

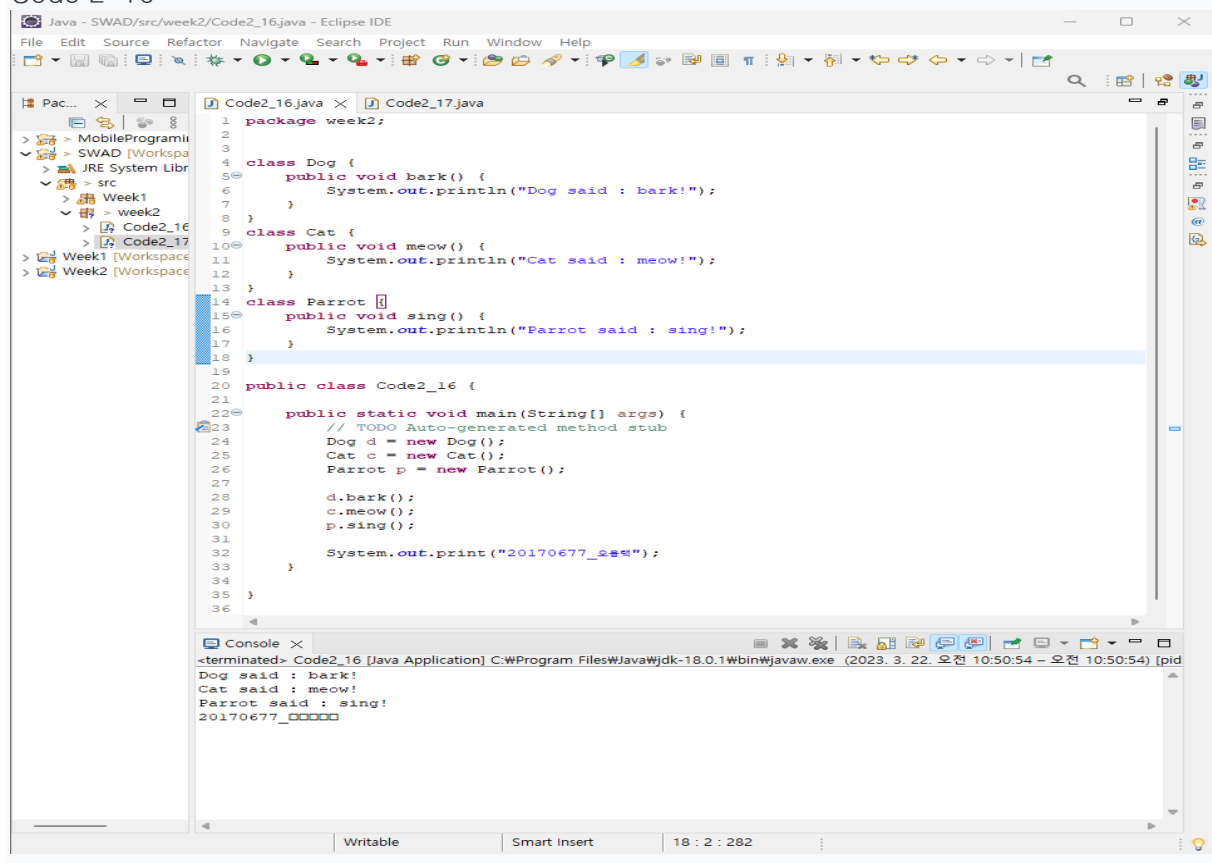
# 소프트웨어 분석 설계

## 3주차

Team 10

[1] 코드 2-16 과 코드 2-17 을 IDE 에서 작성하여 각각 완성하여 실행시켜보시오.  
(울음소리는 Systme.out.println()을 사용하여 콘솔로 문자열을 출력하는 것으로  
대신한다. ) 코드 2-16 과 코드 2-17 의 차이를 매우 구체적으로 이해한대로 잘 설명하시오.

Code 2-16



The screenshot shows the Eclipse IDE with a project named 'SWAD' and a package named 'week2'. The main editor displays the source code for 'Code2\_16.java'. The code defines three classes: 'Dog', 'Cat', and 'Parrot', each with a specific method (bark, meow, sing) that prints a message to the console. A 'main' method in 'Code2\_16' class instantiates one object of each class and calls their respective methods. The console at the bottom shows the output of these methods: 'Dog said : bark!', 'Cat said : meow!', 'Parrot said : sing!', and a final print statement '20170677\_오름'.

```
1 package week2;
2
3
4 class Dog {
5     public void bark() {
6         System.out.println("Dog said : bark!");
7     }
8 }
9 class Cat {
10    public void meow() {
11        System.out.println("Cat said : meow!");
12    }
13 }
14 class Parrot {
15    public void sing() {
16        System.out.println("Parrot said : sing!");
17    }
18 }
19
20 public class Code2_16 {
21
22    public static void main(String[] args) {
23        // TODO Auto-generated method stub
24        Dog d = new Dog();
25        Cat c = new Cat();
26        Parrot p = new Parrot();
27
28        d.bark();
29        c.meow();
30        p.sing();
31
32        System.out.print("20170677_오름");
33    }
34 }
35
36
```

Console Output:

```
<terminated> Code2_16 [Java Application] C:\Program Files\Java\jdk-18.0.1\bin\javaw.exe (2023. 3. 22. 오전 10:50:54 - 오전 10:50:54) [pid
Dog said : bark!
Cat said : meow!
Parrot said : sing!
20170677_오름
```

## Code 2-17

```

1 package week2;
2
3 abstract class Pet {
4     public abstract void talk();
5 }
6
7 class Dog extends Pet {
8     public void talk() {
9         System.out.println("Dog said : bark!");
10    }
11 }
12 class Cat extends Pet {
13     public void talk() {
14         System.out.println("Cat said : meow!");
15    }
16 }
17 class Parrot extends Pet {
18     public void talk() {
19         System.out.println("Parrot said : sing!");
20    }
21 }
22
23 public class Code2_17 {
24
25     public static void groupTalk(Pet[] p) {
26         int i;
27
28         for (i = 0; i < 3; i++)
29             p[i].talk();
30     }
31
32     public static void main(String[] args) {
33         // TODO Auto-generated method stub
34         Pet[] p = {new Cat(), new Dog(), new Parrot()};
35         groupTalk(p);
36         System.out.print("20170677_오늘식");
37     }
38 }
39
40

```

```

<terminated> Code2_17 [Java Application] C:\Program Files\Java\jdk-18.0.1\bin\javaw.exe (2023. 3. 22. 오전 11:20:00 - 오전 11:20:02) [pid
Cat said : meow!
Dog said : bark!
Parrot said : sing!
20170677_오늘식

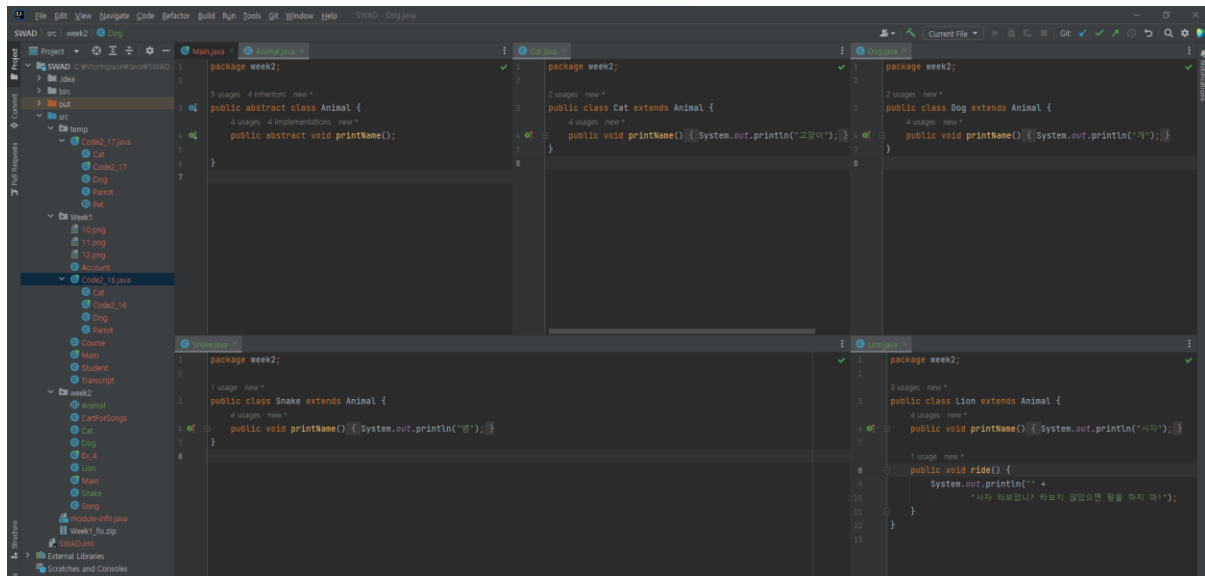
```

두 코드의 가장 큰 차이점은 2-16과는 다르게 2-17에서는 일반화를 통해 Pet이라는 상위 추상클래스를 생성하고 groupTalk()에서 추상 메소드 talk를 호출하여 사용함으로써 다형성을 사용하고 있다는 점입니다.

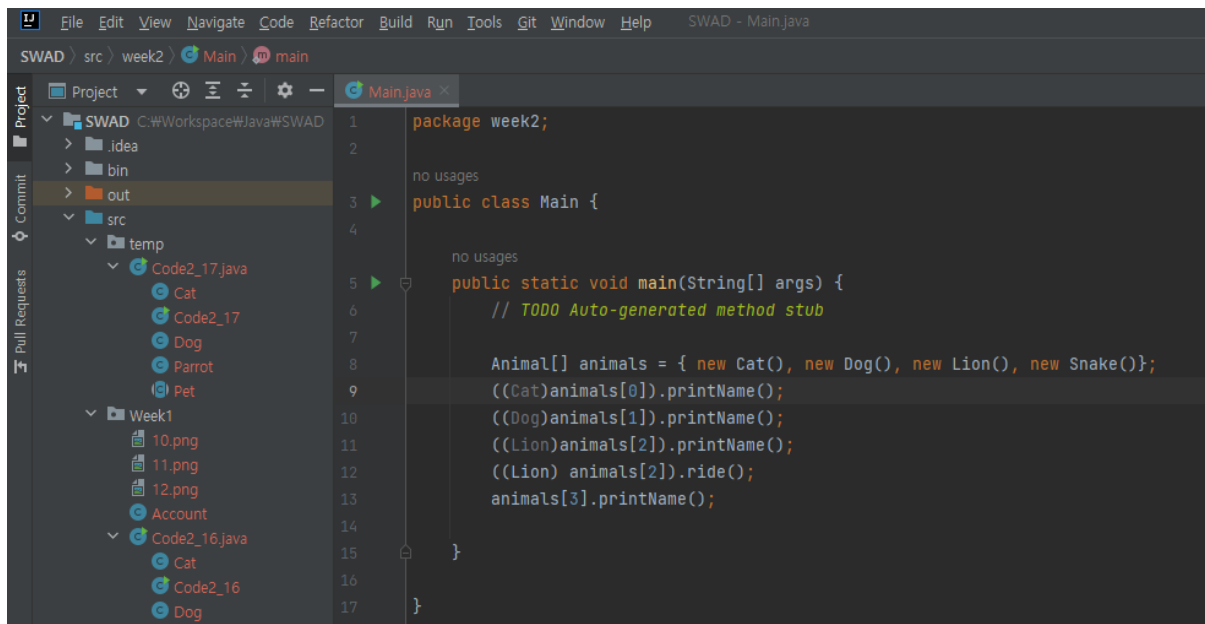
일반화를 통해 객체를 생성하면 부모 클래스의 참조 변수가 자식 클래스의 객체를 참조할 수 있기 때문에 코드가 간결해집니다. 만약 다형성을 형성하지 않았다면 talk()메소드를 사용할 수 없기 때문에 사용하는 클래스(Cat, Dog, Parrot...)마다 talk()에 대응하는 메소드를 구현해 주어야 하며, 요구사항의 변경 등의 이유로 인해 코드가 변경되었을 경우 사용된 코드 모두를 변경해 주어야 하는데, 다형성을 형성할 경우 코드가 변경되었더라도 상속을 통해 사용할 수 있기 때문에 좀 더 유연하고 간결하게 대체할 수 있다는 장점이 있습니다.

[2] p.84 체크포인트 풀이를 이용하여, 잘 동작하는 코드를 작성하고 코드와 실행결과를 설명하시오.

Code [Animal, Cat, Dog,  
Snake, Lion]



Main



문제가 발생하는 줄 : Main.java line 10, 11, 12

Line 10 과 11 은 다형성을 통해 생성된 인스턴스이지만 형제 클래스 끼리는 다형성을 가지고 있더라 하더라도 강제 형변환이 이루어 질 수 없다. 때문에 이를 해결하기 위해선 자신의 생성 타입과 맞는 형태로 형변환을 하든지 부모 클래스의 형태로 호출되어야 한다.

Ex) ((Cat)animals[1]).printName(); -> ((Dog)animals[1]).printName();

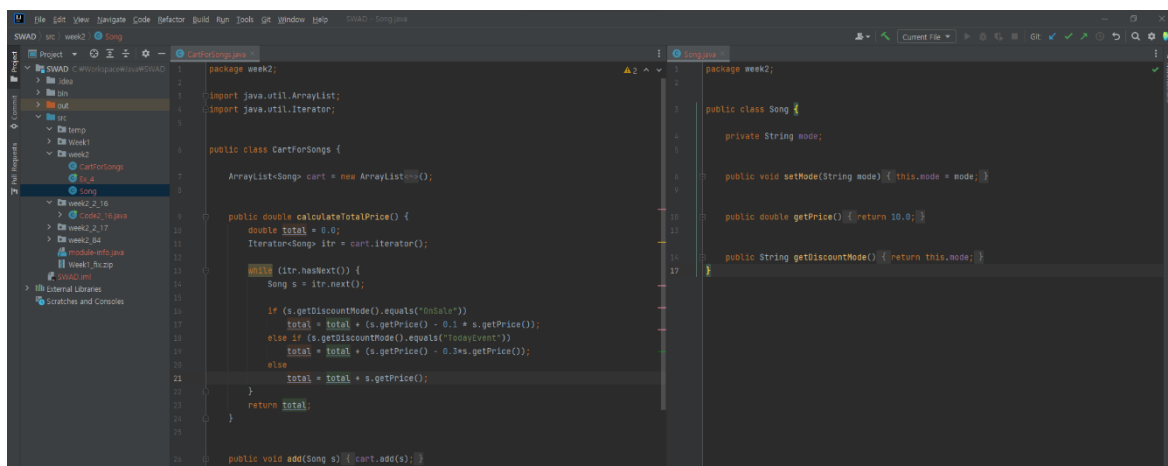
> Animals[1].printName();

또한 line12 번은 부모 클래스에 없는 메서드(ride();)를 사용하려 하고 있기 때문에 에러가 발생합니다. 이를 해결하기 위해서는 부모 클래스에 ride() 라는 추상 메서드를 선언한 뒤 다른 형제 클래스들에 ride();라는 메서드를 구현하거나, animals[2]를 Lion 형으로 강제 형 변환 시키면 에러가 해결됩니다.

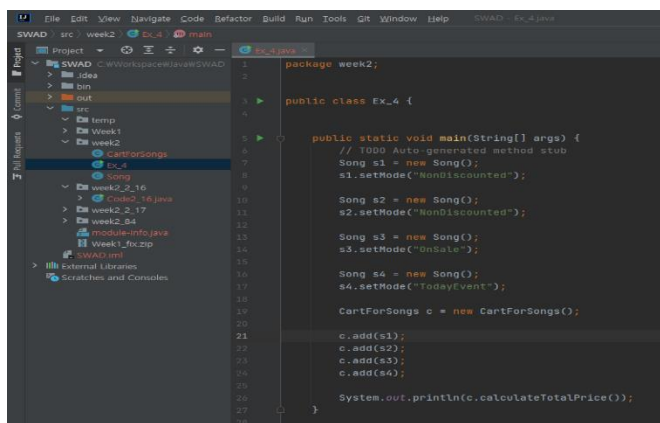
Ex) animals[2].ride(); -> ((Lion)animals[2]).ride();

[2] 연습문제 4 번 : 문제점을 설명하고, 문제점을 해결하도록 코드를 개선하시오. (Hint : p.398~)

Ex4) CartForSongs.java, Song.java



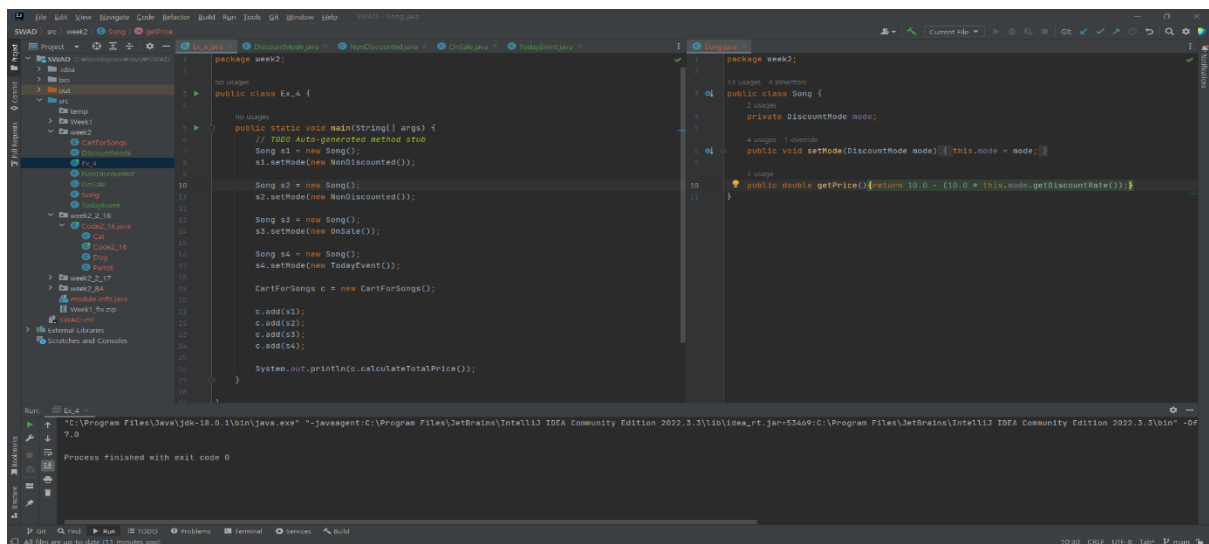
Ex\_4.Java



Ex4의 문제점은 요구사항 등의 변경점이 있을 때마다 메서드를 일일이 변경시켜주고, main에서도 변경시켜주어야 한다는 점이 있습니다. 이렇게 되면 코드가 간결하지 않은 문제가 있습니다. 이를 해결하기 위해서 문자열로 입력받았던 getDiscountMode를 Song의 하위 클래스 DiscountMode로 일반화하여 분리해낸 뒤, 다시 DiscountMode에서 객체를 생성하여 메서드를 오버라이딩을 통해 구분하면 캡슐화를 통해 정보를 은닉할수도 있고, 다형성을 형성할 수도 있습니다.

수정된 코드

Ex4.java, Song.java



```
package week2;

// no usages
public class Ex_4 {
    // no usages
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Song s1 = new Song();
        s1.setMode(new NonDiscounted());

        Song s2 = new Song();
        s2.setMode(new NonDiscounted());

        Song s3 = new Song();
        s3.setMode(new OnSale());

        Song s4 = new Song();
        s4.setMode(new TodayEvent());

        CartForSongs c = new CartForSongs();
        c.add(s1);
        c.add(s2);
        c.add(s3);
        c.add(s4);

        System.out.println(c.calculateTotalPrice());
    }
}

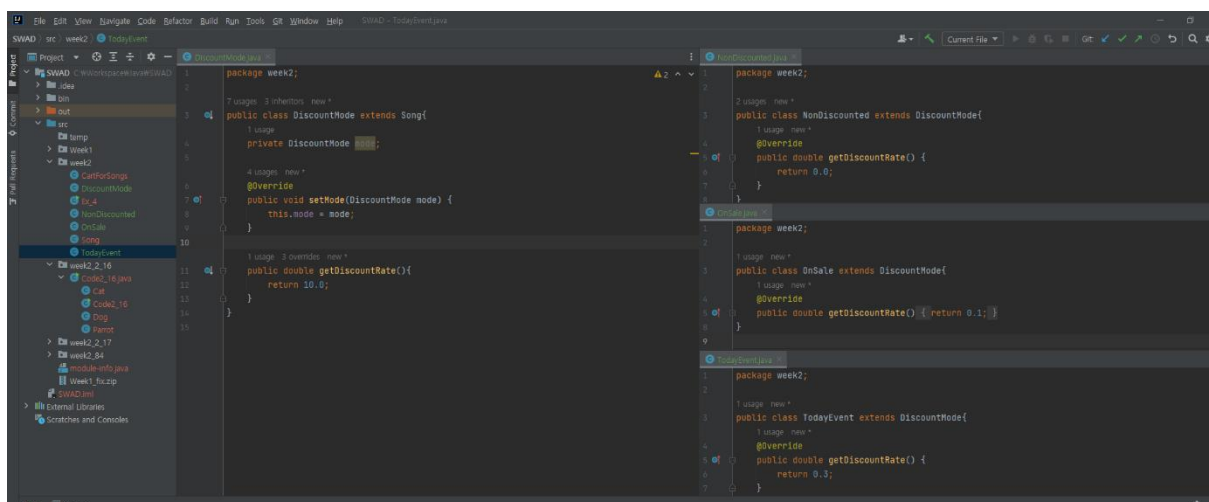
package week2;

// 14 usages, 4 submethods
public class Song {
    // 2 usages
    private DiscountMode mode;

    // 4 usages, 1 override
    public void setMode(DiscountMode mode) { this.mode = mode; }

    // 1 usage
    public double getPrice() { return 10.0 - (10.0 * this.mode.getDiscountRate()); }
}
```

DiscountMode.java, NonDiscounted.java, OnSale.java, TodayEvent.java



```
package week2;

// 7 usages, 3 inheritors, new *
public class DiscountMode extends Song {
    // 1 usage, new *
    private DiscountMode mode;

    // 4 usages, new *
    @Override
    public void setMode(DiscountMode mode) {
        this.mode = mode;
    }

    // 1 usage, 3 overrides, new *
    public double getDiscountRate() {
        return 10.0;
    }
}

package week2;

// 2 usages, new *
public class NonDiscounted extends DiscountMode {
    // 1 usage, new *
    @Override
    public double getDiscountRate() {
        return 0.0;
    }
}

package week2;

// 1 usage, new *
public class OnSale extends DiscountMode {
    // 1 usage, new *
    @Override
    public double getDiscountRate() { return 0.1; }
}

package week2;

// 1 usage, new *
public class TodayEvent extends DiscountMode {
    // 1 usage, new *
    @Override
    public double getDiscountRate() {
        return 0.3;
    }
}
```