

# 8장. 상속과 다형성



객체지향 언어(OOP)

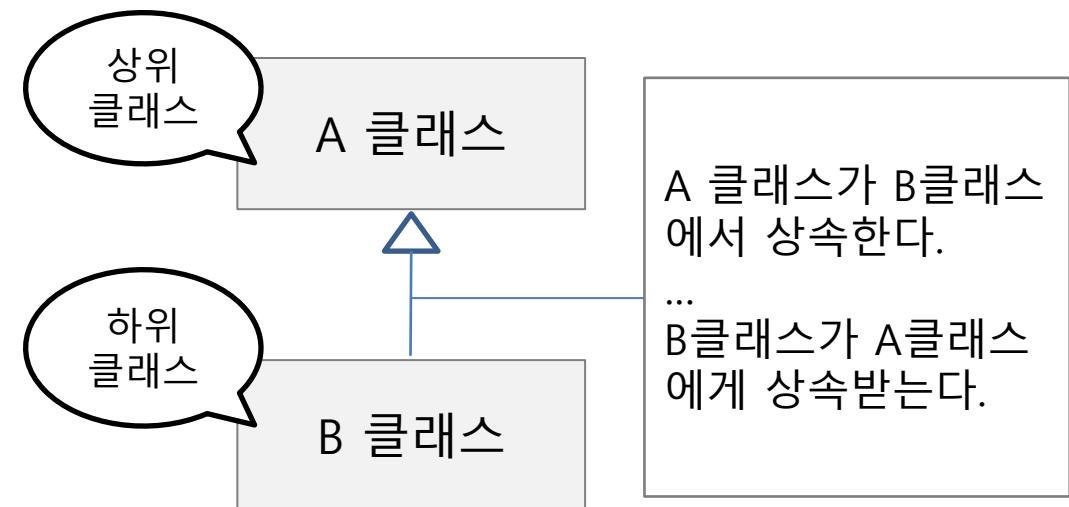


# 상속(Inheritance)

## ■ 상속이란?

- 클래스를 정의할 때 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{  
    ....  
}
```



# 상속(Inheritance)

## ■ 멤버 속성 상속

Person  
(이름, 나이)

Engineer  
(사번)

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

```
public class Engineer extends Person{  
    int companyID;  
}
```

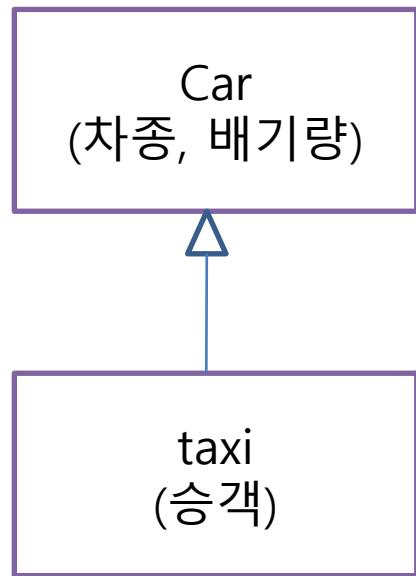
Person으로부터 상  
속받은 멤버변수

```
Engineer engineerKim = new Engineer();  
engineerKim.name = "봉구"; //부모 멤버 접근  
engineerKim.age = 27;  
engineerKim.companyID = 256;
```



# Super 예약어

## ■ 매개변수 있는 생성자 상속 - super()



```
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;  
    }  
}
```

The code illustrates the use of the `super()` constructor in Java. The `Car` class has two instance variables: `String brand` and `int cc`. Its constructor takes `String brand` and `int cc` as parameters and initializes the `this.brand` and `this.cc` fields. The `Taxi` class extends `Car` and has its own instance variable `int passenger`. Its constructor takes `String brand`, `int cc`, and `int passenger` as parameters. It uses `super(brand, cc)` to call the `Car` constructor and initialize the `brand` and `cc` fields. It then initializes its own `passenger` field with the value passed in the constructor.

People으로부터 상  
속받은 멤버변수

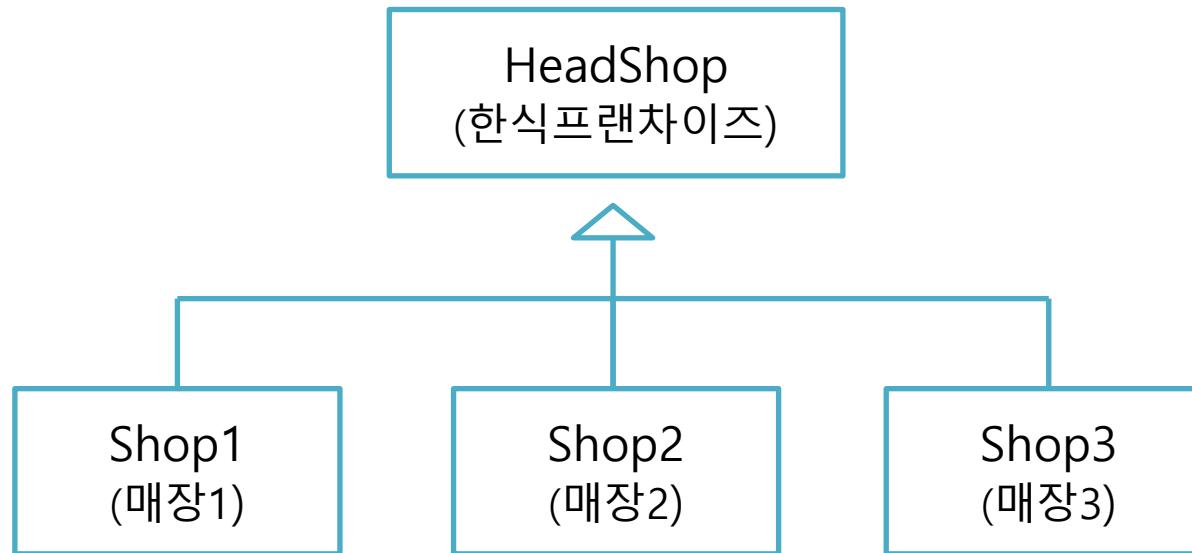


# 메소드 재정의

## ■ 메소드 상속 및 재정의(Method Overriding)

- 상속된 메소드의 내용이 자식 클래스에 맞지 않을 경우, 자식 클래스에서 동일한 메소드를 재정의 하는 것을 말한다.

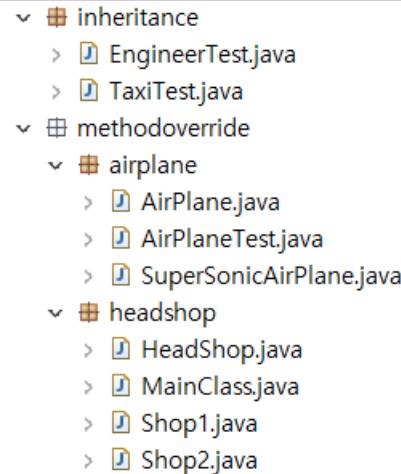
◆ 체인점 사업을 통한 상속의 예.



# 메소드 재정의

## ■ 메소드 재정의(Method Overriding)

@Override 애너테이션을 붙여서 컴파일러에게 재정의한 것을 알려준다.

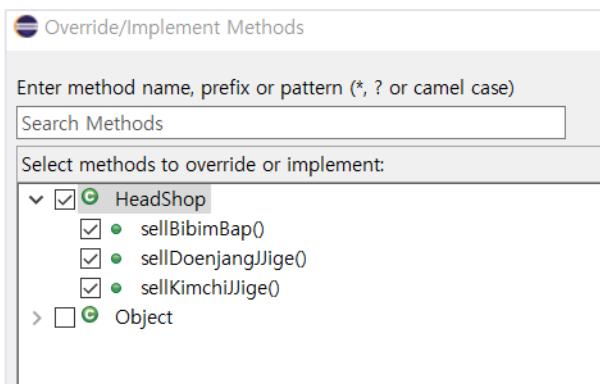
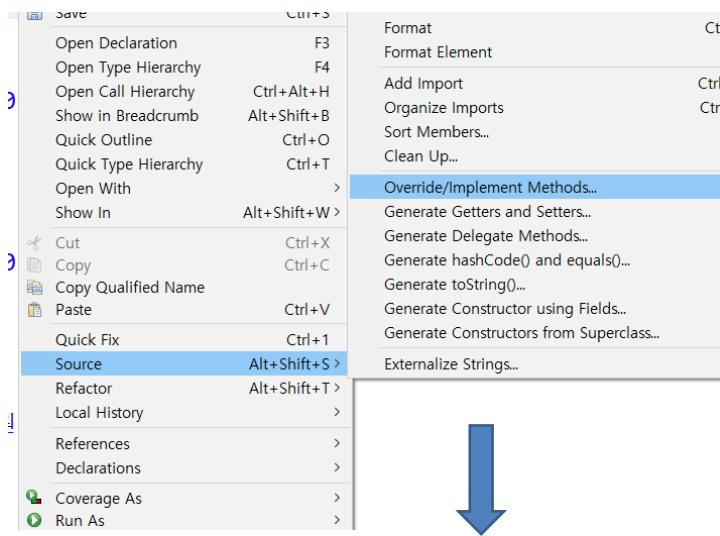


```
public class HeadShop {  
  
    public HeadShop() {  
    }  
  
    public void sellDoenjangJjige() {  
        System.out.println("된장찌개 : 5,500원");  
    }  
  
    public void sellKimchiJjige() {  
        System.out.println("김치찌개 : 6,000원");  
    }  
  
    public void sellBibimBap() {  
        System.out.println("비빔밥 : 6,500원");  
    }  
}
```



# 메소드 재정의

## ■ 메소드 재정의(Method Overriding)



```
public class Shop1 extends HeadShop{  
  
    public Shop1() {  
        System.out.println("대학가 매장입니다.");  
    }  
  
    @Override  
    public void sellDoenjangJjige() {  
        System.out.println("된장찌게 : 5,000원");  
    }  
  
    @Override  
    public void sellKimchiJjige() {  
        System.out.println("김치찌게 : 5,500원");  
    }  
  
    @Override  
    public void sellBibimBap() {  
        System.out.println("비빔밥 : 6,000원");  
    }  
}
```



# 메소드 재정의

## ■ 메소드 재정의(Method Overriding)

```
public class Shop2 extends HeadShop{

    public Shop2() {
        System.out.println("역세권 매장입니다.");
    }

    @Override
    public void sellDoenjangJJige() {
        System.out.println("된장찌게 : 6,000원");
    }

    @Override
    public void sellKimchiJJige() {
        System.out.println("김치찌게 : 6,500원");
    }

    @Override
    public void sellBibimBap() {
        System.out.println("비빔밥 : 7,000원");
    }
}
```



# 메소드 재정의

## ▪ 체인점 사업을 통한 상속

MainClass.java

```
HeadShop shop1 = new Shop1();
shop1.sellDoenjangJJige();
shop1.sellKimchiJJige();
shop1.sellBibimBap();
System.out.println("=====");  
  
HeadShop shop2 = new Shop2();
shop2.sellDoenjangJJige();
shop2.sellKimchiJJige();
shop2.sellBibimBap();
System.out.println("=====");
```

대학가 매장입니다.

된장찌개 : 5,000원

김치찌개 : 5,500원

비빔밥 : 6,000원

=====

역세권 매장입니다.

된장찌개 : 6,000원

김치찌개 : 6,500원

비빔밥 : 7,000원

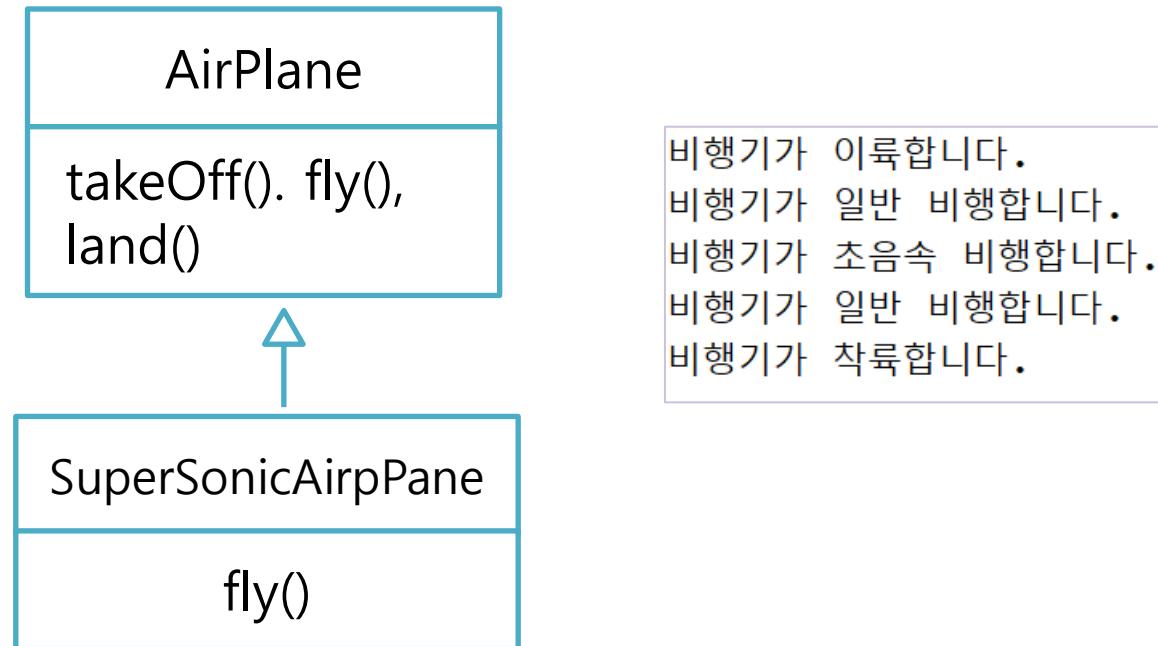
=====



# 비행기의 비행모드 바꾸기

## 비행모드 전환하기

- AirPlane 클래스 만들기
- AriPlane을 상속한 SuperSonicAirplane 만들기



# 메서드 상속

## ➤ AirPlane 클래스

```
public class AirPlane {  
  
    public void takeOff() {  
        System.out.println("이륙합니다.");  
    }  
  
    public void fly() {  
        System.out.println("일반 비행합니다.");  
    }  
  
    public void land() {  
        System.out.println("착륙합니다.");  
    }  
}
```



# 메서드 상속

## ➤ SuperSonicAirPlane 클래스

```
public class SuperSonicAirPlane extends AirPlane{
    public static final int NORMAL = 1;
    public static final int SUPERSONIC = 2;

    int flyMode = NORMAL;

    @Override
    public void fly() {
        if(flyMode==SUPERSONIC) {
            System.out.println("비행기가 초음속 비행합니다.");
        }
        else {
            super.fly();
        }
    }
}
```



# 메서드 상속

## ➤ AirPlaneTest 클래스

```
public class AirPlaneTest {  
    public static void main(String[] args) {  
        SuperSonicAirPlane sa = new SuperSonicAirPlane();  
        sa.takeOff();  
        sa.fly();  
        sa.flyMode = SuperSonicAirPlane.SUPERSONIC;  
        sa.fly();  
  
        sa.flyMode = SuperSonicAirPlane.NORMAL;  
        sa.fly();  
        sa.land();  
    }  
}
```

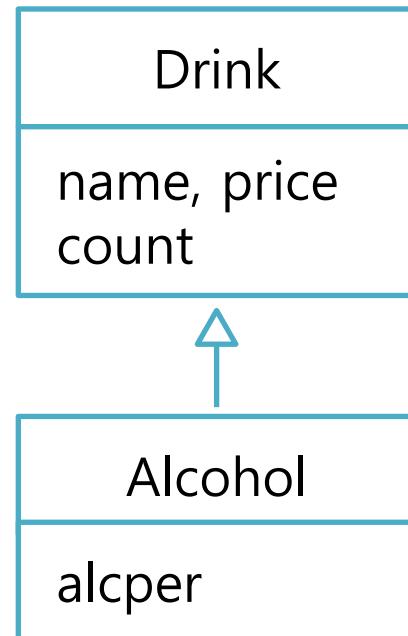
비행기가 이륙합니다.  
비행기가 일반 비행합니다.  
비행기가 초음속 비행합니다.  
비행기가 일반 비행합니다.  
비행기가 착륙합니다.



# 매출전표 만들기

## 매출 전표 만들기

- Drink(음료) 클래스 만들기
- Drink를 상속한 Alcohol(술) 클래스 만들기



상품명	가격	수량	금액
커피	2500	10	25000
녹차	3000	4	12000
상품명(도수[%])	가격	수량	금액
소주(17.3)	4000	5	20000
*** 합계 금액 57000원 ***			



# 메서드 상속

## Drink 클래스

```
package salestatement;

public class Drink { //음료
    String name; //상품명
    int price; //가격
    int count; //수량

    Drink(String name, int price, int count) { //생성자
        this.name = name;
        this.price = price;
        this.count = count;
    }

    int getTotalPrice() { //금액 계산

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

        void printData() { //데이터 출력
            System.out.println(name + "\t" + price + "\t" +
                               count + "\t" + getTotalPrice());
        }
    }
}
```



# 메서드 상속

## Alcohol 클래스

```
package salestatement;

public class Alcohol extends Drink{

    float alcper;      //알콜 도수

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

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

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



# 메서드 상속

## Payment 클래스

```
public class SaleStatement {
    public static void main(String[] args) {
        Drink coffee = new Drink("커피", 2500, 10);
        Drink tea = new Drink("녹차", 3000, 4);
        Alcohol soju = new Alcohol("소주", 4000, 5, 17.3f);

        Drink.printTitle();
        coffee.printData();
        tea.printData();

        System.out.println();

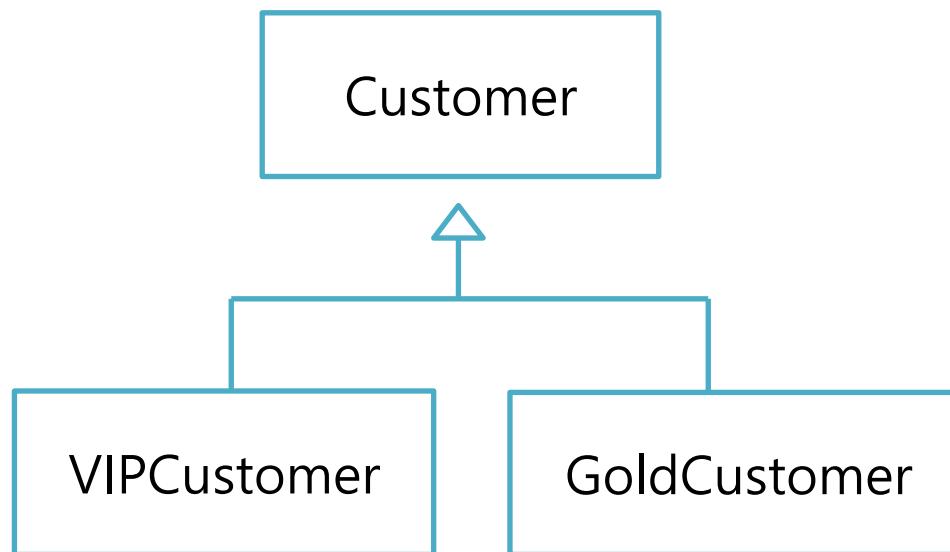
        Alcohol.printTitle();
        soju.printData();
        int sum = coffee.getTotalPrice() + tea.getTotalPrice() +
                  soju.getTotalPrice();
        System.out.println("\n*** 합계 금액 " + sum + "원 ***");
    }
}
```



# 고객관리 프로그램

## ● 상속을 활용한 고객관리 프로그램

- 고객의 정보를 활용하여 고객 맞춤 서비스를 구현
- 고객의 등급에 따라 차별화된 할 일과 포인트를 지급



# 고객 관리 프로그램

## Customer 클래스

### 예제 시나리오

고객 등급은 Silver 이고, 다음과 같은 혜택을 제공합니다.

- 보너스 포인트를 1% 적립해 줍니다.

멤버 변수	설 명
customerID	고객 아이디
customerName	고객 이름
customerGrade	고객 등급
bonusPoint	고객의 보너스 포인트(마일리지)
bonusRatio	고객의 포인트 적립 비율



# 고객 관리 프로그램

## Customer 클래스

```
public class Customer {  
    protected int customerID;  
    protected String customerName;  
    protected String customerGrade;  
    int bonusPoint;  
    double bonusRatio;  
  
    public Customer() {  
        customerGrade = "SILVER";  
        bonusRatio = 0.01;  
    }  
  
    public void setCustomerName(String customerName) {  
        this.customerName = customerName;  
    }  
  
    public int calcPrice(int price) {  
        //보너스 포인트 = 가격 x 보너스할인율  
        bonusPoint += price * bonusRatio;  
        return price;  
    }  
  
    public String showInfo() {  
        return customerName + "님의 등급은 " + customerGrade +  
               "이고, 보너스 포인트는 " + bonusPoint + "입니다.";  
    }  
}
```



# 고객 관리 프로그램

## CustomerTest 클래스

```
public class CustomerTest {  
    public static void main(String[] args) {  
        Customer customerLee = new Customer();  
        customerLee.setCustomerName("이순신");  
  
        int price = 10000; //10000원 구매  
        customerLee.calcPrice(price);  
  
        System.out.println(customerLee.showInfo());  
    }  
}
```

이순신님의 등급은 SILVER이며, 보너스포인트는 100입니다.



# 고객 관리 프로그램

## VIPCustomer 클래스

### 예제 시나리오

고객이 점점 늘어나고 판매도 많아지다 보니 단골 고객이 생겼습니다. 단골 고객은 회사 매출에 많은 기여를 하는 우수 고객입니다. 우수 고객 등급은 VIP이고, 다음과 같은 혜택을 제공합니다.

- 제품을 살 때는 항상 10%를 할인해 줍니다.
- 보너스 포인트를 5% 적립해 줍니다.
- 담당 전문 상담원을 배정합니다.



# 고객 관리 프로그램

## VIPCustomer 클래스

```
public class VIPCustomer extends Customer {
    private int agentID; //VIP 고객 상담원 아이디
    double saleRatio; //구매 할인율
    public VIPCustomer() {
        super(); //기본생성자이므로 생략 가능
        customerGrade = "VIP";
        bonusRatio = 0.05;
        saleRatio = 0.1;
    }
    @Override
    public void setAgentID(int agentID) {
        this.agentID = agentID;
    }
    @Override
    public int calcPrice(int price) {
        price -= (int)(price*saleRatio); //가격 = 가격 - 할인가격
        bonusPoint += price * bonusRatio;
        return price;
    }
    @Override
    public String showInfo() {
        return super.showInfo() + "담당 상담원 ID는 " + agentID + "입니다." ;
    }
}
```



# 고객 관리 프로그램

## VIPCustomerTest 클래스

```
public class VIPCustomerTest {  
    public static void main(String[] args) {  
        VIPCustomer customerShin = new VIPCustomer();  
        customerShin.setCustomerName("신사임당");  
        customerShin.setAgentID(12345);  
  
        int price = 10000;  
        int shinPrice = customerShin.calcPrice(price);  
  
        System.out.println("지불해야하는 금액은 " + shinPrice + "입니다.");  
        System.out.println(customerShin.showInfo());  
    }  
}
```

지불해야하는 금액은 9000원입니다.

신사임당님의 등급은 VIP이고, 보너스 포인트는 450입니다. 담당 상담원 ID는 12345입니다.



# 고객 관리 프로그램

## 매개 변수가 있는 생성자로 구현하기 – Customer 클래스

```
public class Customer {  
    private int customerID;          //고객 아이디  
    private String customerName;     //고객 이름  
    protected String customerGrade; //고객 등급  
    int bonusPoint;                 //보너스 포인트  
    double bonusRatio;              //보너스 적립율  
  
    public Customer() { //기본 생성자  
        customerGrade = "SILVER"; //실버  
        bonusRatio = 0.01;         //보너스 적립율 1%  
    }  
  
    public Customer(int customerID, String customerName) {  
        this.customerID = customerID;  
        this.customerName = customerName;  
        customerGrade = "SILVER";  
        bonusRatio = 0.01;  
    }  
}
```



# 고객 관리 프로그램

## 매개 변수가 있는 생성자로 구현하기 – VIPCustomer 클래스

```
public class VIPCustomer extends Customer{
    private int agentID;
    double saleRatio;      //구매 할인율

    public VIPCustomer() {
        super();
        customerGrade = "VIP";
        bonusRatio = 0.05;
        saleRatio = 0.1;
    }

    public VIPCustomer(int customerID, String customerName, int agentID) {
        super(customerID, customerName);
        this.agentID = agentID;
        customerGrade = "VIP";
        bonusRatio = 0.05;
        saleRatio = 0.1;
    }
}
```



# 고객 관리 프로그램

## 매개 변수가 있는 생성자로 테스트하기

```
public class CustomerTest {  
  
    public static void main(String[] args) {  
        //매개변수가 있는 생성자로 구현하기  
        Customer c = new Customer(1001, "이대한");  
        VIPCustomer vip = new VIPCustomer(1002, "장민국", 777);  
  
        int price = 10000;  
        c.calcPrice(price);  
  
        int saledPrice = vip.calcPrice(price);  
  
        System.out.println(c.showInfo());  
        System.out.println();  
        System.out.printf(vip.getCustomerName() + "님의 구매 가격 : %,d원\n", saledPrice);  
        System.out.println(vip.showInfo());  
    }  
}
```



# 묵시적 클래스 형 변환

## ■ 묵시적 클래스 형변환(자동 형변환) → 다형성으로 확장

상속에서 상위 클래스와 하위 클래스에 같은 이름의 메서드가 존재할 때 호출되는 메서드는 인스턴스에 따라 결정된다.

```
public class CustomerTest2 {  
  
    public static void main(String[] args) {  
        //부모 클래스로 자식 클래스의 인스턴스 생성하기 - 자동 형변환  
        Customer c = new Customer(1001, "이대한");  
        Customer vip = new VIPCustomer(1002, "장민국", 777);  
  
        int price = 10000;  
        c.calcPrice(price);  
  
        int saledPrice = vip.calcPrice(price);  
  
        System.out.println(c.showInfo());  
        System.out.println();  
        System.out.printf(vip.getCustomerName() + "님의 구매 가격 : %,d원\n", saledPrice);  
        System.out.println(vip.showInfo());  
    }  
}
```



# 가상 메서드

## ■ 가상 메서드

클래스를 생성하여 인스턴스가 만들어지면 멤버 변수는 힙 메모리에 위치한다. 그러나 메서드는 메모리의 데이터 영역에 위치한다.

```
package virtualfunction;

public class TestA {
    int num;

    void aaa() {
        System.out.println("aaa() 출력");
    }

    public static void main(String[] args) {
        TestA a1 = new TestA();
        a1.num = 10;
        a1.aaa();

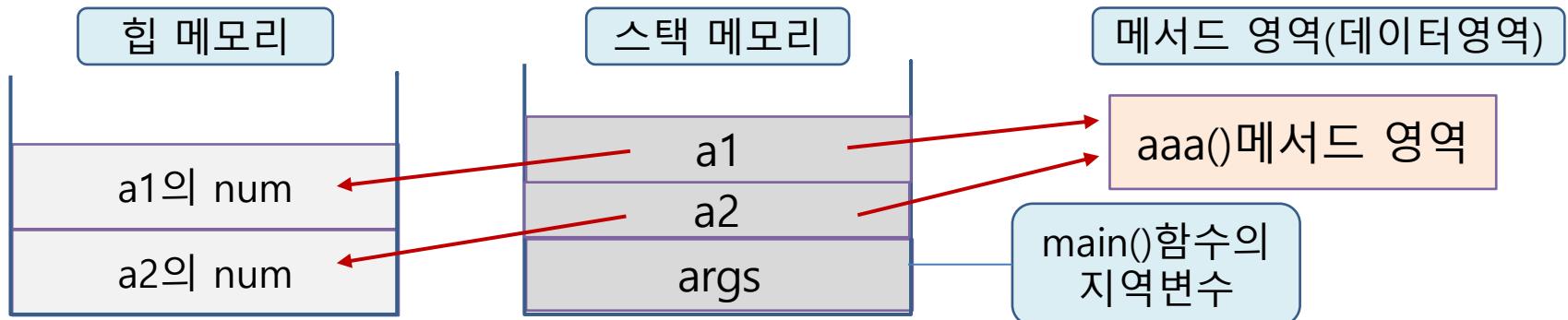
        TestA a2 = new TestA();
        a2.num = 20;
        a2.aaa();
    }
}
```

aaa() 출력  
aaa() 출력



# 가상 메서드

## ■ 가상 메서드



main() 함수가 실행되면 지역 변수는 스택 메모리에 위치한다.

참조 변수 `a1`과 `a2`가 가리키는 인스턴스는 힙 메모리에 생성된다.

그런데 메서드는 데이터 영역 메모리에 위치하고, 메서드를 호출하면

메서드의 영역 주소를 참조하여 명령이 실행된다.

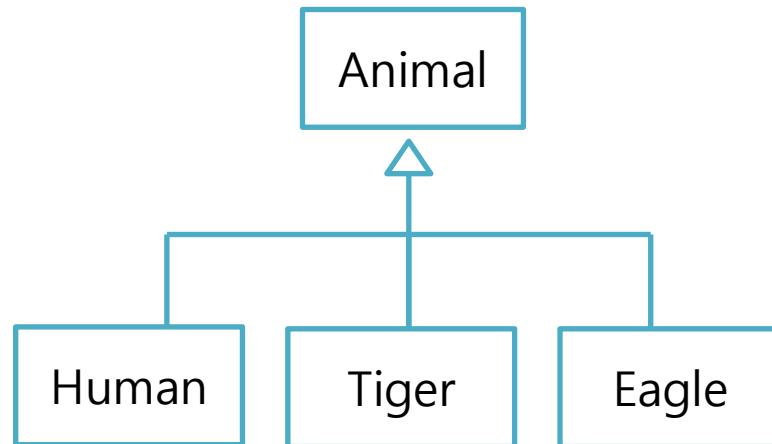
따라서 인스턴스가 달라도 동일한 메서드가 호출된다.



# 다형성(polymorphism)

## ● 다형성이란?

- **다형성(polymorphism)**이란 하나의 타입(자료형)에 대입되는 객체에 따라서 실행결과가 다양한 형태로 나오는 성질을 말한다.
- 묵시적 클래스 형 변환과 가상메서드를 바탕으로 다형성을 구현할 수 있다.



# 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 Eagle extends Animal{
    public void move() {
        System.out.println("독수리가 하늘을 납니다.");
    }
}
```



# 다형성(polymorphism)

```
class Tiger extends Animal{
    public void move() {
        System.out.println("호랑이가 네 발로 뛽니다.");
    }
}

public class AnimalTest {
    //매개변수의 자료형이 상위 클래스
    public void moveAnimal(Animal animal){
        animal.move();
    }

    public static void main(String[] args) {
        AnimalTest aTest = new AnimalTest();
        Animal human = new Human();
        Animal eagle = new Eagle();
        Animal tiger = new Tiger();

        aTest.moveAnimal(human);
        aTest.moveAnimal(eagle);
        aTest.moveAnimal(tiger);
    }
}
```

매개변수의 다형성

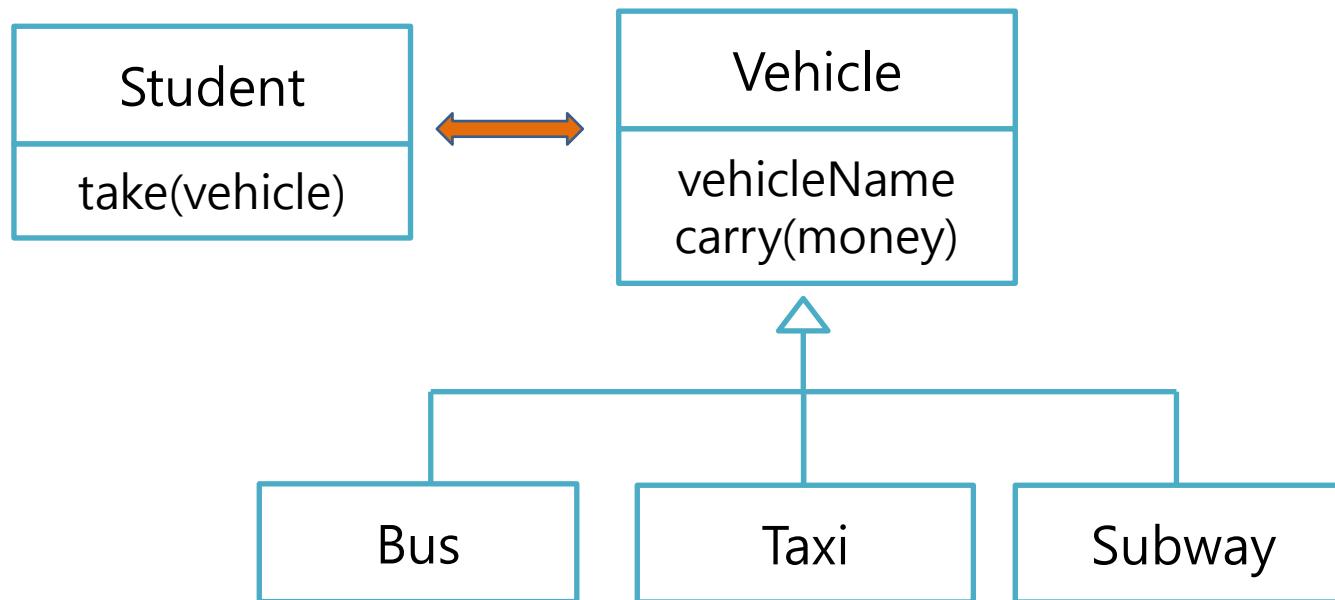
사람이 두 발로 걷습니다.  
독수리가 하늘을 납니다.  
호랑이가 네 발로 뛽니다.



# 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;  
    int money;  
    int passengerCount;  
  
    public Vehicle(String vehicleName) {  
        this.vehicleName = vehicleName;  
    }  
  
    public void carry(int money) {  
        this.money += money;  
        passengerCount++;  
    }  
  
    public void showInfo() {  
        System.out.printf("%s의 수입은 %,d원이고, 승객수는 %d명입니다.\n",  
                         vehicleName, money, passengerCount);  
    }  
}
```



# 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

## Student 클래스

```
public class Student {  
    String name;  
    int money;  
  
    public Student(String name, int money) {  
        this.name = name;  
        this.money = money;  
    }  
  
    //요금을 내고 교통 수단을 이용하기(다형성)  
    public void take(Vehicle vehicle, int fee) {  
        vehicle.carry(fee);  
        this.money -= fee;  
    }  
  
    public void showInfo() {  
        System.out.printf("%s의 남은 돈은 %,d원입니다.\n", name, money);  
    }  
}
```



# 교통수단 이용하기

## TakeTrans 클래스

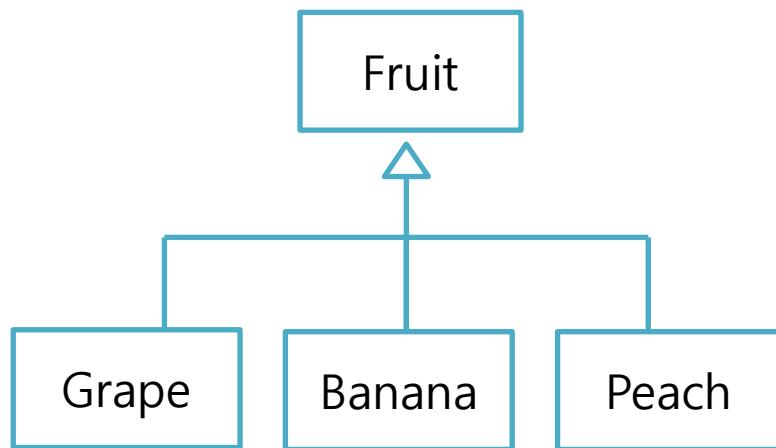
```
public class TakeTrans {  
  
    public static void main(String[] args) {  
        Student sohee = new Student("박소희", 10000);  
        Student daeho = new Student("이대호", 20000);  
  
        Bus bus100 = new Bus("bus100");  
        Taxi kakaoTaxi = new Taxi("카카오택시");  
  
        sohee.take(bus100, 1200);  
        daeho.take(kakaoTaxi, 3800);  
  
        sohee.showInfo();  
        bus100.showInfo();  
  
        daeho.showInfo();  
        kakaoTaxi.showInfo();  
    }  
}
```

박소희의 남은 돈은 8,800원입니다.  
bus100의 수입은 1,200원이고, 승객수는 1명입니다.  
이대호의 남은 돈은 16,200원입니다.  
카카오택시의 수입은 3,800원이고, 승객수는 1명입니다.



# 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

```
public class Fruit {  
    String name;      //과일 이름  
    String weight;    //무게  
    int price;        //가격  
  
    public Fruit() {} //생성자  
  
    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

```
public class Main {  
  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in);  
        System.out.println("=====");  
        System.out.println("1.포도 | 2.바나나 | 3.복숭아");  
        System.out.println("=====");  
        System.out.println("선택>");  
  
        int selNum = scan.nextInt();  
  
        Fruit fruit = null;  
        if(selNum == 1) {  
            fruit = new Grape();  
        }else if(selNum == 2) {  
            fruit = new Banana();  
        }else if(selNum == 3) {  
            fruit = new Peach();  
        }else {  
            System.out.println("지원하지 않는 기능입니다.");  
            return;  
        }  
        fruit.showInfo();  
        scan.close();  
    }  
}
```



# 다형성 예제

## NullPointerException 예외 처리

```
int selNum = scan.nextInt();

try {
    Fruit fruit = null;
    if(selNum == 1) {
        fruit = new Grape();
    }else if(selNum == 2) {
        fruit = new Banana();
    }else if(selNum == 3) {
        fruit = new Peach();
    }else {
        System.out.println("지원하지 않는 기능입니다.");
    }
    fruit.showInfo();
}catch(NullPointerException e) {
    System.out.println(e);
}
scan.close();
```



# 다형성 활용하기

## ● 일반 고객과 VIP 고객의 중간 등급 만들기

### 예제 시나리오

고객이 늘어 VIP 고객만큼 물건을 많이 구매하지는 않지만, 그래도 단골인 고객들에게 혜택을 주고 싶습니다.

GOLD 고객 등급을 하나 추가하고 혜택을 줍니다.

- 제품을 살 때는 항상 10%를 할인해 줍니다.
- 보너스 포인트를 2% 적립해 줍니다.
- 담당 전문 상담원을 없습니다.



# 다형성 활용하기

- 일반 고객과 VIP 고객의 중간 등급 만들기

```
public class GoldCustomer extends Customer{  
  
    double saleRatio;    //구매 할인율  
  
    public GoldCustomer(int customerID, String customerName) {  
        super(customerID, customerName);  
        customerGrade = "Gold";  
        bonusRatio = 0.02;  
        saleRatio = 0.1;  
    }  
  
    //Customer를 재정의한 메서드  
    @Override  
    public int calcPrice(int price) {  
        price -= (int) (price * saleRatio);  
        bonusPoint += price * bonusRatio;  
        return price;  
    }  
}
```



# 다형성 활용하기

- 배열을 활용하여 5명 구현하기

## 예제 시나리오

이 회사의 고객은 현재 5명입니다. 5명 중 VIP 1명, GOLD 2명, SILVER 2명입니다.

이 고객들이 10000원짜리 상품을 구매했을 때의 결과를 출력합니다.

```
package witharraylist;

import java.util.ArrayList;
public class CustomerTest {
    public static void main(String[] args) {
        ArrayList<Customer> customerList = new ArrayList<>();

        Customer customerLee = new Customer(10010, "이순신");
        Customer customerShin = new Customer(10020, "신사임당");
        Customer customerHong = new GoldCustomer(10030, "홍길동");
        Customer customerYoul = new GoldCustomer(10040, "이율곡");
        Customer customerKing = new VIPCustomer(10050, "세종대왕", 65536);
```



# ArrayList를 활용한 고객관리 프로그램 완성

## CustomerTest 클래스

```
customerList.add(customerLee);
customerList.add(customerShin);
customerList.add(customerHong);
customerList.add(customerYoul);
customerList.add(customerKing);

System.out.println("===== 구매 가격과 보너스 포인트 계산 =====");
int price = 10000;
for(Customer customer : customerList) {
    int cost = customer.calcPrice(price);
    System.out.printf("%s님이 %,d원 지불하셨습니다.\n", customer.getCustomerName(), cost);
}

System.out.println("===== 고객 정보 출력 =====");
for(Customer customer : customerList) {
    System.out.println(customer.showInfo());
}
```



# ArrayList를 활용한 고객관리 프로그램 완성

## CustomerTest 클래스 출력

```
===== 구매 가격과 보너스 포인트 계산 =====
```

이순신님이 10,000원 지불하셨습니다.

신사임당님이 10,000원 지불하셨습니다.

홍길동님이 10,000원 지불하셨습니다.

이율곡님이 10,000원 지불하셨습니다.

세종대왕님이 9,000원 지불하셨습니다.

```
===== 고객 정보 출력 =====
```

이순신님의 등급은 SILVER이고, 보너스 포인트는 100입니다.

신사임당님의 등급은 SILVER이고, 보너스 포인트는 100입니다.

홍길동님의 등급은 Gold이고, 보너스 포인트는 200입니다.

이율곡님의 등급은 Gold이고, 보너스 포인트는 200입니다.

세종대왕님의 등급은 VIP이고, 보너스 포인트는 450입니다. 담당 상담원 ID는 65536입니다.



# 다운 캐스팅

## 하위 클래스로 형 변환 -> 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 h = (Human)animal;  
    h.readBook();  
}
```



# 다운 캐스팅

하위 클래스로 형 변환 -> 다운 캐스팅

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

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

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

class Eagle extends Animal{
    public void move() {
        System.out.println("독수리가 하늘을 납니다.");
    }

    public void flying() {
        System.out.println("독수리가 멀리 날아갑니다.");
    }
}
```



# 다운 캐스팅

하위 클래스로 형 변환 -> 다운 캐스팅

```
public class AnimalTest {  
  
    public void moveAnimal(Animal animal) {  
        animal.move();  
    }  
  
    public static void main(String[] args) {  
        AnimalTest aTest = new AnimalTest();  
        Human human = new Human();  
        aTest.moveAnimal(human);  
  
        Animal animal1 = new Human();  
        //Animal에 readBook() 없으므로 보이지 않음  
        //다운캐스팅이 필요함  
  
        if(animal1 instanceof Human) {  
            human = (Human)animal1;  
            human.readBook();  
        }  
    }  
}
```



# 다운 캐스팅

## ArrayList로 구현

```
public class AnimalTest2 {  
  
    public static void main(String[] args) {  
        ArrayList<Animal> animalList = new ArrayList<>();  
  
        Animal human = new Human();  
        Animal eagle = new Eagle();  
        Animal tiger = new Tiger();  
  
        animalList.add(human);  
        animalList.add(eagle);  
        animalList.add(tiger);  
  
        // 출력  
        for(Animal animal : animalList)  
            animal.move();  
    }  
}
```



# 다운 캐스팅

## ArrayList로 구현

```
System.out.println("== 원래 형으로 다운 캐스팅 ==");
for(int i = 0; i < animalList.size(); i++) { //모든 리스트 요소를 하나씩 돌면서
    Animal animal = animalList.get(i);           //animal형으로 가져옴
    if(animal instanceof Human) { //animal이 Human 클래스의 인스턴스라면
        Human h = (Human)animal; //animal을 Human으로 형변환
        h.readBook();
    }else if(animal instanceof Eagle) {
        Eagle e = (Eagle) animal;
        e.flying();
    }else if(animal instanceof Tiger) {
        Tiger t = (Tiger)animal;
        t.hunting();
    }else {
        System.out.println("지원되지 않는 형입니다.");
    }
}
```



# 다운 캐스팅

## ArrayList로 구현 – 함수로 모듈화

```
public class AnimalTest3 {  
  
    ArrayList<Animal> animalList = new ArrayList<>();  
  
    public void addAnimal() {  
        //인스턴스 생성 및 저장  
        Animal human = new Human();  
        Animal eagle = new Eagle();  
        Animal tiger = new Tiger();  
  
        animalList.add(human);  
        animalList.add(eagle);  
        animalList.add(tiger);  
  
        //출력  
        for(Animal animal : animalList)  
            animal.move();  
    }  
}
```



# 다운 캐스팅

```
public void testCasting() {  
    for(int i = 0; i < animalList.size(); i++) { //모든 리스트 요소를 하나씩 돌면서  
        Animal animal = animalList.get(i); //animal형으로 가져옴  
        if(animal instanceof Human) { //animal이 Human 클래스의 인스턴스라면  
            Human h = (Human)animal;  
            h.readBook();  
        }else if(animal instanceof Eagle) {  
            Eagle e = (Eagle) animal;  
            e.flying();  
        }else if(animal instanceof Tiger) {  
            Tiger t = (Tiger)animal;  
            t.hunting();  
        }else {  
            System.out.println("지원되지 않는 형입니다.");  
        }  
    }  
  
    public static void main(String[] args) {  
        AnimalList aniList = new AnimalList();  
        aniList.addAnimal();  
  
        System.out.println("== 원래 형으로 다운 캐스팅 ==");  
        aniList.testCasting();  
    }  
}
```



# 다운 캐스팅

Array로 구현  
배열로 구현

```
public class AnimalArray {
    Animal[] animal = new Animal[3];

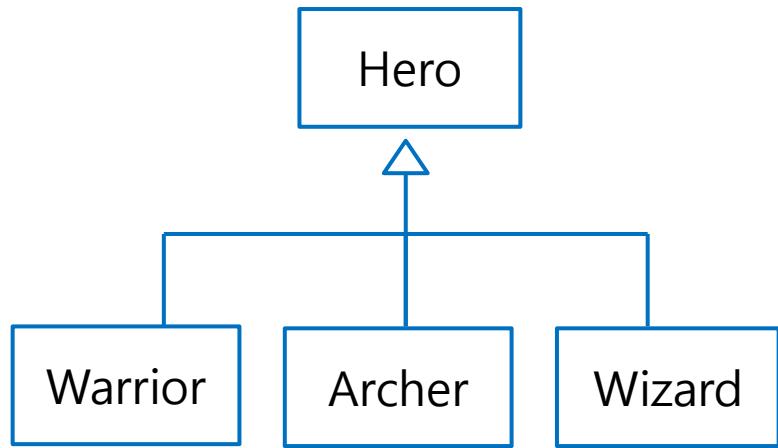
    public void addAnimal() {
        animal[0] = new Human();
        animal[1] = new Tiger();
        animal[2] = new Eagle();
    }

    public void testCasting() {
        for(int i=0; i<animal.length; i++) {
            if(animal[i] instanceof Human) {
                Human human = (Human)animal[i];
                human.readBook();
            }
            else if(animal[i] instanceof Tiger) {
                Tiger tiger = (Tiger)animal[i];
                tiger.hunting();
            }
            else if(animal[i] instanceof Eagle) {
                Eagle eagle = (Eagle)animal[i];
                eagle.flying();
            }
            animal[i].move();
            System.out.println("=====");
        }
    }
}
```



# 다운 캐스팅 활용 예제

## ■ 영웅(Hero) 캐릭터 만들기



```
public class Hero {  
    String name;  
  
    public Hero(String name) {  
        this.name = name;  
    }  
  
    public void attack() {  
        System.out.println("기본 : 주먹 지르기");  
    }  
  
    public void showInfo() {  
        System.out.println("직업 : " + name);  
    }  
}
```



# 영웅 캐릭터 만들기

## ■ 영웅(Hero) 캐릭터 만들기

```
public class Warrior extends Hero{  
    public Warrior(String name) {  
        super(name);  
    }  
  
    public void groundCutting() {  
        System.out.println("특기 : 대지 가르기");  
    }  
}
```

```
public class Archer extends Hero {  
    public Archer(String name) {  
        super(name);  
    }  
  
    public void fireArrow() {  
        System.out.println("특기 : 불화살 쏘기");  
    }  
}
```

```
public class Wizard extends Hero{  
    public Wizard(String name) {  
        super(name);  
    }  
  
    public void freezing() {  
        System.out.println("특기 : 얼음 얼리기");  
    }  
}
```



# 영웅 캐릭터 만들기

- ArrayList로 구현

```
public class HeroTest {  
    ArrayList<Hero> heroList = new ArrayList<>();  
  
    public void addHero() {  
        heroList.add(new Warrior("전사"));  
        heroList.add(new Archer("궁수"));  
        heroList.add(new Wizard("마법사"));  
    }  
  
    public void testCasting() {  
        for(Hero hero : heroList) {  
            if(hero instanceof Warrior) {  
                Warrior warrior = (Warrior)hero;  
                warrior.showInfo();  
                warrior.groundCutting();  
            }  
            else if(hero instanceof Archer) {  
                Archer archer = (Archer)hero;  
                archer.showInfo();  
                archer.fireArrow();  
            }  
            else if(hero instanceof Wizard) {  
                Wizard wizard = (Wizard)hero;  
                wizard.showInfo();  
                wizard.freezing();  
            }  
            hero.attack();  
            System.out.println("=====");  
        }  
    }  
}
```



# 영웅 캐릭터 만들기

- ArrayList로 구현

```
public static void main(String[] args) {  
    HeroTest hTest = new HeroTest();  
  
    hTest.addHero();           // 객체 생성  
  
    hTest.testCasting();     // 다형성 - 다운캐스팅  
}
```

직업 : 전사
특기 : 대지 가르기
기본 : 주먹 지르기
=====
직업 : 궁수
특기 : 불화살 쏘기
기본 : 주먹 지르기
=====
직업 : 마법사
특기 : 얼음 얼리기
기본 : 주먹 지르기
=====



# 영웅 캐릭터 만들기

- Array로 구현

```
public class HeroTest2 {  
    Hero[] hero = new Hero[3];  
  
    public void addHero() {  
        hero[0] = new Warrior("전사");  
        hero[1] = new Archer("궁수");  
        hero[2] = new Wizard("마법사");  
    }  
  
    public void testCasting() {  
        for(int i=0; i<hero.length; i++){  
            if(hero[i] instanceof Warrior) {  
                Warrior warrior = (Warrior)hero[i];  
                warrior.showInfo();  
                warrior.groundCutting();  
            }  
            else if(hero[i] instanceof Archer) {  
                Archer archer = (Archer)hero[i];  
                archer.showInfo();  
                archer.fireArrow();  
            }  
            else if(hero[i] instanceof Wizard) {  
                Wizard wizard = (Wizard)hero[i];  
                wizard.showInfo();  
                wizard.freezing();  
            }  
            hero[i].attack();  
            System.out.println("=====");  
        }  
    }  
}
```

