면적 : " + computer.areaCircle(r));
    }
}
```

메소드 재정의

- 부모 메소드 호출
  - 자식 클래스 내부에서 재정의된 부모 클래스 메소드 호출해야 하는 경우
  - 명시적으로 super 키워드 붙여 부모 메소드 호출

```
super.부모메소드();
```



final 클래스와 final 메소드

❖ final 키워드

- 해당 선언이 최종 상태이며 수정될 수 없음을 의미
- 클래스 및 메소드 선언 시 final 키워드를 사용하면 상속과 관련됨

❖ 상속할 수 없는 final 클래스

- 부모 클래스가 될 수 없어 자식 클래스 만들 수 없음을 의미

```
public final class 클래스 { … }
```

```
public final class String { … }
```

```
public class NewString extends String { … }
```

❖ 재정의할 수 없는 final 메소드

- 부모 클래스에 선언된 final 메소드는 자식 클래스에서 재정의 할 수 없음

```
public final 리턴타입 메소드( [매개변수, …] ) { … }
```

키워드로 끝내는 핵심 포인트

- 상속: 부모 클래스의 필드와 메소드를 자식 클래스에서 사용할 수 있도록 한다.
- 메소드 재정의 : 부모 메소드를 자식 클래스에서 다시 정의하는 것을 의미한다.
- final 클래스 : final 클래스는 부모 클래스로 사용할 수 없다.
- final 메소드 : 자식 클래스에서 재정의할 수 없는 메소드이다.


혼자 공부하는 자바
혼자 공부하는 자바
1:1 과외처럼 공부하고 싶을 때
작심삼일, 하다가 포기하게 될 때
어제 배운 내용이 생각나지 않을 때
신용권 지음
JAVA 8&11 지원
245개의 손코딩 예제
무료 동영상 강의
별책 용어 노트
한빛미디어

Thank You!