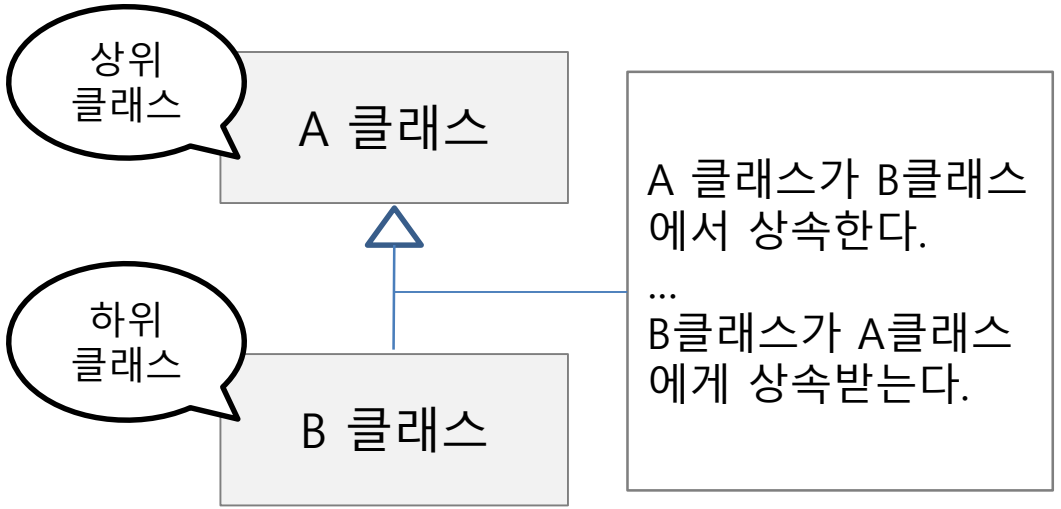
2장. 상속과 다형성

Inheritance & Polymorphism

상속(Inheritance)

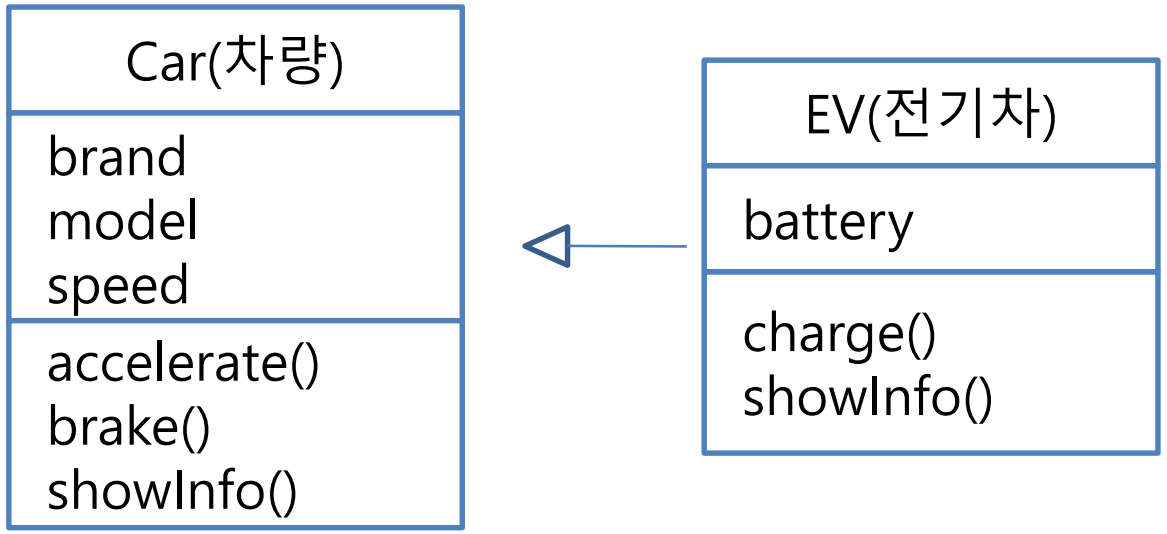
- **상속이란?**
  - 클래스를 정의할때 이미 구현된 클래스를 상속(inheritance) 받아서 속성이나 기능(메서드)이 확장되는 클래스를 구현할 수 있다.
  - 상속하는 클래스 : 상위 클래스, parent class
  - 상속받는 클래스 : 하위 클래스, child class
  - 클래스 상속 문법

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



차량과 전기차 클래스 상속

- **차량과 전기차 클래스 상속 다이어그램**



차량과 전기차 클래스 상속

- **테스트 결과 출력**

```
========== 자동차 정보 ==========
제조사: Kia
모델명: EV6
EV6 가속 - 현재 속도: 60km/h
EV6 감속 - 현재 속도: 30km/h
EV6 충전됨 - 배터리: 100%
========== 자동차 정보 ==========
제조사: Kia
모델명: EV6
```

차량과 전기차 클래스 상속

- **차량과 전기차 클래스 상속**

```
package inheritance.car;

public class Car {
    protected String brand; //브랜드명
    protected String model; //모델명
    protected int speed;    //속도

    public Car(String brand, String model) {
        this.brand = brand;
        this.model = model;
        this.speed = 0;
    }

    public void accelerate(int amount) { //가속 메서드
        speed += amount;
        System.out.println(model + " 가속 - 현재 속도: " + speed + "km/h");
    }
```

차량과 전기차 클래스 상속

- **차량과 전기차 클래스 상속**

```
    public void brake(int amount) { //감속 메서드
        speed -= amount;
        if(speed < 0)
            speed = 0;
        System.out.println(model + " 감속 - 현재 속도: " + speed + "km/h");
    }

    public void showInfo() {
        System.out.println("== 자동차 정보 ==");
        System.out.println("제조사: " + brand);
        System.out.println("모델명: " + model);
    }
}
```

차량과 전기차 클래스 상속

- 생성자 및 메서드 상속 - super() 사용

```
public class EV extends Car{
    private int battery;  //배터리 잔량(0~100%)

    public EV(String brand, String model, int battery) {
        super(brand, model);  //Car의 멤버 변수 상속
        this.battery = battery;
    }

    public void charge(int amount) {
        battery += amount;
        if (battery > 100)
            battery = 100;
        System.out.println(model + " 충전됨 - 배터리: " + battery + "%");
    }

    @Override //메서드 재정의
    public void showInfo() {
        super.showInfo(); //Car의 showInfo() 상속
        System.out.println("배터리 잔량: " + battery + "%");
    }
}
```

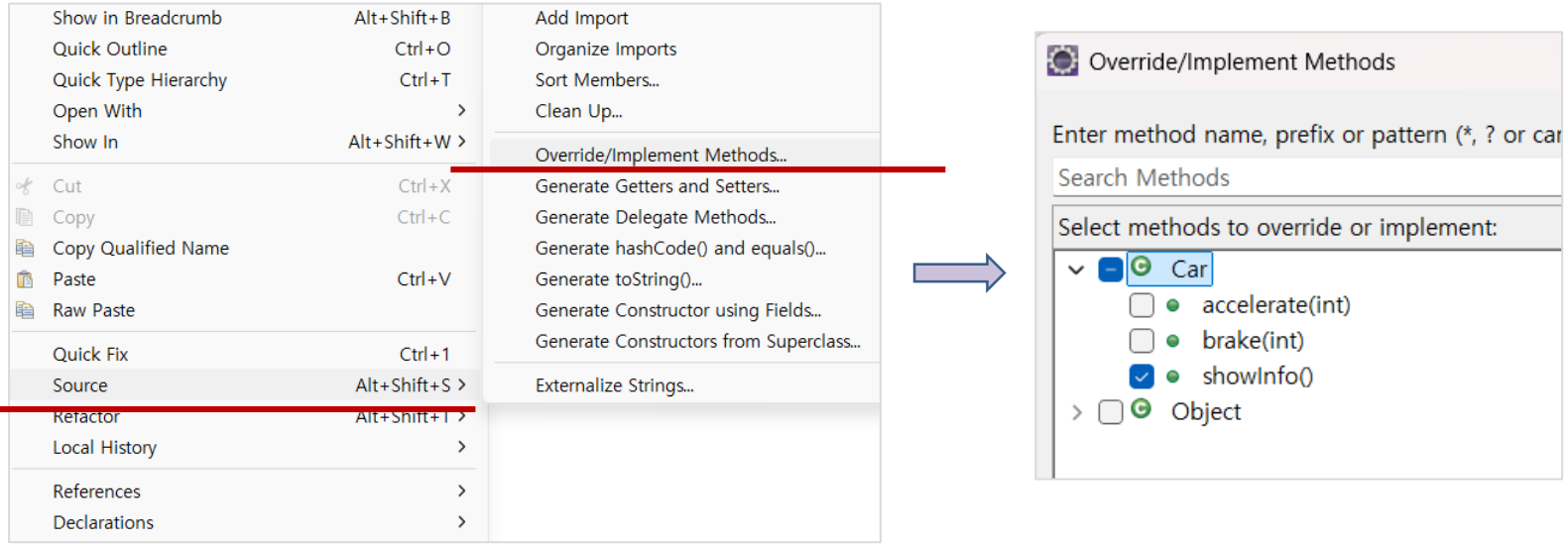
차량과 전기차 클래스 상속

- **Main Test**

```
EV myEV = new EV("Kia", "EV6", 50);

//출력
myEV.showInfo();

//기능 테스트
myEV.accelerate(60);
myEV.brake(30);
myEV.charge(60); //최대 100%

//테스트 후 출력
myEV.showInfo();
```

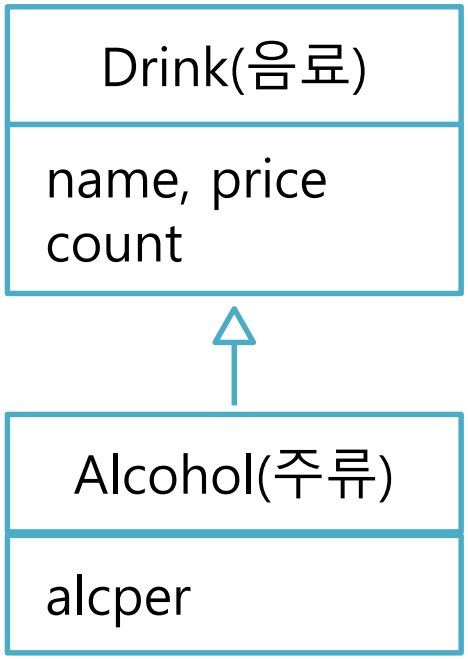
메서드 재정의(Overriding)

- **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**



매출 전표 만들기

- **매출전표(salestatement) 만들기**
  - Drink(음료) 클래스 만들기
  - Drink를 상속한 Alcohol(술) 클래스 만들기




```
상품명        가격        수량        금액
커피          2500        4           10000
녹차          3500        3           10500

상품명(도수[%])      가격        수량        금액
소주(15.2)           4000        2           8000
맥주(5.5)3000        3           9000
***** 합계 금액 : 28500원 *****
```

매출 전표 – Drink 클래스

- **Drink 클래스**

```java
package inheritance.salestatement;

public class Drink {
    protected String name;  //상품명
    protected int price;    //가격
    protected int quantity;     //수량

    public Drink(String name, int price, int quantity){
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public int calcPrice() {
        return price * quantity;  //금액 = 가격 * 수량
    }
```

매출 전표 – Drink 클래스

- **Drink 클래스**

```
    public static void printTitle() { //제목 출력
        System.out.println("상품명\t가격\t수량\t금액");
    }

    public 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{

    private float alcper; //알콜 도수

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

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

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

매출 전표 – Main 클래스

- **SaleStatement 테스트**

```
package inheritance.salestatement;

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, 2, 15.2f);
        Alcohol beer = new Alcohol("맥주", 3000, 3, 5.5f);

        Drink.printTitle();  //클래스 이름으로 직접 접근
        coffee.printData();
        tea.printData();
        System.out.println();
```

매출 전표 – Main 클래스

- **SaleStatement 테스트**

```
        Alcohol.printTitle();
        soju.printData();
        beer.printData();

        //총금액 계산하기
        int total = 0;
        total = coffee.calcPrice() + tea.calcPrice() + soju.calcPrice();
        System.out.println("***** 합계 금액 : " + total + "원 *****");
    }
}
```

실습 문제 – 상속

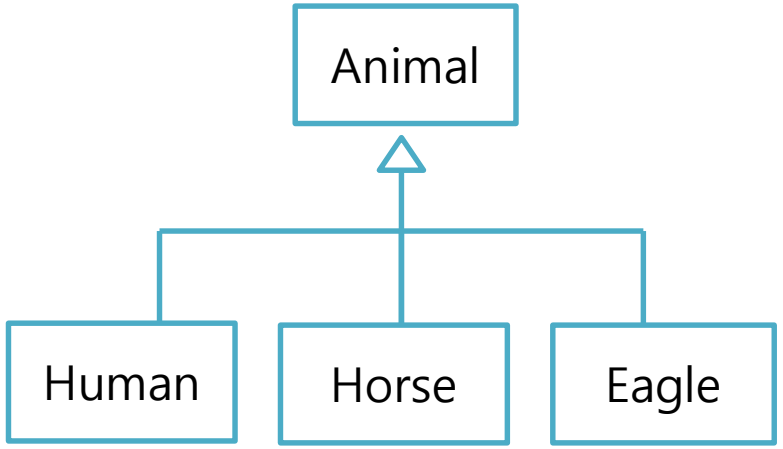
다음 코드는 오류가 발생합니다. 오류를 설명하고 해결해 보세요

```
class Car{
    protected String brand;
    protected int carNumber;

    public Car(String brand) {
        this.brand = brand;
    }

}

public class Taxi extends Car{

    private int passenger;

    public void setPassenger(int passenger) {
        this.passenger = passenger;
    }

    public int getPassenger() {
        return passenger;
    }
}
```

다형성(polymorphism)

다형성이란?

- **다형성(polymorphism)**이란 하나의 타입(자료형)에 대입되는 객체에 따라서 실행결과가 다양한 형태로 나오는 성질을 말한다.
- 동적 바인딩 – 메서드가 인스턴스의 타입에 맞게 실행되는 것을 말한다.
- 장점 : 코드의 재사용성 향상, 유지 보수 용이



다형성(polymorphism)

- **다형성 예제**

```
package polymorphism.animal;

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 뜁니다.");
    }
}
```

다형성(polymorphism)

- **다형성 – 인스턴스 생성**

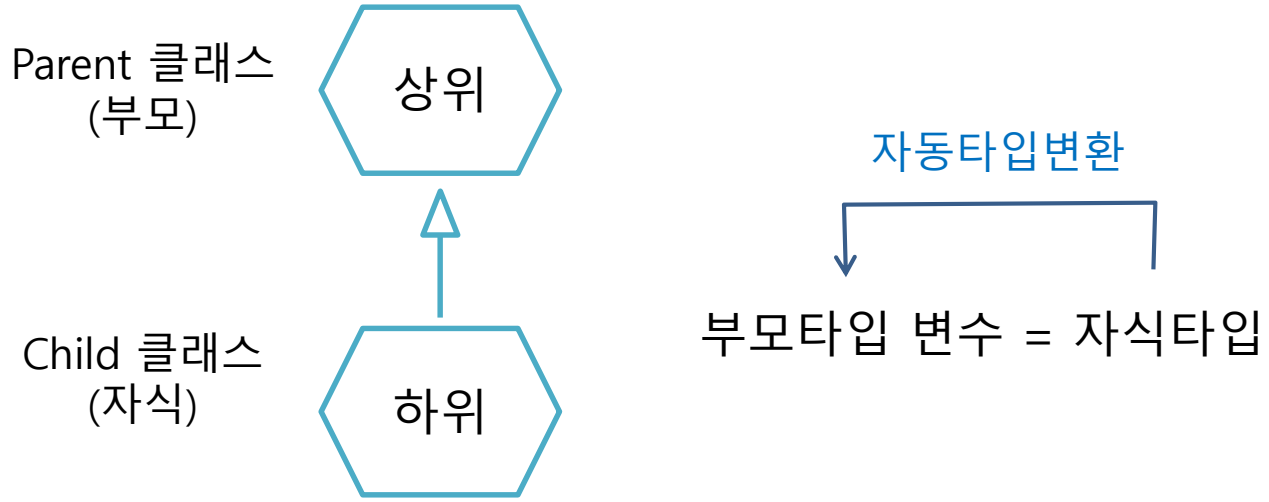
```
public class AnimalTest {
    public static void main(String[] args) {
        //다형성 -> 부모 타입 = 자식 타입
        /*Animal human = new Human();
        Animal horse = new Horse();

        human.move(); //메서드 - 동적 바인딩
        horse.move(); */

        //배열로 관리
        Animal[] animals = {
            new Human(),
            new Horse(),
            new Eagle()
        };

        for(Animal animal : animals)
            animal.move();
    }
}
```

```
사람이 두 발로 걷습니다.
말이 네 발로 뜁니다.
독수리가 날개를 쭉 펴고 날아갑니다.
```

자동 타입 변환

- **타입변환이란?**

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



다형성(polymorphism)

- **매개변수의 다형성**

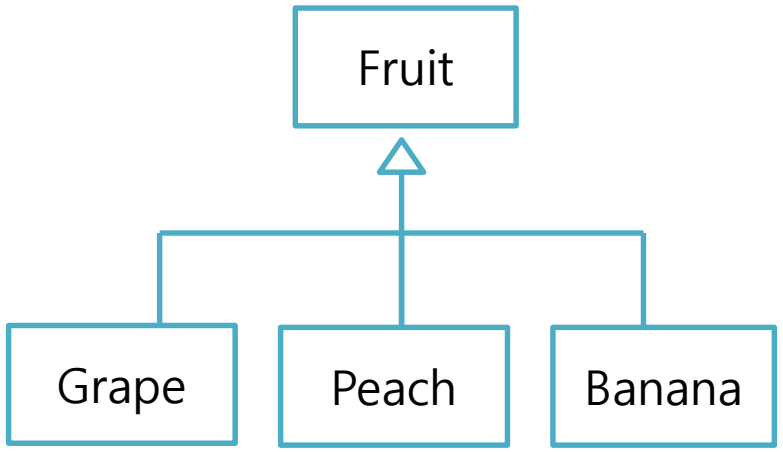
  매개값을 다양화하기 위해 매개변수를 부모타입으로 선언하고 호출할때 자식 객체를 대입한다.

```
public static void main(String[] args) {

    //다형성 -> 매개 변수의 다형성
    moveAnimal(new Human());
    moveAnimal(new Horse());
    moveAnimal(new Eagle());

}

//moveAnimal 메서드 정의
public static void moveAnimal(Animal animal) {
    animal.move();
}
```

매개변수의 다형성

다형성 예제

- 과일의 종류를 선택하는 다형성 예제




```
1.포도 | 2.복숭아 | 3.바나나
선택> 2
복숭아는 당도가 높고 부드럽습니다.
```

다형성 예제

- **Fruit, Grape 클래스**

```
public class Fruit {

    public void showInfo() {
        System.out.println("과일 정보 없음");
    }
}
```

```
public class Grape extends Fruit{

    @Override
    public void showInfo() {
        System.out.println("포도는 보라색이고, 달콤하고 건강에 좋습니다.");
    }

}
```

다형성 예제

- **Peach, Banana 클래스**

```
public class Peach extends Fruit{

    @Override
    public void showInfo() {
        System.out.println("복숭아는 당도가 높고 부드럽습니다.");
    }
}
```

```
public class Banana extends Fruit{

    @Override
    public void showInfo() {
        System.out.println("바나나는 노랑색이고 달콤합니다.");
    }
}
```

다형성 예제

- **FruitTest 클래스**

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

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

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();
}
fruit.showInfo();  //메서드 - 동적 바인딩

scanner.close();
```

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

```
class Animal{

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

//Animal을 상속받은 Human 클래스 정의
class Human extends Animal{

    @Override
    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

```
class Horse extends Animal{

    @Override
    public void move() {
        System.out.println("말이 네 발로 뜁니다.");
    }

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

class Eagle extends Animal{

    @Override
    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 뜁니다.
독수리가 하늘을 납니다.
===== 원래 형으로 다운캐스팅 =====
사람이 책을 읽습니다.
말이 사람을 태우고 빠르게 달립니다.
독수리가 물고기를 사냥합니다.
```