6주차 과제

study 할래?

스터디 내용

목표 : 자바의 상속에 대해 학습

1. 자바 상속의 특징
2. super 키워드
3. 메소드 오버라이딩
4. 다이나믹 메소드 디스패치 (Dynamic Method Dispatch)
5. 추상 클래스
6. final 키워드
7. Object 클래스

1. Java 상속의 특징

상속이란?

상속이라는 단어의 뜻과 마찬가지로 자바에서도 상속은 부모클래스의 변수와 메소드를 물려받는것을 말한다. 이런 상속은 코드의 재사용성을 통해 코드의 간결성을 확보해준다.

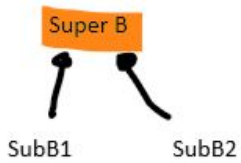
특징

1. 상속은 단일 상속만 가능하다.
2. 자바의 계층 구조 최상위에는 java.lang.Object 클래스가 존재한다.
3. 자바에서는 상속의 횟수에 제한을 두지 않는다.
4. 부모의 메소드와 변수만 상속되며, 생성자는 상속되지 않는다.
   (부모의 메소드는 재정의 하여 사용 가능하다 - 오버라이딩)

1. 상속 메모리

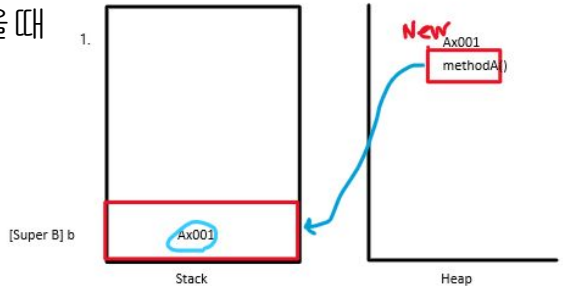
"그렇다면 오버라이딩을 했을때 슈퍼클래스의 메서드를 사용하고 싶다면?"

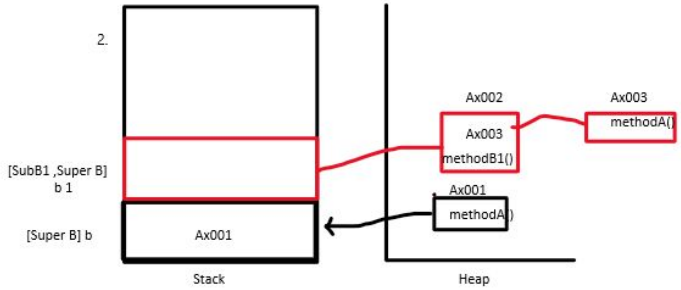
```
1  class SuperB{
2  public void methodA(){
3      System.out.println("SuperB's methodA()");
4  }
5  }//SuperB
6  class SubB1 extends SuperB{
7      public void methodB1(){
8          System.out.println("SubB1's methodB1()");
9      }
10 }
11 class SubB2 extends SuperB{
12     public void methodA(){
13         System.out.println("SubB2's methodA()");
14     }
15 }
16
17 public class InherTest_02 {
18     public static void main(String[] args) {
19         SuperB b=new SuperB();
20         b.methodA();
21         SubB1 b1=new SubB1();
22         b1.methodA();
23         SubB2 b2=new SubB2();
24         b2.methodA();
25     }//main
26 }
27
```

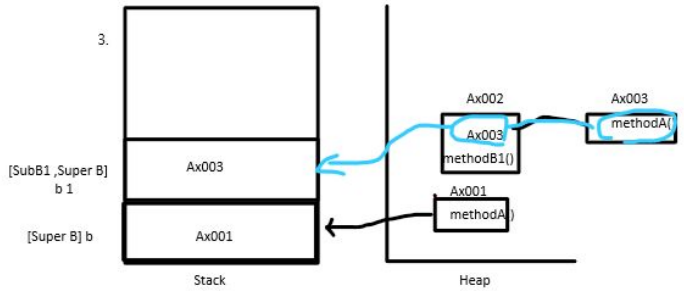



```
SuperB's methodA()
SuperB's methodA()
SubB2's methodA()
```

Stack 메모리는 컴파일할때 잡힘







2. SUPER 키워드

"그렇다면 오버라이딩을 했을때 슈퍼클래스의 메서드를 사용하고 싶다면?"

→ SUPER 키워드를 사용하면 된다.

Super 키워드는 자식클래스가 부모클래스로부터 상속받은 멤버를 사용하고자 할때 사용된다.

```
SubB2's methodA()
SuperB's methodA()
```

```java
class SuperB{
    public void methodA(){
        System.out.println("SuperB's methodA()");
    }
}
class SubB1 extends SuperB {
    public void methodB1(){
        System.out.println("SubB1's methodB1()");
    }
}
class SubB2 extends SuperB {
    public void methodA(){
        System.out.println("SubB2's methodA()");
        super.methodA();
    }
}


public class InherTest_02 {
    public static void main(String[] args) {
        SubB2 b2 = new SubB2();
        b2.methodA();

    }
}
```

3. 메소드 오버라이딩

오버라이딩은 부모의 함수를 재정의하는 기능이다. 같은 동작에서 다른 기능을 구현해야 하는 경우가 있는데 그때 사용한다. 그렇기 때문에 함수명, 리턴값, 파라미터가 모두 동일해야한다.

```
class SuperB{
    public void methodA(){
        System.out.println("SuperB's methodA()");
    }
}
class SubB1 extends SuperB {
    public void methodB1(){
        System.out.println("SubB1's methodB1()");
    }
}
class SubB2 extends SuperB {
    public void methodA(){
        System.out.println("SubB2's methodA()");
        super.methodA();
    }
}


public class InherTest_02 {
    public static void main(String[] args) {
        SubB2 b2 = new SubB2();
        b2.methodA();

    }
}
```

4. 다이나믹 메소드 디스패치_메소드 디스패치

Method Dispatch는 어떤 메소드를 호출할 지 결정하여 실제로 실행시키는 과정을 말한다. 이런 메소드 디스패치에는 정적 메소드 디스패치(Static Method Dispatch), 동적 메소드 디스패치(Dynamic Method Dispatch), 더블 디스패치(Double Dispatch) 세 가지가 존재한다.

1. Static Method Dispatch

   상속 관계에서 오버라이딩 된 소스가 있다. 메인함수에서 서브클래스에서 오버라이딩 된 함수를 부르게되면 슈퍼클래스의 것이 아닌 서브클래스의 메소드가 호출되고 실행된다는걸 우리와 컴파일러 역시 명확하게 알고있다. 이를 정적메소드 디스패치라고 부른다.

4. 다이나믹 메소드 디스패치_메소드 디스패치

2. Dynamic Method Dispatch

동적 메소드 디스패치는 정적 디스패치와는 다르게 컴파일러가 어떤 메소드를 호출해야 되는지 모르는 것을 말한다.

우측 예제에서 guideLine이라는 추상 클래스는 B, C로 각각 구현되고 있다. 또한 A라는 클래스는 interface를 받아 print라는 함수를 사용하고있다. 그렇다면 여기서 A의 print를 사용하면 어떤 함수가 호출될까? 우리는 해당 함수의 객체를 선언할 때에 할당 된 Object를 보고 어떤 함수를 실행할지 결정하게 될 것이다.

하지만 컴파일러는 유추가 불가능하다. 즉, 컴파일러는 어떤 함수가 실행될 지 모르고 런타임 시점이 되서야 알 수 있다.

```
interface guideLine{
    void print();
}

class A{
    private guideLine gl;

    public A(guideLine gl) {
        this.gl = gl;
    }

    public void print() {
        gl.print();

    }
}
class B implements guideLine {
    @Override
    public void print() {
        System.out.println("B's print()");
    }
}
class C implements guideLine {
    @Override
    public void print() {
        System.out.println("C's print()");
    }
}
```

4. 더블디스패치

https://blog.naver.com/swoh1227/222181505425

5. 추상클래스

추상클래스는 클래스를 만들기 위한 일종의 설계도로 인스턴스를 생성할 수는 없는 클래스이다. 이를 사용하기 위해서는 반드시 자식 클래스에서 상속받아 클래스를 모두 구현해야만 한다. 동시에 추상클래스는 반드시 하나 이상의 추상 메서드를 포함하고 있으며, 생성자와 멤버변수, 일반메서드 모두를 가질 수 있다.

https://m.blog.naver.com/PostView.nhn?blogId=ljhjjang0125&logNo=110136634405&proxyReferer=https:%2F%2Fwww.google.com%2F

6. final 키워드

7. Object 클래스