

7장. 상속과 다형성



객체지향 언어(OOP)

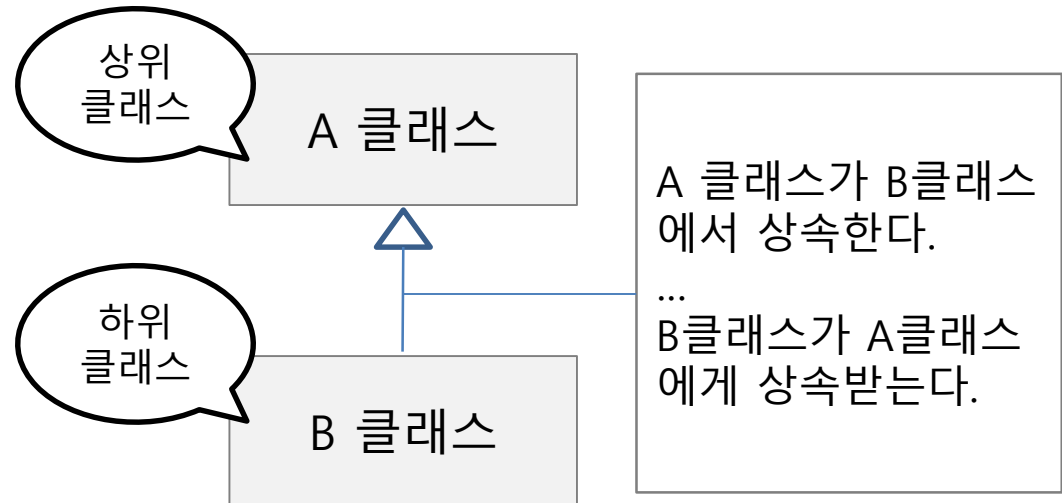


상속(Inheritance)

■ 상속이란?

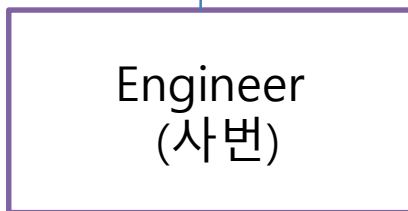
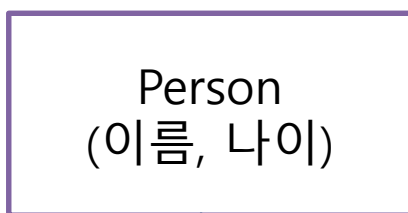
- 클래스를 정의할때 이미 구현된 클래스를 상속(inheritance) 받아서 속성이 나 기능(메서드)이 확장되는 클래스를 구현할 수 있다.
- 상속하는 클래스 : 상위 클래스, parent class
- 상속받는 클래스 : 하위 클래스, child class
- 클래스 상속 문법

```
class B extends A{  
    ....  
}
```



상속(Inheritance)

■ 멤버 속성 상속



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

```
class Engineer extends Person{  
    int companyId;  
}
```

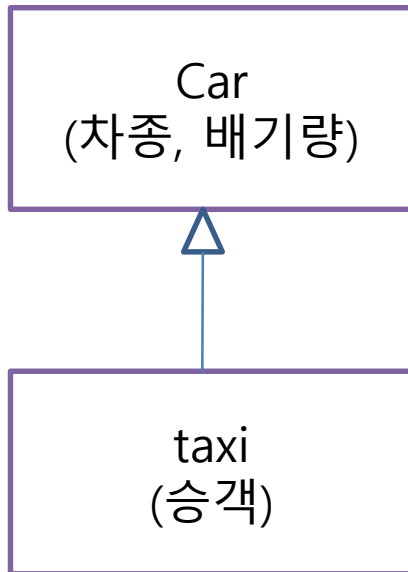
Person으로부터 상속받은 멤버변수

```
Engineer e1 = new Engineer();  
e1.name = "봉구"; //부모 멤버에 접근  
e1.age = 27;  
e1.companyId = 256;
```



super() – 부모 멤버 상속

▪ 매개변수 있는 생성자 상속 - super()



People로부터 상속받은 멤버변수

```
class Car{
    String brand;
    int cc;

    Car(String brand, int cc){
        this.brand = brand;
        this.cc = cc;
    }
}
```

```
class Taxi extends Car{
    int passenger;

    Taxi(String brand, int cc, int passenger){
        super(brand, cc); //부모 멤버 상속
        this.passenger = passenger;
    }
}
```



super() – 부모 멤버 상속

- 매개변수 있는 생성자 상속 - super() 사용

```
class Car{
    String brand;
    int cc;

    Car(String brand, int cc){
        this.brand = brand;
        this.cc = cc;
    }

    void showCarInfo() {
        System.out.println("모델: " + brand + ", 배기량: " + cc);
    }
}
```



super() – 부모 멤버 상속

- 매개변수 있는 생성자 상속 - super() 사용

```
class Taxi extends Car{
    int passenger;

    Taxi(String brand, int cc, int passenger){
        super(brand, cc);
        this.passenger = passenger;
    }

    @Override //메서드 재정의
    void showCarInfo() {
        System.out.println("모델: " + brand + ", 배기량: " + cc
            + ", 승객수: " + passenger);
    }
}
```



super() – 부모 멤버 상속

▪ 매개변수 있는 생성자 상속 - super() 사용

```
public class CarTest {  
    public static void main(String[] args) {  
        //Car 객체 생성  
        Car car = new Car("스포츠카", 2000);  
        car.showCarInfo();  
  
        //Taxi 객체 생성  
        Taxi 카카오T = new Taxi("카카오T", 2500, 2);  
        카카오T.showCarInfo();  
    }  
}
```

모델: 스포츠카, 배기량: 2000

모델: 카카오T, 배기량: 2500, 승객수: 2



메서드 재정의(Overriding)

■ 메소드 상속 및 재정의(Method Overriding)

- 상속된 메서드의 내용이 자식 클래스에 맞지 않을 경우, 자식 클래스에서 동일한 메서드를 재정의 하는 것을 말한다.
- 단축메뉴 > **Source** > **Override/Implement Methods**

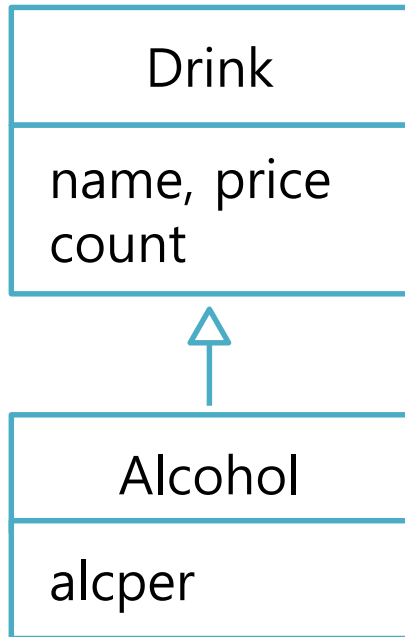
The image shows the Eclipse IDE interface. On the left, the 'Source' menu is open, and the 'Override/Implement Methods' option is highlighted. A blue arrow points from this menu item to a dialog box on the right titled 'Override/Implement Methods'. The dialog box has a search bar labeled 'Enter method name, prefix or pattern (*, ?)' and a 'Search Methods' button. Below this, it says 'Select methods to override or implement:' and lists two options: 'Car' (checked with a blue box) and 'Object' (unchecked). Under 'Car', the method 'showCarInfo()' is also checked with a blue box.



매출전표 만들기

■ 매출전표(salestatement) 만들기

- Drink(음료) 클래스 만들기
- Drink를 상속한 Alcohol(술) 클래스 만들기
- protected 접근 제어자 사용



상품명	가격	수량	금액
커피	2500	4	10000
녹차	3500	3	10500
상품명(도수[%])	가격	수량	금액
소주(15.2)	4000	5	20000

합계 금액 : 40500원			



매출 전표 – Drink 클래스

▪ Drink 클래스

```
public class Drink {  
    //protected는 상속받는 클래스에서 접근 가능  
    protected String name;    //상품명  
    protected int price;      //가격  
    protected int quantity;    //수량  
  
    Drink(String name, int price, int quantity){  
        this.name = name;  
        this.price = price;  
        this.quantity = quantity;  
    }  
  
    int calcPrice() {  
        return price * quantity;    //금액 = 가격 * 수량  
    }  
  
    static void printTitle() { //제목 출력  
        System.out.println("상품명\t가격\t수량\t금액");  
    }  
  
    void printData() { //데이터 출력  
        System.out.println(name + "\t" + price + "\t" +  
            quantity + "\t" + calcPrice());  
    }  
}
```



매출 전표 – Alcohol 클래스

■ Alcohol 클래스

```
public class Alcohol extends Drink{

    float alcper; //알콜 도수

    Alcohol(String name, int price, int quantity, float alcper){
        super(name, price, quantity);
        this.alcper = alcper;
    }

    static void printTitle() {
        System.out.println("상품명(도수[%])\t가격\t수량\t금액");
    }

    @Override
    void printData() {
        System.out.println(name + "(" + alcper + ")\t" + price +
            "\t" + quantity + "\t" + calcPrice());
    }
}
```



매출 전표 – Main 클래스

▪ SaleStatement 클래스

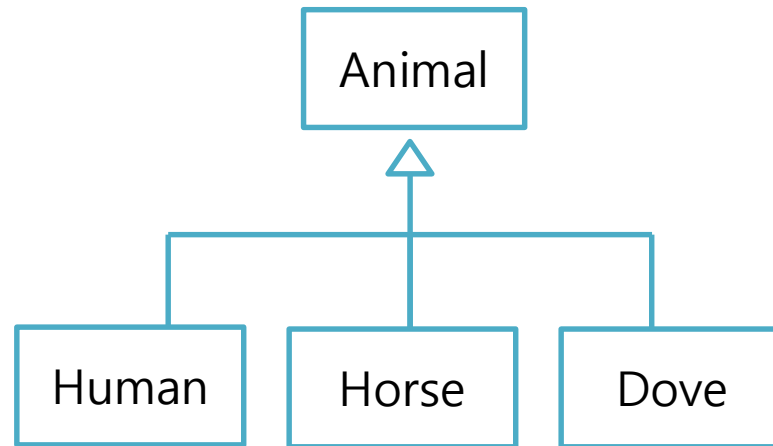
```
public class SaleStatement {  
  
    public static void main(String[] args) {  
  
        Drink coffee = new Drink("커피", 2500, 4);  
        Drink tea = new Drink("녹차", 3500, 3);  
        Alcohol soju = new Alcohol("소주", 4000, 5, 15.2f);  
  
        Drink.printTitle(); //클래스 이름으로 직접 접근  
        coffee.printData();  
        tea.printData();  
        System.out.println();  
  
        Alcohol.printTitle();  
        soju.printData();  
  
        //총금액 계산하기  
        int total = 0;  
        total = coffee.calcPrice() + tea.calcPrice() + soju.calcPrice();  
        System.out.println("***** 합계 금액 : " + total + "원 *****");  
    }  
}
```



다형성(polymorphism)

● 다형성이란?

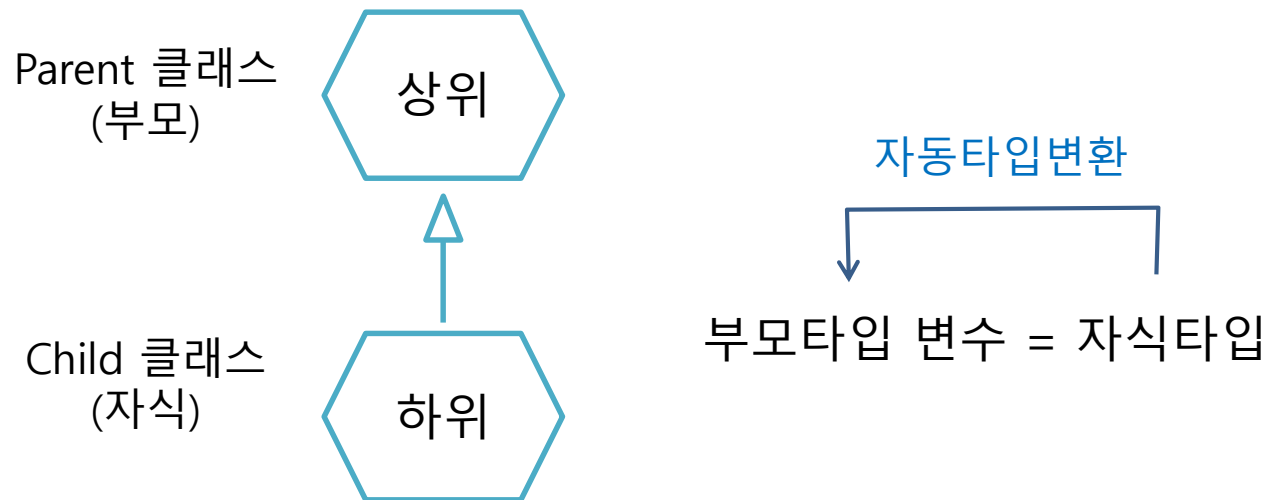
- 다형성(polymorphism)이란 하나의 타입(자료형)에 대입되는 객체에 따라서 실행결과가 다양한 형태로 나오는 성질을 말한다.
- 장점 : 코드의 재사용성 향상, 유지 보수 용이



자동 타입 변환

● 타입변환이란?

타입 변환이란 다른 타입으로 변환하는 행위를 말한다. 클래스도 기본타입 처럼 형 변환을 하는데 상속 관계에 있는 클래스 사이에서 발생한다.



다형성(polymorphism)

● 매개변수의 다형성

매개값을 다양화하기 위해 매개변수를 부모타입으로 선언하고 호출할때 자식객체를 대입한다.

```
package polymorphism;

class Animal{
    public void move() {
        System.out.println("동물이 움직입니다.");
    }
}

class Human extends Animal{
    public void move() {
        System.out.println("사람이 두 발로 걷습니다.");
    }
}

class Horse extends Animal{
    public void move() {
        System.out.println("말이 네 발로 뛴니다.");
    }
}
```



다형성(polymorphism)

● 매개변수의 다형성

```
public class AnimalTest {  
    //매개 변수의 다형성 - (Animal animal)  
    public void moveAnimal(Animal animal) {  
        animal.move();  
    }  
  
    public static void main(String[] args) {  
        AnimalTest aTest = new AnimalTest();  
        //부모 타입 = 자식 타입 (자동 형변환)  
        /*Animal human = new Human();  
        Animal horse = new Horse();  
  
        aTest.moveAnimal(human);  
        aTest.moveAnimal(horse);*/  
  
        aTest.moveAnimal(new Human());  
        aTest.moveAnimal(new Horse());  
    }  
}
```

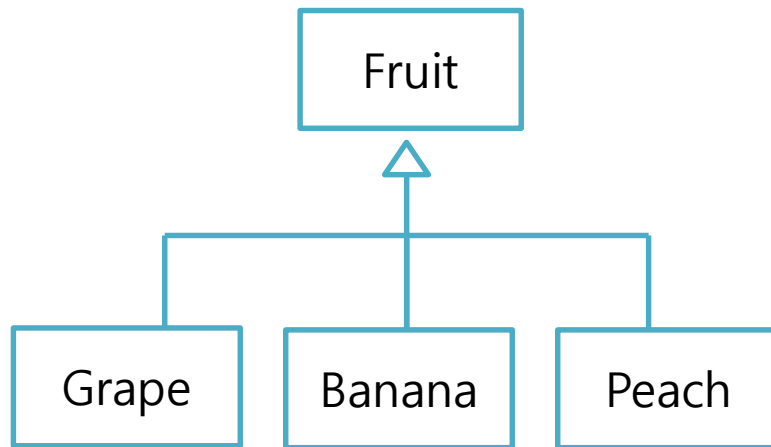
매개변수의 다형성

사람이 두 발로 걷습니다.
말이 네 발로 뜀니다.



다형성 예제

- 과일의 종류를 선택하는 다형성 예제



```
=====
1. 포도 | 2. 바나나 | 3. 복숭아
=====
선택>
2
과일 이름 : 바나나
과일 무게 : 650g
과일 가격 : 3000
```



다형성 예제

- Fruit 클래스

```
package polymorphism.fruit;

public class Fruit {
    String name;
    String weight;
    int price;

    public Fruit() {}

    public void showInfo() {
        System.out.println("과일 이름: " + name);
        System.out.println("과일 무게: " + weight);
        System.out.println("과일 가격: " + price);
    }
}
```



다형성 예제

- Fruit 클래스를 상속받은 Grape, Banana, Peach 클래스

```
public class Grape extends Fruit{  
  
    public Grape() {  
        name = "포도";  
        weight = "700g";  
        price = 6000;  
    }  
}
```

```
public class Banana extends Fruit{  
  
    public Banana() {  
        name = "바나나";  
        weight = "650g";  
        price = 3000;  
    }  
}
```

```
public class Peach extends Fruit{  
  
    public Peach() {  
        name = "복숭아";  
        weight = "900g";  
        price = 7500;  
    }  
}
```



다형성 예제

- FruitTest 클래스

```
Scanner scanner = new Scanner(System.in);

System.out.println("1. 포도 | 2. 복숭아 | 3. 바나나");
System.out.print("선택> ");

// try ~ finally (예외 처리)
// 스캐너 리소스 누수 방지 및 메뉴 잘못 선택시 프로그램 종료)
try {
    int menu = scanner.nextInt(); //메뉴 선택

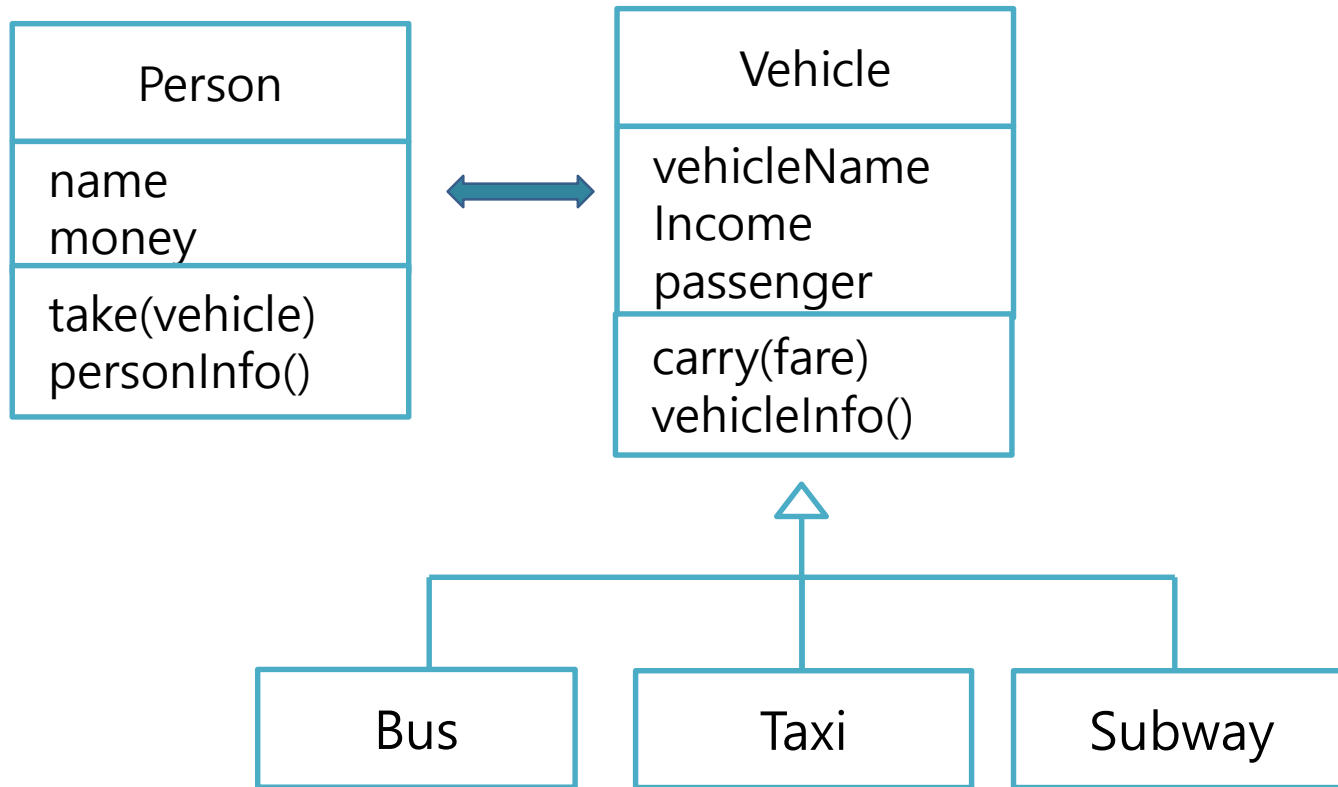
    Fruit fruit = null; //부모 타입 객체 선언
    if(menu == 1) {
        fruit = new Grape(); //자식 타입 객체 생성
    }else if(menu == 2) {
        fruit = new Peach();
    }else if(menu == 3) {
        fruit = new Banana();
    }else {
        System.out.println("올바른 메뉴를 선택하세요.");
        return; //프로그램 종료
    }
    fruit.showInfo();
}finally { //항상 닫기
    scanner.close();
}
```



교통수단 이용하기

❖ 버스, 택시, 지하철 교통수단 이용하기

- 사람이 교통 수단을 이용한다.
- 차량은 사람을 태우고 수입을 얻고, 승객수가 증가한다.



교통수단 이용하기

▪ Vehicle 클래스

```
public class Vehicle {  
    String vehicleName; //차량 이름  
    int income;          //수입  
    int passenger;       //승객수  
  
    public Vehicle(String vehicleName) {  
        this.vehicleName = vehicleName;  
    }  
  
    //승객 운송  
    public void carry(int fare) {  
        this.income += fare; //요금  
        passenger++;  
    }  
  
    public void vehicleInfo() {  
        System.out.printf("%s의 수입은 %,d원이고, 승객수는 %d명입니다.\n",  
                            vehicleName, income, passenger);  
    }  
}
```



교통수단 이용하기

▪ Bus, Taxi 클래스

```
public class Bus extends Vehicle{  
    public Bus(String vehicleName) {  
        super(vehicleName);  
    }  
}
```

```
public class Taxi extends Vehicle{  
    public Taxi(String vehicleName) {  
        super(vehicleName);  
    }  
}
```



교통수단 이용하기

▪ Person 클래스

```
public class Person {  
    private String name;  
    int money;  
  
    public Person(String name, int money) {  
        this.name = name;  
        this.money = money;  
    }  
  
    //요금을 내고 교통 수단 이용하기  
    public void take(Vehicle vehicle, int fare) {  
        vehicle.carry(fare);  
        this.money -= fare;  
    }  
  
    public void personInfo() {  
        System.out.printf("%s의 남은 돈은 %,d원입니다\n",  
            name, money);  
    }  
}
```



교통수단 이용하기

▪ Main 클래스

```
// 사람 객체 생성
Person p1 = new Person("이정후", 10000);
Person p2 = new Person("신유빈", 20000);

//버스와 택시 객체 생성
Bus bus330 = new Bus("bus330");
Taxi kakaoT = new Taxi("카카오T");

int fareOfBus = 1500;    //버스 요금
int fareOfTaxi = 6500;   //택시 요금

//교통 수단 이용하기 - 매개 변수의 다형성
p1.take(bus330, fareOfBus);
p2.take(bus330, fareOfBus);
p2.take(kakaoT, fareOfTaxi);

// 객체의 정보
p1.personInfo();
p2.personInfo();
bus330.vehicleInfo();
kakaoT.vehicleInfo();
```

이정후의 남은 돈은 8,500원입니다
신유빈의 남은 돈은 12,000원입니다
bus330의 수입은 3,000원이고, 승객수는 2명입니다.
카카오T의 수입은 6,500원이고, 승객수는 1명입니다.



강제 타입 변환

- 하위 클래스로 형 변환 -> 강제 타입변환(다운 캐스팅)

- 상위 클래스로 형 변환되었던 하위 클래스를 다시 원래 자료형으로 형 변환하는 것을 다운 캐스팅이라고 한다.
- 하위 클래스의 메소드를 사용해야 할 때 형 변환한다.
- **instanceof** 예약어 사용

```
if(animal instanceof Human) {  
    Human human = (Human)animal;  
    human.readBook();  
}
```



강제 타입 변환

- 부모에 없는 자식 클래스의 메서드 사용 예제

```
package downcasting;

class Animal{
    public void move() {
        System.out.println("동물이 움직입니다.");
    }
}

//Animal을 상속받은 Human 클래스 정의
class Human extends Animal{
    public void move() {
        System.out.println("사람이 두 발로 걷습니다.");
    }

    public void readBook() {
        System.out.println("사람이 책을 읽습니다.");
    }
}
```



강제 타입 변환

- 부모에 없는 자식 클래스의 메서드 사용 예제

```
//Animal을 상속받은 Horse 클래스 정의
class Horse extends Animal{
    public void move() {
        System.out.println("말이 네 발로 뛴니다.");
    }

    public void run() {
        System.out.println("말이 사람을 태우고 빠르게 달립니다.");
    }
}

//Animal을 상속받은 Eagle 클래스 정의
class Eagle extends Animal{
    public void move() {
        System.out.println("독수리가 하늘을 납니다.");
    }
    public void hunting() {
        System.out.println("독수리가 물고기를 사냥합니다.");
    }
}
```



강제 타입 변환

- 부모에 없는 자식 클래스의 메서드 사용 예제

```
public class AnimalTest {
    static Animal[] animals = new Animal[3]; //객체를 저장할 배열 생성

    public static void main(String[] args) {
        //객체를 생성하여 저장(다형성)
        animals[0] = new Human();
        animals[1] = new Horse();
        animals[2] = new Eagle();

        //부모 타입으로 객체를 생성하므로 자식 객체의 메서드에 접근할 수 없음
        animals[0].move();
        animals[1].move();
        animals[2].move();
        //readBook()에 접근할 수 없음

        /*Animal animal = animals[0]; //new Human();

        if(animal instanceof Human) { //animal이 Human의 객체라면
            Human human = (Human)animal; //Human 타입으로 다운 캐스팅
            human.readBook();
        }*/
    }
}
```



강제 타입 변환

■ 부모에 없는 자식 클래스의 메서드 사용 예제

```
System.out.println("=== 원래 형으로 다운캐스팅 ===");
downCasting(); //downCasting() 호출

} //main 닫기

//다운 캐스팅 메서드로 정의
public static void downCasting() {
    //하위 클래스로 형변환 - instanceof 키워드 사용
    for(int i = 0; i < animals.length; i++) {
        Animal animal = animals[i];
        if(animal instanceof Human) {
            Human human = (Human)animal;
            human.readBook();
        }else if(animal instanceof Horse) {
            Horse horse = (Horse)animal;
            horse.run();
        }else if(animal instanceof Eagle) {
            Eagle eagle = (Eagle)animal;
            eagle.hunting();
        }else {
            System.out.println("지원하지 않는 타입입니다.");
        }
    }
}
```

사람이 두 발로 걷습니다.
말이 네 발로 뛰니다.
독수리가 하늘을 납니다.
===== 원래 형으로 다운캐스팅 =====
사람이 책을 읽습니다.
말이 사람을 태우고 빠르게 달립니다.
독수리가 물고기를 사냥합니다.

