02. 자바와 객체지향

2018-02

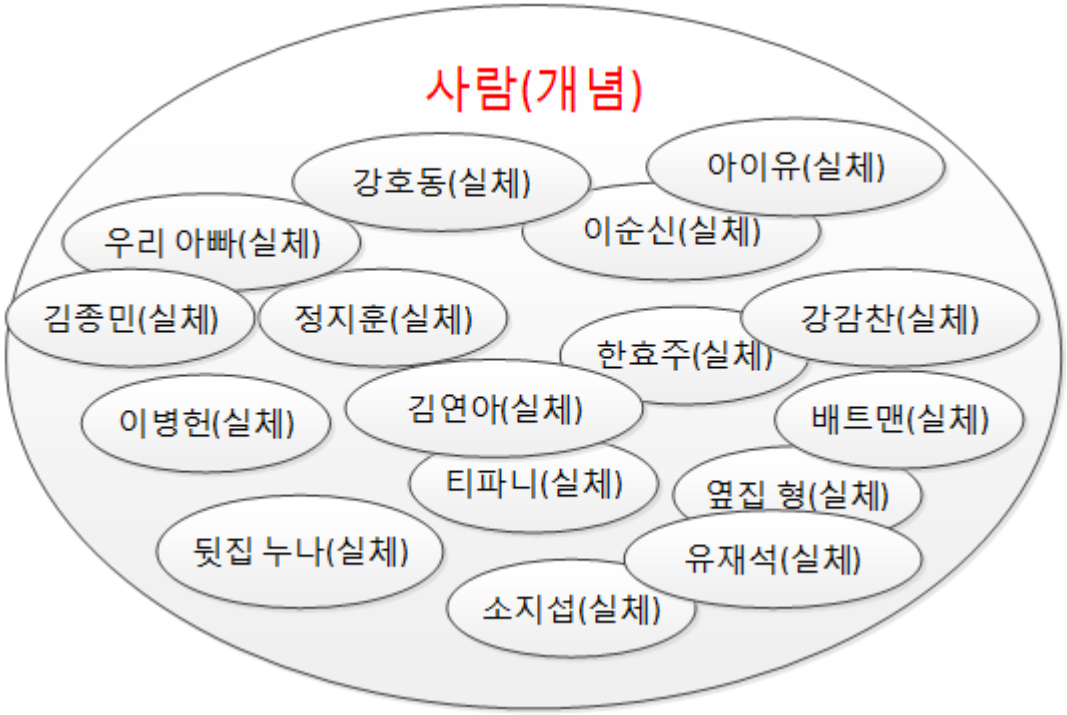
고급객체지향프로그래밍

contents

- 클래스와 객체
- 추상화
- 상속
- 다형성
- 캡슐화

클래스와 객체

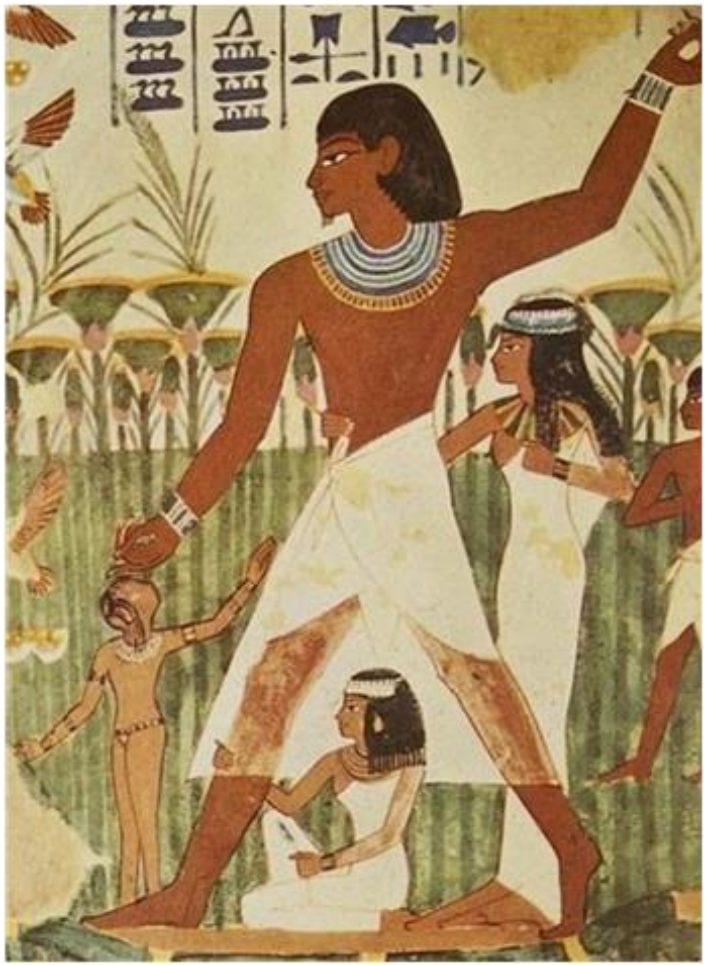
- object: 유일무이(unique)한 사물
- class: 같은 특성을 지닌 여러 개체를 총칭하는 집합의 개념



객체지향의 4대 특성

- 추상화
- 상속
- 다형성
- 캡슐화

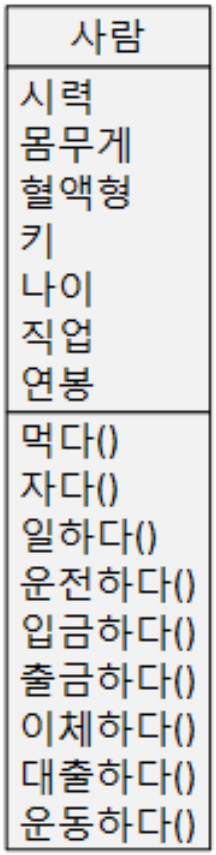
추상화 (1)



추상화 (2)

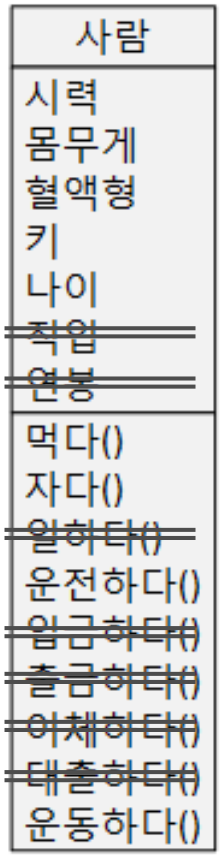
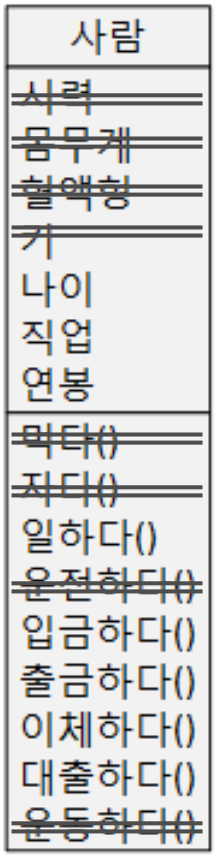
- 추상화 => 모델링
  - 구체적인 것을 분해해서 관찰자가 관심 있는 특성만 가지고 재조합 하는 것
  - 구체적인 것을 분해해서 관심 영역(애플리케이션 경계, Application Boundary) 에 있는 특성만 가지고 재조합 하는 것 (=> 모델링)

추상(抽象) [명사][심리] 여러 가지 사물이나 개념에서 공통되는 특성이나 속성 따위를 추출하여 파악하는 작용.



추상화 (3)

- 애플리케이션 경계에 따른 클래스 모델링의 차이

애플리케이션 경계	병원 애플리케이션	은행 애플리케이션
사람이란 클래스 모델링	사람은 환자다 	사람은 고객이다 

자바의 추상화

추상화 = 모델링 = 자바의 class 키워드

- 자바의 클래스와 객체의 관계

클래스	**객체_참조_변수**	**=**	**new**	**클래스**	**( )**
객체_참조_변수의 자료형(type)	생성된 객체를 참조할 수 있는 변수	할당문	새로운	만들고자 하는 객체의 분류	메서드

클래스의 인스턴스, 즉 객체를 생성하기 위해 객체 생성자를 호출

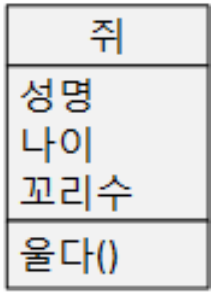
새로운 객체를 하나 생성해 그 객체의 주소값을 객체 참조 변수에 할당

추상화와 메모리 (1)

- 예: 애니메이션의 쥐 캐릭터 관리 프로그램

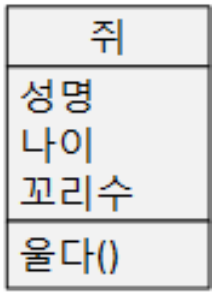
① 객체들의 공통 특성 추출

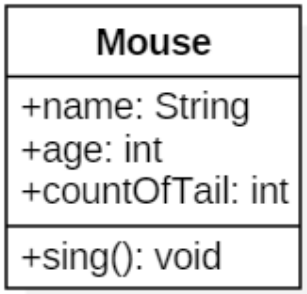
객체명	미키마우스	제리
속성들	성명: 미키마우스 국적: 미국 나이: 87 종교: 무교 신장: 70cm 체중: 11.5kg 애완동물: 플루토 여자친구: 미니마우스 꼬리: 1개 …	성명: 제리 국적: 미국 나이: 75 종교: 기독교 앙숙: 톰 여자친구: toots 꼬리: 1개 …
행위들	달리다( ) 먹다( ) 휘파람불다( ) 데이트하다( ) 울다( ) …	달리다( ) 먹다( ) 장난치다( ) …



추상화와 메모리 (2)

② 쥐 클래스의 논리적 / 물리적 설계





③ 쥐 클래스의 자바 프로그래밍

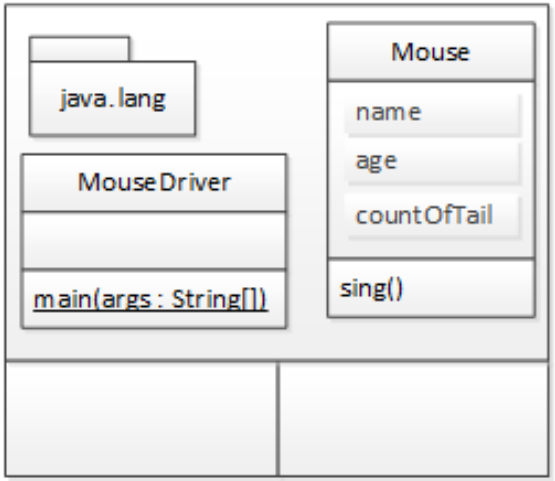
```java
public class Mouse {
    public String name;
    public int age;
    public int countOfTail;

    public void sing() {
        System.out.println(name + " 찍찍!!!");
    }
}
```

```java
public class MouseDriver {
    public static void main(String[] args) {
        Mouse mickey = new Mouse();
        mickey.name = "미키";
        mickey.age = 85;
        mickey.countOfTail = 1;
        mickey.sing();

        mickey = null;

        Mouse jerry = new Mouse();
        jerry.name = "제리";
        jerry.age = 73;
        jerry.countOfTail = 1;
        jerry.sing();
    }
}
```

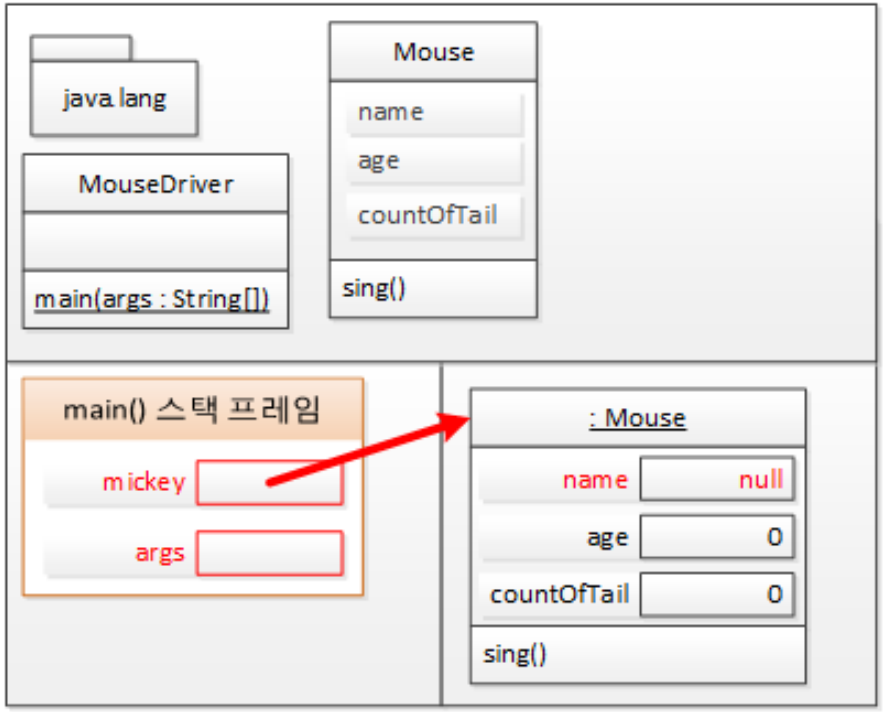
추상화와 메모리 (3)

④ 쥐 클래스/객체와 메모리 – stack 영역, heap 영역

ⓐ main( ) 메서드 실행 직전



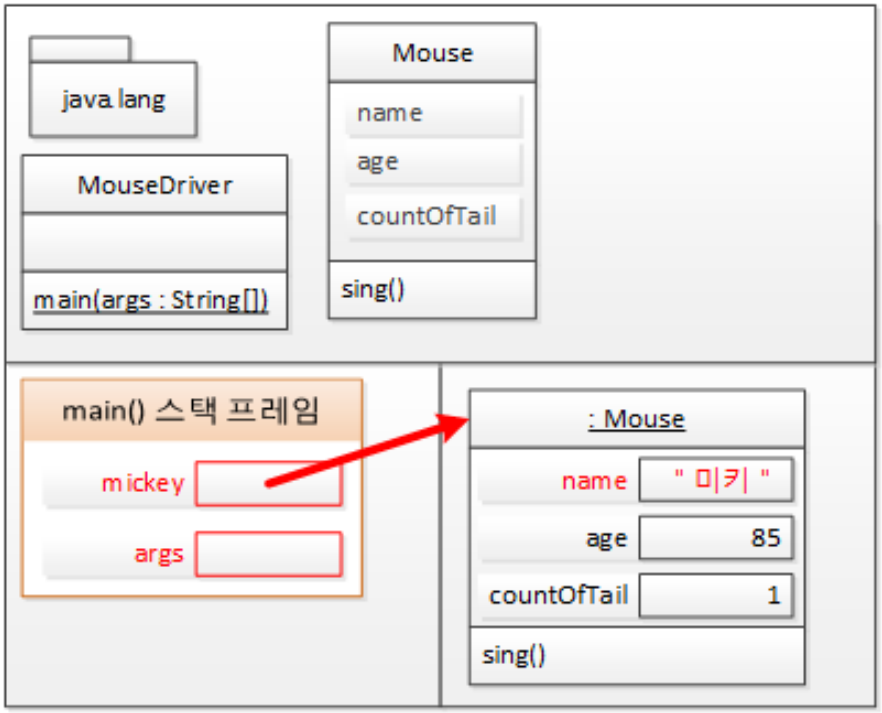
ⓑ line 5: 실행 후



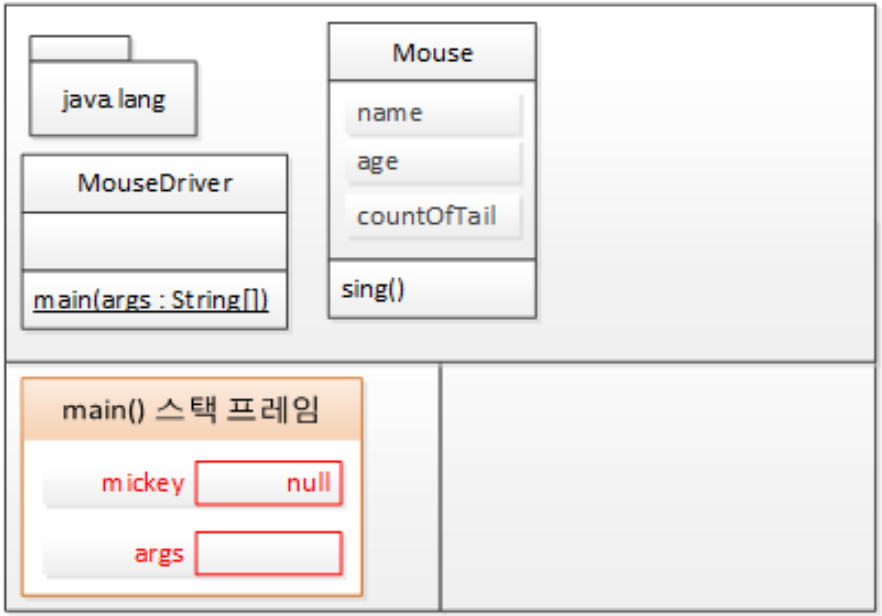
추상화와 메모리 (4)

④ 쥐 클래스/객체와 메모리 – stack 영역, heap 영역

ⓒ line 9: 실행 후



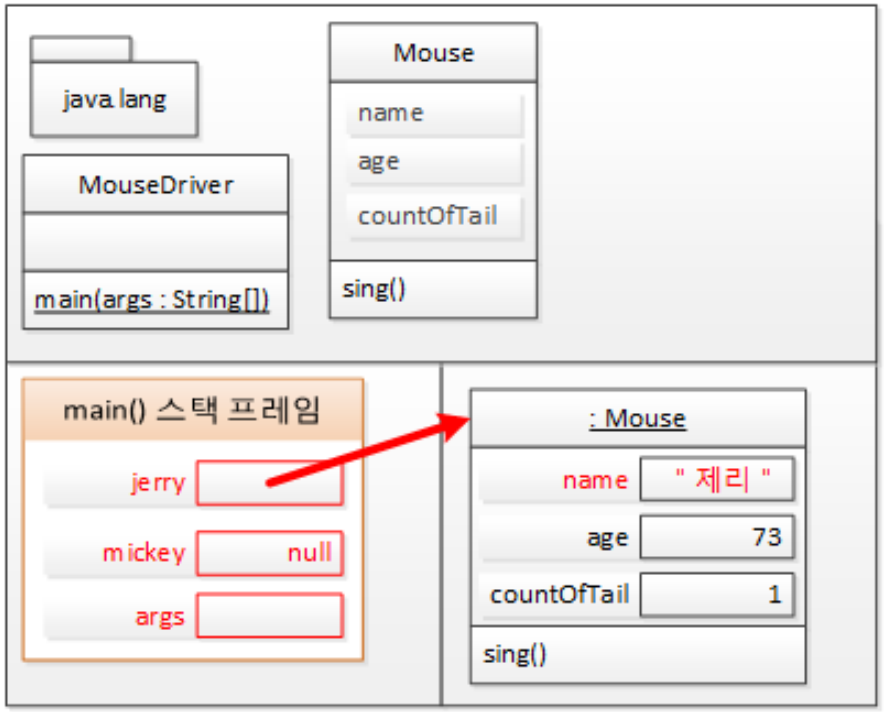
ⓓ line 13: 실행 후



추상화와 메모리 (5)

④ 쥐 클래스/객체와 메모리 – stack 영역, heap 영역

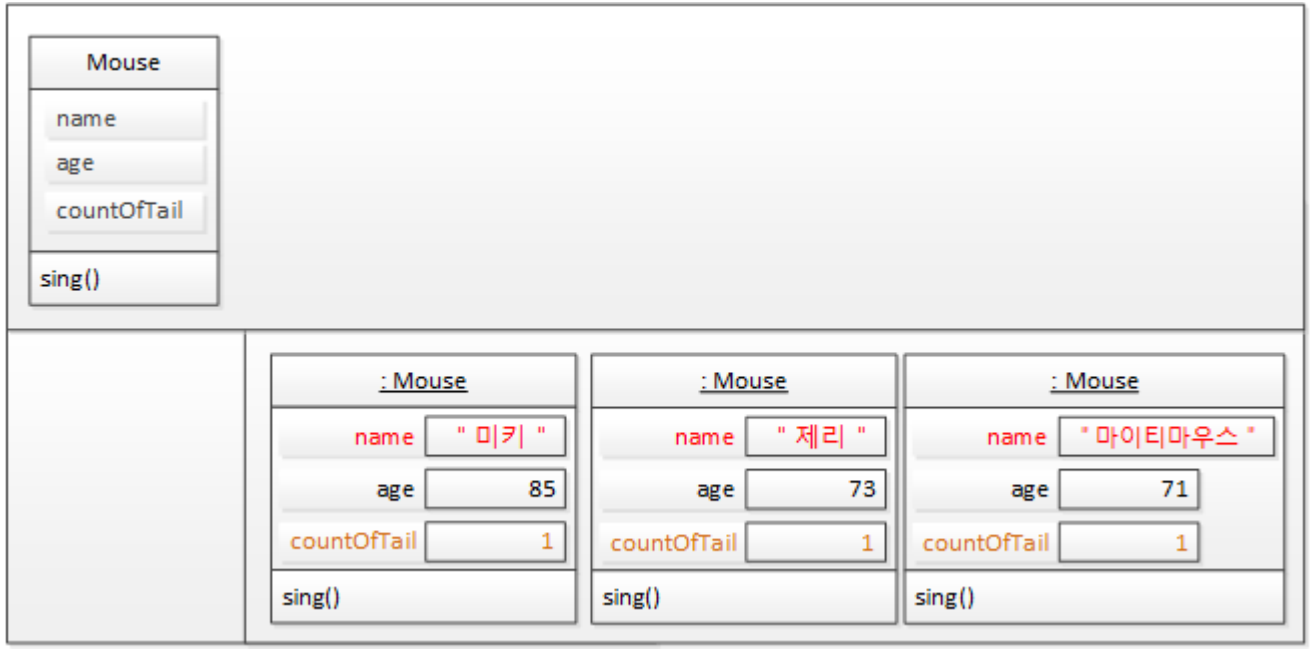
ⓔ line 19: 실행 후



추상화와 메모리 (6)

⑤ 쥐 클래스/객체와 메모리 - static영역, heap 영역

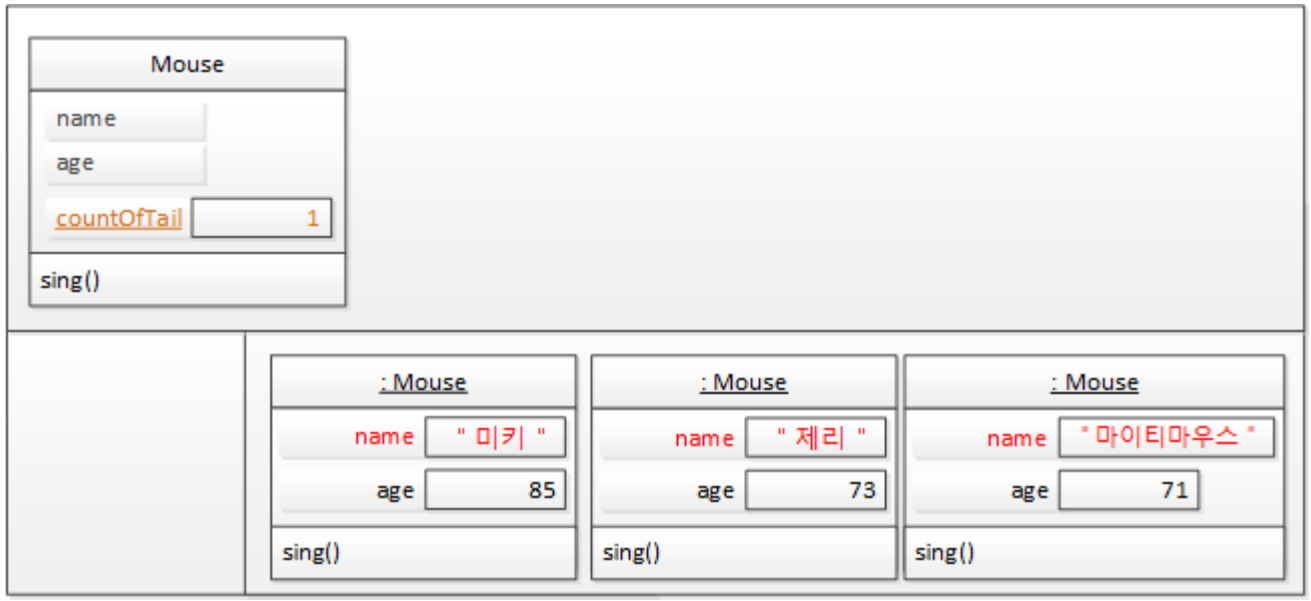
ⓐ Mouse 클래스의 모든 객체들이 같은 countOfTail값을 가짐



추상화와 메모리 (7)

⑤ 쥐 클래스/객체와 메모리 - static영역, heap 영역

ⓑ 공통된 값을 갖는 속성을 클래스 레벨로 옮김

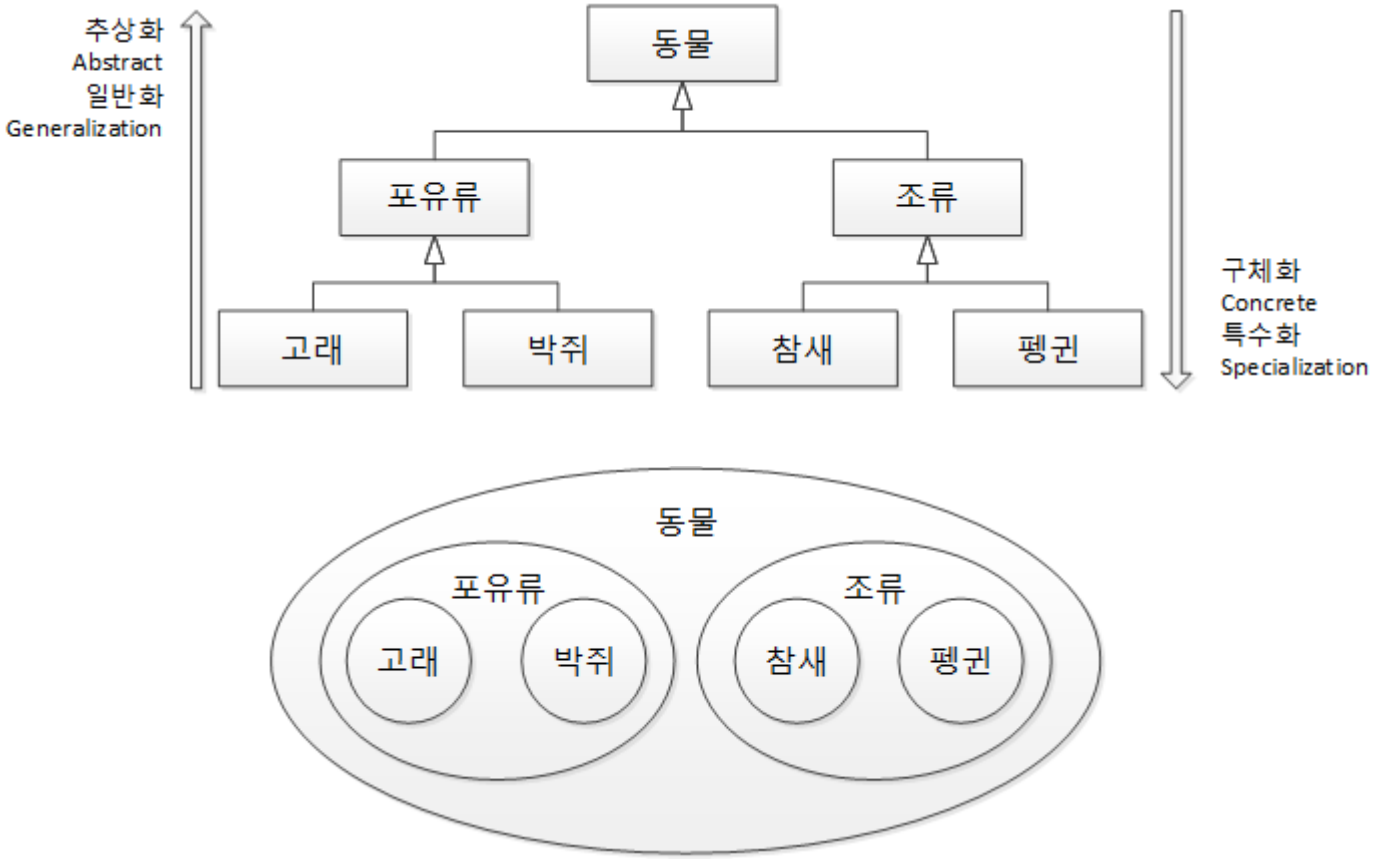


추상화와 메모리 (8)

```java
public class Mouse {
    public String name;
    public int age;
    public static int countOfTail = 1;
    // public final static int countOfTail = 1;
    public void sing() {
        System.out.println(name + " 찍찍!!!");
    }
}
```

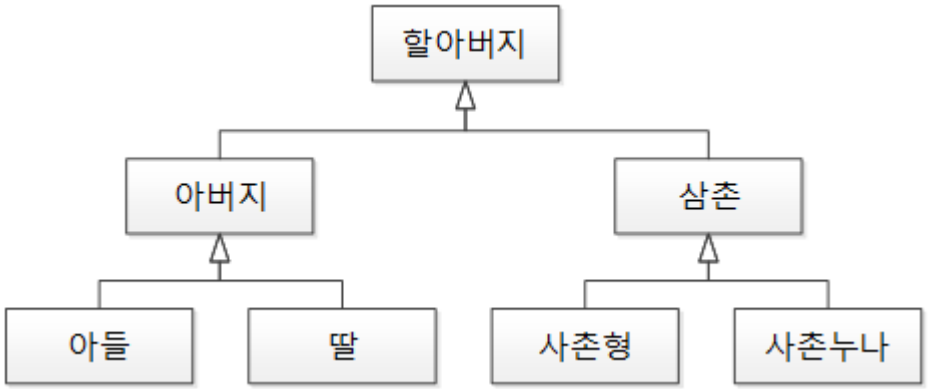
```java
public class MouseDriver {
    public static void main(String[] args) {
        // 클래스명.countOfTail
        Mouse.countOfTail = 1;
        Mouse mickey = new Mouse();
        Mouse jerry = new Mouse();
        Mouse mightyMouse = new Mouse();

        // 객체명.countOfTail
        System.out.println(mickey.countOfTail);
        System.out.println(jerry.countOfTail);
        System.out.println(mightyMouse.countOfTail);
        // 클래스명.countOfTail
        System.out.println(Mouse.countOfTail);
    }
}
```

상속 (1)


추상화
Abstract
일반화
Generalization
동물
포유류
조류
고래
박쥐
참새
펭귄
구체화
Concrete
특수화
Specialization
동물
포유류
조류
고래
박쥐
참새
펭귄

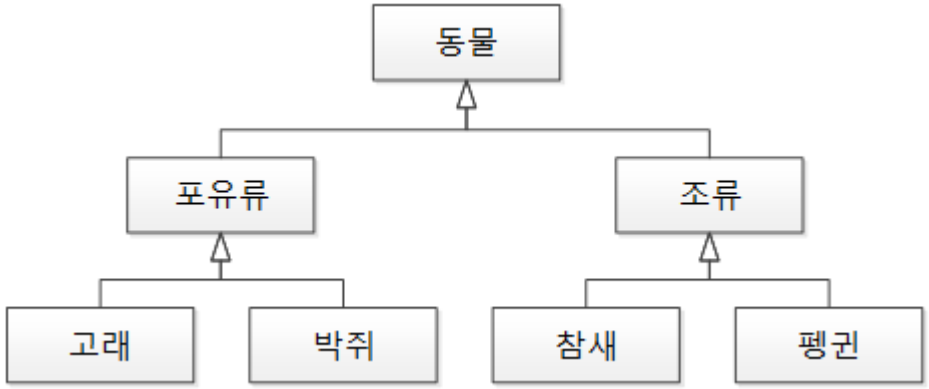
상속 (2)

하위 클래스는 상위 클래스이다.



- 아버지는 할아버지이다?
- 아들은 아버지이다?
- 딸은 아버지이다?

```
아버지 영희아빠 = new 딸( );
```



- 포유류는 동물이다.
- 고래는 포유류이다.
- 고래는 동물이다.

```
동물 뽀로로 = new 펭귄( );
```

상속 (3)

하위 클래스 is a kind of 상위 클래스

- 객체 지향의 상속은 상위 클래스의 특성을 재사용하는 것
- 객체 지향의 상속은 상위 클래스의 특성을 확장하는 것
- 객체 지향의 상속은 is a kind of 관계를 만족해야 한다.

- 펭귄 is a kind of 조류 -> 펭귄은 조류의 한 분류이다.
- 펭귄 is a kind of 동물-> 펭귄은 동물의 한 분류이다.
- 고래 is a kind of 조류 -> 고래는 동물의 한 분류이다.
- 조류 is a kind of 조류 -> 조류는 동물의 한 분류이다.

상속의 예 (1)

```
3  public class 동물 {
4      String myClass;
5
6      동물() {
7          myClass = "동물";
8      }
9
10     void showMe() {
11         System.out.println(myClass);
12     }
13 }
```

```
3  public class 포유류 extends 동물 {
4      포유류() {
5          myClass = "포유류";
6      }
7  }
```

```
3  public class 조류 extends 동물 {
4      조류() {
5          myClass = "조류";
6      }
7  }
```

```
3  public class 고래 extends 포유류 {
4      고래() {
5          myClass = "고래";
6      }
7  }
```

```
3  public class 박쥐 extends 포유류 {
4      박쥐() {
5          myClass = "박쥐";
6      }
7  }
```

```
3  public class 참새 extends 조류 {
4      참새() {
5          myClass = "참새";
6      }
7  }
```

```
3  public class 펭귄 extends 조류 {
4      펭귄() {
5          myClass = "펭귄";
6      }
7  }
```

상속의 예 (2)

```java
public class Driver01 {
    public static void main(String[] args) {
        동물 animal = new 동물();
        포유류 mammalia = new 포유류();
        조류 bird = new 조류();
        고래 whale = new 고래();
        박쥐 bat = new 박쥐();
        참새 sparrow = new 참새();
        펭귄 penguin = new 펭귄();

        animal.showMe();
        mammalia.showMe();
        bird.showMe();
        whale.showMe();
        bat.showMe();
        sparrow.showMe();
        penguin.showMe();
    }
}
```

```java
public class Driver02 {
    public static void main(String[] args) {
        동물 animal = new 동물();
        동물 mammalia = new 포유류();
        동물 bird = new 조류();
        동물 whale = new 고래();
        동물 bat = new 박쥐();
        동물 sparrow = new 참새();
        동물 penguin = new 펭귄();

        animal.showMe();
        mammalia.showMe();
        bird.showMe();
        whale.showMe();
        bat.showMe();
        sparrow.showMe();
        penguin.showMe();
    }
}
```

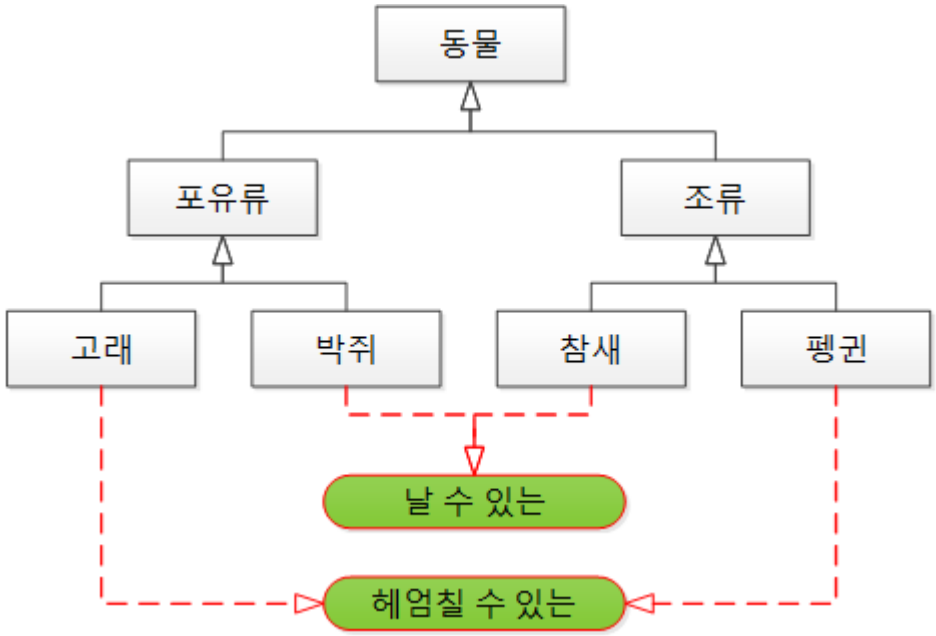
상속의 예 (3)

```
 3 public class Driver03 {
 4     public static void main(String[] args) {
 5         동물[] animals = new 동물[7];
 6 
 7         animals[0] = new 동물();
 8         animals[1] = new 포유류();
 9         animals[2] = new 조류();
10         animals[3] = new 고래();
11         animals[4] = new 박쥐();
12         animals[5] = new 참새();
13         animals[6] = new 펭귄();
14 
15         for (int index = 0; index < animals.length; index++) {
16             animals[index].showMe();
17         }
18     }
19 }
```

상속과 인터페이스

- 상속관계: 하위 클래스 is a kind of 상위 클래스
- 해석: 하위 클래스는 상위 클래스의 한 분류이다.
- 예제: 고래는 동물의 한 분류이다.

- 인터페이스: 구현 클래스 is able to 인터페이스
- 해석: 구현 클래스는 인터페이스 할 수 있다.
- 예제: 고래는 헤엄칠 수 있다.



인터페이스 사용 예 (1)

```
3  public class 동물 {
4      String myClass;
5
6      동물() {
7          myClass = "동물";
8      }
9
10     void showMe() {
11         System.out.println(myClass);
12     }
13 }
```

```
3  public interface 날수있는 {
4      void fly();
5  }
```

```
3  public interface 헤엄칠수있는 {
4      void swim();
5  }
```

```
3  public class 포유류 extends 동물 {
4      포유류() {
5          myClass = "포유류";
6      }
7  }
```

```
3  public class 조류 extends 동물 {
4      조류() {
5          myClass = "조류";
6      }
7  }
```

인터페이스 사용 예 (2)

```
3  public class 고래 extends 포유류 implements 헤엄칠수있는 {
4      고래() {
5          myClass = "고래";
6
7      }
8
9      @Override
10     public void swim() {
11         System.out.println(myClass + " 수영 중. 어프!!! 어프!!!");
12     }
13 }
```

```
3  public class 펭귄 extends 조류 implements 헤엄칠수있는 {
4      펭귄() {
5          myClass = "펭귄";
6
7      }
8
9      @Override
10     public void swim() {
11         System.out.println(myClass + " 수영 중. 푸악!!! 푸악!!!");
12     }
13 }
```

인터페이스 사용 예 (3)

```
3  public class 박쥐 extends 포유류 implements 날수있는 {
4      박쥐() {
5          myClass = "박쥐";
6
7      }
8
9      @Override
10     public void fly() {
11         System.out.println(myClass + " 날고 있삼.. 슈웅!!! 슈웅!!!");
12     }
13 }
```

```
3  public class 참새 extends 조류 implements 날수있는 {
4      참새() {
5          myClass = "참새";
6      }
7
8      @Override
9      public void fly() {
10         System.out.println(myClass + " 날고 있삼.. 허우적!!! 허우적!!!");
11     }
12 }
```

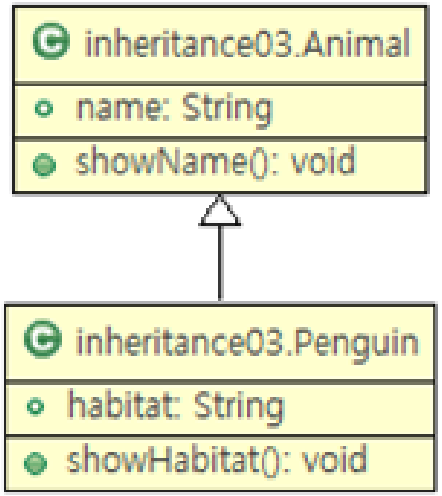
인터페이스 사용 예 (4)

```
3  public class Driver {
4      public static void main(String[] args) {
5          날수있는 날라리1 = new 박쥐();
6          날라리1.fly();
7
8          날수있는 날라리2 = new 참새();
9          날라리2.fly();
10
11         헤엄칠수있는[] 맥주병들 = new 헤엄칠수있는[2];
12
13         맥주병들[0] = new 고래();
14         맥주병들[1] = new 펭귄();
15
16         for (헤엄칠수있는 맥주병 : 맥주병들) {
17             맥주병.swim();
18         }
19     }
20 }
```

상속과 메모리 (1)

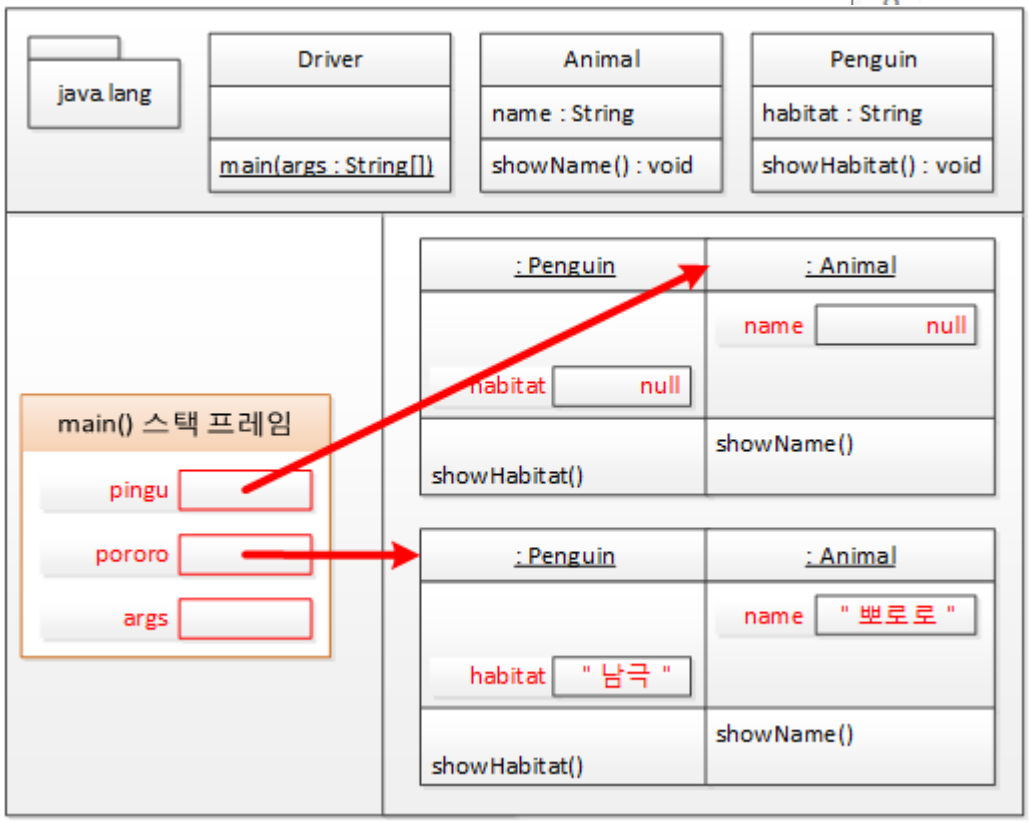
```java
public class Animal {
    public String name;

    public void showName() {
        System.out.printf("안녕 나는 %s야. 반가워\n", name);
    }
}
```

```java
public class Penguin extends Animal {
    public String habitat;

    public void showHabitat() {
        System.out.printf("%s는 %s에 살아\n", name, habitat);
    }
}
```



상속과 메모리 (2)

```
3  public class Driver {
4      public static void main(String[] args) {
5          Penguin pororo = new Penguin();
6
7          pororo.name = "뽀로로";
           pororo.habitat = "남극";

           pororo.showName();
           pororo.showHabitat();

           Animal pingu = new Penguin();

           pingu.name = "핑구";
           // pingu.habitat = "EBS";

           pingu.showName();
           // pingu.showHabitat();

           // Penguin happyfeet = new Animal();
```



다형성

- 오버라이딩, 오버로딩

```java
public class Animal {
    public String name;

    public void showName() {
        System.out.printf("안녕 나는 %s야. 반가워\n", name);
    }
}
```

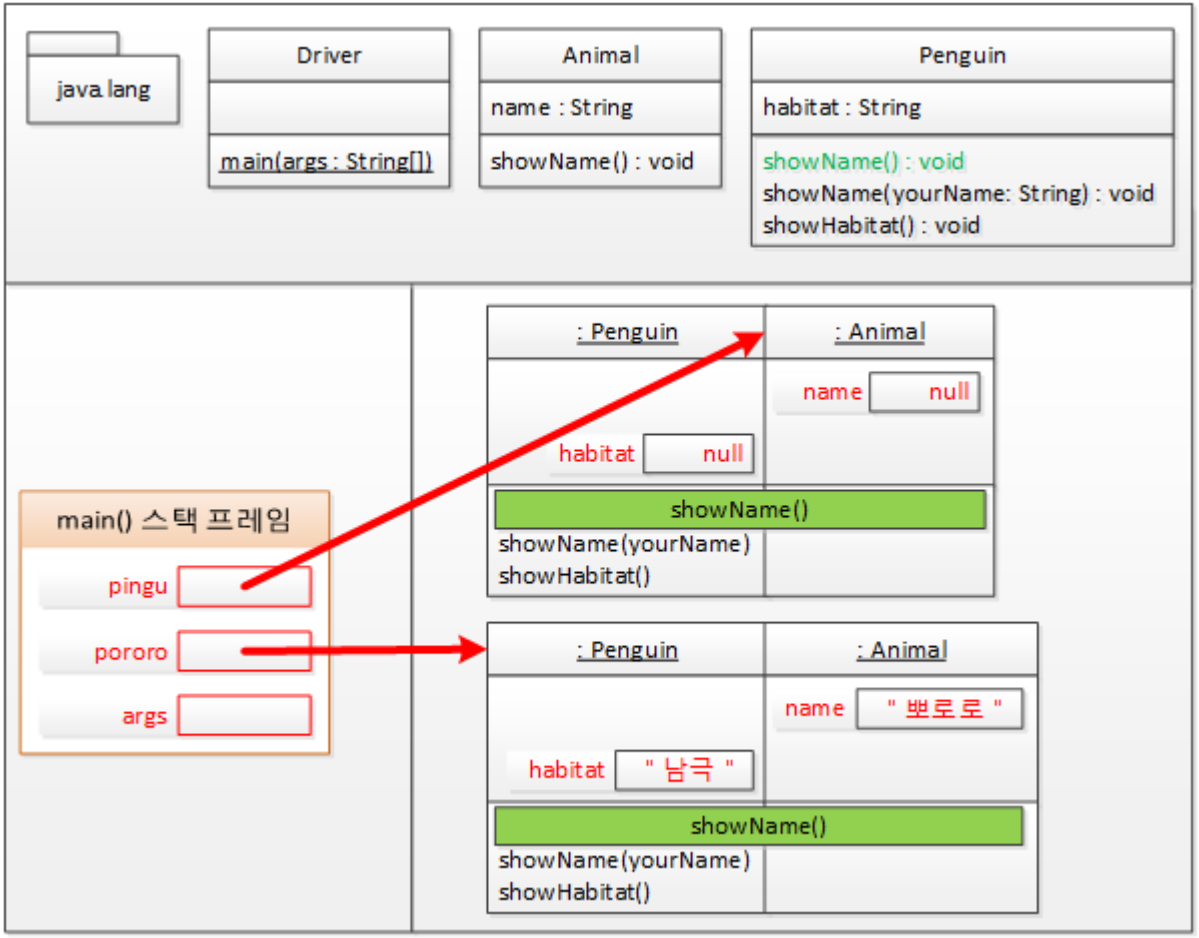
```java
public class Penguin extends Animal {
    public String habitat;

    public void showHabitat() {
        System.out.printf("%s는 %s에 살아\n", name, habitat);
    }

    //오버라이딩 - 재정의: 상위클래스의 메서드와 같은 메서드 이름, 같은 인자 리스트
    public void showName() {
        System.out.println("어머 내 이름은 알아서 뭐하게요?");
    }

    // 오버로딩 - 중복정의: 같은 메서드 이름, 다른 인자 리스트
    public void showName(String yourName) {
        System.out.printf("%s 안녕, 나는 %s라고 해\n", yourName, name);
    }
}
```

다형성과 메모리 (1)

```
3  public class Driver {
4      public static void main(String[] args) {
5          Penguin pororo = new Penguin();
6
7          pororo.name = "뽀로로";
8          pororo.habitat = "남극";
9
10         pororo.showName();
11         pororo.showName("뽀로롱");
12         pororo.showHabitat();
13
14         Animal pingu = new Penguin();
15
16         pingu.name = "핑구";
17         pingu.showName();
18     }
19 }
```

다형성과 메모리 (2)

```
Penguin pororo = new Penguin();

pororo.name = "뽀로로";
pororo.habitat = "남극";

pororo.showName();
pororo.showName("뽀로롱");
pororo.showHabitat();

Animal pingu = new Penguin();

pingu.name = "핑구";
pingu.showName();
```



캡슐화 (1)

- 객체 멤버의 접근 제어자
  - 상속을 받지 않았다면 객체 멤버는 객체를 생성한 후 객체 참조 변수를 이용해 접근해야 한다.
  - 정적 멤버는 클래스명.정적멤버 형식으로 접근하는 것을 권장한다.

- public 정적 멤버의 접근 방법

ClassA		ClassA.pubSt	pubSt	this.pubSt
		○	○	○
같은 패키지	상속한 경우	○	○	○
	상속하지 않은 경우	○	X	X
다른 패키지	상속한 경우	○	○	○
	상속하지 않은 경우	○	X	X

캡슐화 (2)

- 정적 멤버와 메모리 접근