제11장 상속

상속 (inheritance)

“이전에 만든 클래스와 닮았지만, 일부 다른 클래스"를 만들 필요가 있을 경우가 늘어날 것이다.

리스트 11-1 “싸우기" 와 “도망"만 되는 Hero 클래스


```java
public class Hero {
    private String name = "김영웅";
    private int hp = 100;

    // 싸우기
    public void attack(Kinoko enemy) {
        System.out.println(this.name + "의 공격!");
        this.hp -= 5;
        System.out.println("5포인트의 데미지를 주었다!");
    }
    // 도망
    public void run() {
        System.out.println(this.name + "는 도망쳤다!");
    }
}
```

리스트 11-2 SuperHero 클래스

```java
public class SuperHero {
    private String name = "김영웅";
    private int hp = 100;
    private boolean flying;      // 필드 추가

    // 싸우기
    public void attack(Kinoko enemy) {
        System.out.println(this.name + "의 공격!");
        this.hp -= 5;
        System.out.println("5포인트의 데미지를 주었다!");
    }
    // 도망
    public void run() {
        System.out.println(this.name + "는 도망쳤다!");
    }
    // 날기
    public void fly() {
        flying = true;
        System.out.println("날았다!");
    }
    // 착지
    public void land() {
        flying = false;
        System.out.println("착지했다!");
    }
}
```

복사 붙여넣기의 문제점

- 추가, 수정에 시간이 걸림
- 소스의 파악이나 관리가 어려워 짐

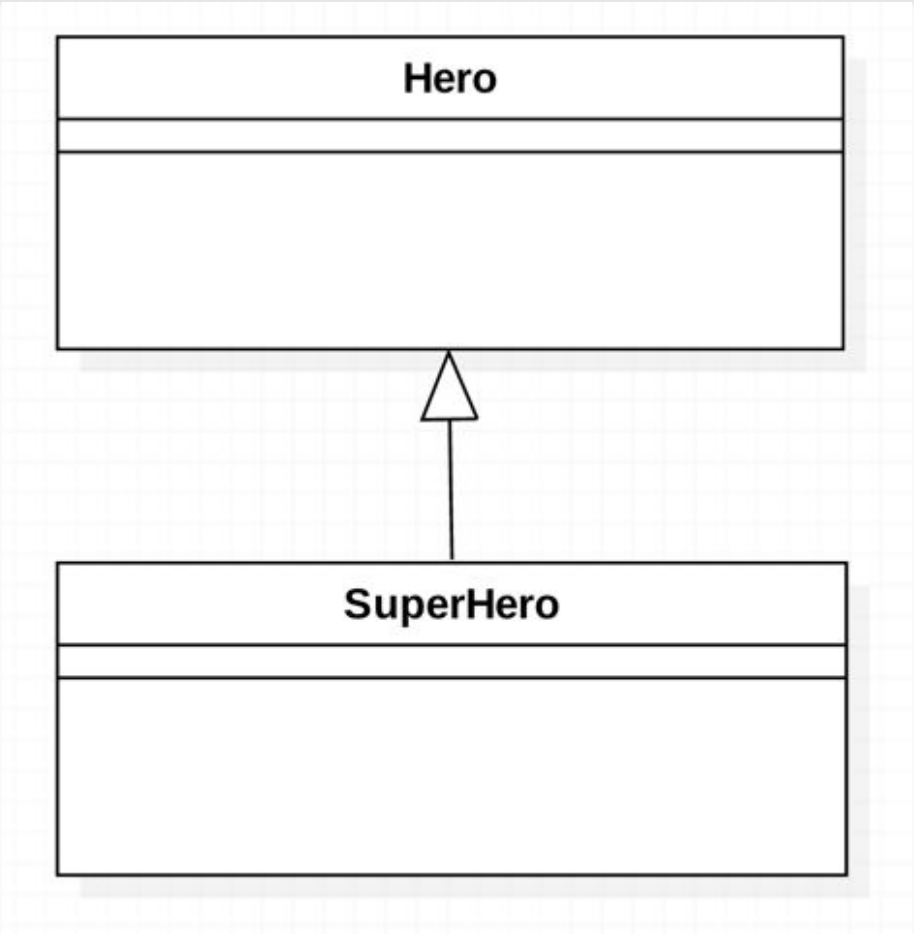
해결책으로 “상속" 을 활용

리스트 11-3 Hero 클래스를 상속한 SuperHero

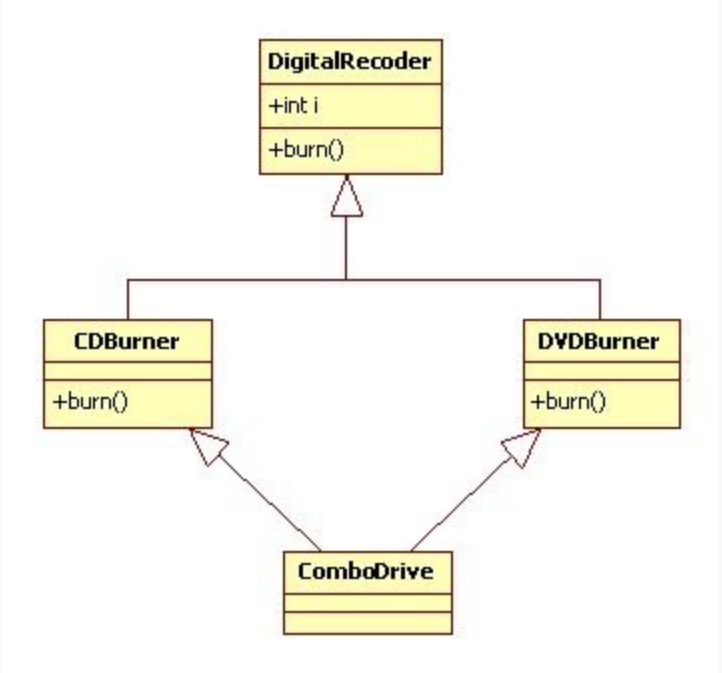

```java
public class SuperHero extends Hero {
    private boolean flying;      // 추가한 필드

    // 추가한 메소드
    public void fly() {
        flying = true;
        System.out.println("날았다!");
    }
    // 추가한 메소드
    public void land() {
        flying = false;
        System.out.println("착지했다!");
    }
}
```

```
1  public class Main {
2      public static void main(String[] args) {
3          SuperHero superHero = new SuperHero();
4          superHero.run();
5      }
6  }
```

상속관계의 표현 방법



다중상속은 Java에서 금지



리스트 11-5 오버라이드 (override)


```java
public class SuperHero extends Hero {
    private boolean flying;

    public void fly() {
        flying = true;
        System.out.println("날았다!");
    }

    public void land() {
        flying = false;
        System.out.println("착지했다!");
    }

    @Override
    public void run() {
        System.out.println("퇴각했다");
    }
}
```

```
1  public class Main {
2      public static void main(String[] args) {
3          Hero hero = new Hero();
4          hero.run();
5          SuperHero superHero = new SuperHero();
6          superHero.run();
7      }
8  }
```

상속을 금지하는 방법. final


```
1  // String 클래스를 상속하면 에러 ??
2  public class LimitString extends String {
3
4  }
```

```
public final class String
    implements java.io.Serializable, Comparable<String>, CharSequence {
```

```java
public class Hero {
    private String name = "김영웅";
    private int hp = 100;

    public final void slip() {
        hp -= 5;
        System.out.println(name + "는 미끄러졌다!");
        System.out.println("5의 데미지!");
    }
    // 도망
    public void run() {
        System.out.println(name + "는 도망쳤다!");
    }

```

리스트 11-8 SuperHero의 추가 사양


```
1   public class SuperHero extends Hero {
2       private boolean flying;
3
4       @Override
5       public void attack(Kinoko enemy) {
6           System.out.println(this.name + "의 공격!");
7           enemy.hp -= 5;
8           System.out.println("5포인트의 데미지를 주었다!");
9
10          if (this.flying) {
11              System.out.println(this.name + "의 공격!");
12              enemy.hp -= 5;
13              System.out.println("5포인트의 데미지를 주었다!");
14          }
15      }
16
```

리스트 11-9 부모 객체를 참조하는 super 키워드


```
1   public class SuperHero extends Hero {
2       private boolean flying;
3
4       @Override
5       public void attack(Kinoko enemy) {
6           super.attack(enemy);
7
8           if (this.flying) {
9               System.out.println(this.name + "의 공격!");
10              enemy.hp -= 5;
11              System.out.println("5포인트의 데미지를 주었다!");
12          }
13      }
14
```

리스트 11-10 상속과 생성자


```java
public class Hero {
    public Hero() {
        System.out.println("Hero 생성자");
    }
}
```

```java
public class SuperHero extends Hero {
    public SuperHero() {
        System.out.println("SuperHero의 생성자");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        SuperHero superHero = new SuperHero();
    }
}
```

리스트 11-11 기본 생성자가 없을 때의 에러 해결


```java
public class Item {
    private String name;
    private int price;

    public Item(String name) {
        this.name = name;
        this.price = 0;
    }

    public Item(String name, int price) {
        this.name = name;
        this.price = price;
    }
}
```

```java
public class Weapon extends Item {
    public Weapon(String name) {
        super(name);
    }
    public Weapon(String name, int price) {
        super(name, price);
    }
}
```

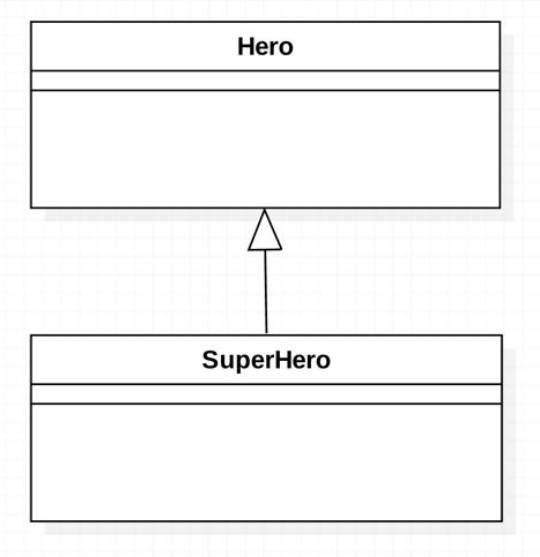
```java
public class Main {
    public static void main(String[] args) {
        Weapon weapon = new Weapon();
    }
}
```

올바른 상속

올바른 상속은 **"is-a 원칙" 이라고 하는 규칙에 따른 상속**을 말한다

SuperHero is a Hero

(SuperHero 는 Hero의 한 종류 이다)



잘못 된 상속

현실 세계의 등장인물 사이에 **개념적으로 is-a 관계가 되지 못 함에도 불구하고 상속을 사용한 경우가 “잘못 된 상속"** 이다.

예) 필드로 이름과 가격을 가지는 Item 클래스 (약초, 포션) 를 상속 받는 House

잘못된 상속을 하면 안 되는 이유

- 클래스를 확장할 때 현실세계와의 모순이 생긴다
- 객체 지향의 3대 특징 중 1가지 “다형성" 을 이용할 수 없게 된다

```
1  public class Item {
2      // 적에게 던졌을 때 입힐 수 있는 데미지
3      public int getDagame() {
4          return 10;
5      }
```

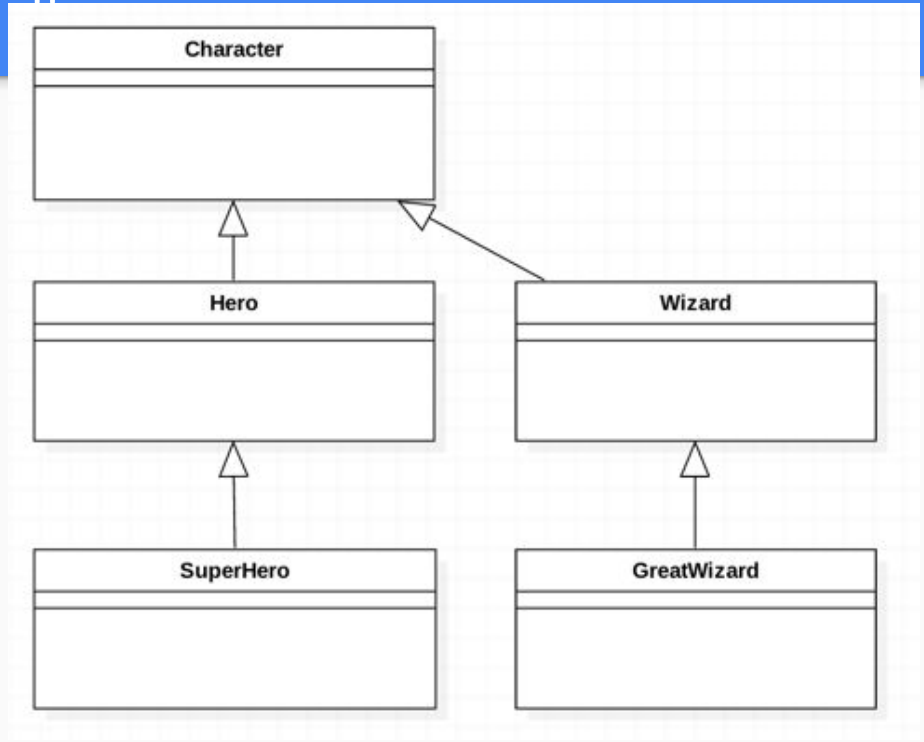
House 가 Item을 상속 받았다면, House를 던질 수 있지만, 현실세계에서
집을 던지는 것은 이상하다

구체화와 일반화의 관계

자식클래스 일 수록 **구체화** 되고,

부모 클래스 일 수록 **추상적**인 것으로

일반화 된다.



제11장의 정리

상속의 기초
- extends를 사용하여 기존 클래스를 기초로 하는 새로운 클래스를 정의 할 수 있다
- 부모 클래스의 멤버는 자동적으로 자식 클래스에 상속되므로, 자식 클래스에는 추가 된 부분만 기술하면 된다
- 부모 클래스에 있는 메소드를, 자식 클래스에서 재작성 할 경우 이것을 오버라이드 한다고 한다
- final 을 붙인 클래스는 상속이 되지 않고, final 이 붙은 메소드는 오버라이드 되지 않는다
- 올바른 상속이란 “자식 클래스 is-a 부모 클래스"
- 상속에는 “추상적, 구체적" 관계에 있다는 것을 정의하는 역할도 있음

인스턴스
- 인스턴스는 내부에 부모클래스의 인스턴스를 가지는 다중구조를 가진다
- 보다 외측의 인스턴스에 속하는 메소드가 우선적으로 동작한다
- 외측의 인스턴스에 속하는 메소드는 super 을 사용하여 내측 인스턴스의 멤버에 점근할 수 있다

생성자 동작
- 다중구조의 인스턴스가 생성되는데, JVM 는 자동적으로 가장 외측 인스턴스의 생성자를 호출
- 모든 생성자는, “부모 인스턴스의 생성자"를 호출 할 필요가 있다
- 생성자의 선두에 super() 가 없으면, 암묵적인 “super();” 가 추가 됨

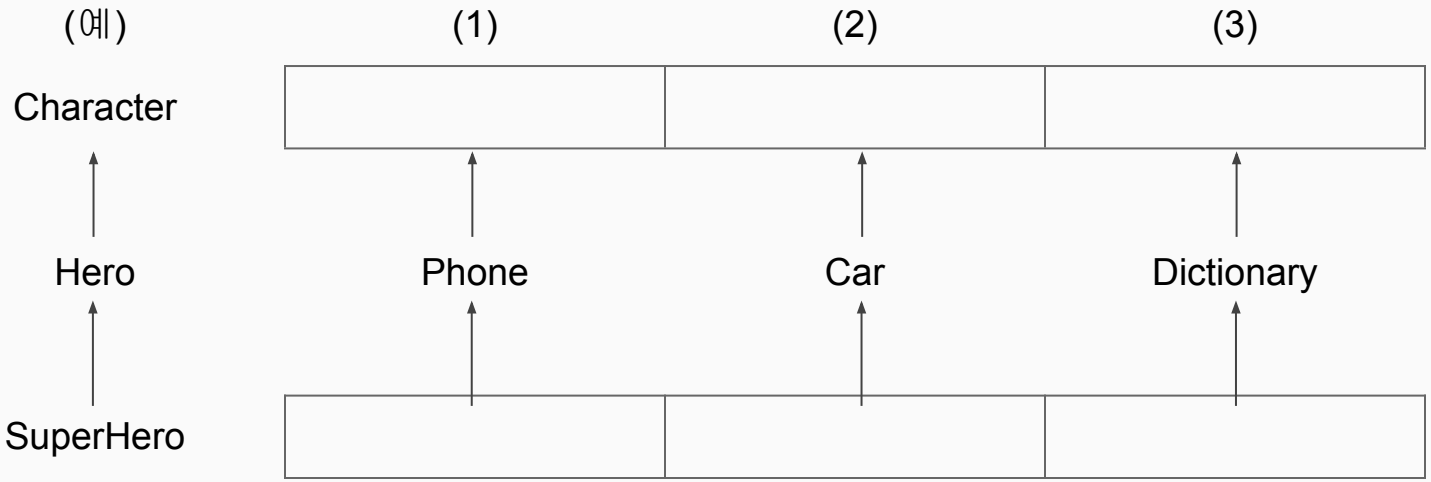
연습문제 11-1

다음 중에서 “잘못 된 상속" 인 것을 모두 구하시오

	슈퍼클래스	서브클래스
1	Person	Student
2	Car	Engine
3	Father	Child
4	Food	Susi
5	SuperMan	Man

연습문제 11-2

다음 클래스에 대해 “부모 클래스" 와 “자식 클래스" 를 1개씩 생각 해 보시오



연습문제 11-3

```java
public class Kinoko {
    int hp = 50;
    private char suffix;

    public Kinoko(char suffix) {
        this.suffix = suffix;
    }

    public void attack(Hero hero) {
        System.out.println("키노코 " + this.suffix + " 의 공격");
        System.out.println("10의 데미지");
        hero.setHp(hero.getHp() - 10);
    }
}
```

이 클래스를 이용해, 다음 사양을 따르는 PoisonKinoko 클래스를 작성하시오

1. 괴물 독버섯(PoisonKinoko) 는, 괴물버섯 (Kinoko) 중에서도 특히 “독 공격" 이 되는 것
2. PoisonKinoko 는 아래의 코드로 인스턴스화 되는 클래스임
   PoisonKinoko poisonKinoko = new PoisonKinoko(‘A’);
3. PoisonKinoko는 독 공격이 가능한 남은 횟수를 int 형 필드를 가지고 있고 초기값은 5 이다
4. PoisonKinoko는 attack() 메소드가 호출되면 다음 내용의 공격을 한다
   a. 우선, “보통 괴물버섯과 같은 공격"을 한다
   b. “독 공격의 남은 횟수"가 0이 아니면 다음을 추가로 수행한다
   c. 화면에 “추가로, 독 포자를 살포했다!” 를 표시
   d. 용사의 HP 의 1/5에 해당하는 포인트를 용사의 HP 로부터 감소시키고, “~포인트의 데미지" 라고 표시
   e. “독 공격의 남은 횟수" 를 1 감소 시킨다