

능동적 사고 방식의

java

강사 박주병

Park Ju Byeong

Park Ju Byeong



# Part11 다형성



01 다형성

02 인터페이스

03 내부 클래스

04 실습 문제

# 01 —

## 다형성

## 1-1 실습문제 (normal)

### 쇼핑몰 장바구니를 구현해보자

- 클래스 : ShoppingBasket , TV, Sofa, Bicycle
- 장바구니는 여러 아이템들을 가진다.(장바구니와 아이템은 포함관계이다.)
- 장바구니는 여러 개의 TV, Sofa, Bicycle을 가질 수 있다.

따라서 멤버변수로 TV[], Sofa[],Bicycle[] 처럼 배열로써 3가지 종류의 물건을 가져야 하지만 배열 대신 앞서 배운 List를 이용하여 보자

```
public class Main {  
    public static void main(String[] args) {  
  
        ShoppingBasket basket = new ShoppingBasket();  
  
        //장바구니의 멤버변수로 물건을 담아줄 3개의 List를 가지고 있다.  
        basket.bicycleList.add(new Bicycle());  
        basket.tvList.add(new TV());  
        basket.sofaList.add(new Sofa());  
        basket.sofaList.add(new Sofa());  
  
    }  
}
```

## 1-1 문제풀이 (normal)

```
public class ShoppingBasket {  
  
    public List<TV> tvList = new ArrayList<TV>();  
    public List<Sofa> sofaList = new ArrayList<Sofa>();  
    public List<Bicycle> bicycleList = new ArrayList<Bicycle>();  
}
```

물건 여러 개가 들어가야 하니 List를 활용한다. List의 요소는 어떤타입이 될지는 <> 사이에 작성한다.

List 역시 클래스 이므로 객체 생성을 해야 사용할수 있다.

## 1-2 실습문제 (normal)

1-1에서 만든 장바구니 클래스는 물건의 종류마다 별도의 멤버변수로 가지고 있다. 이러한 방식의 문제점은 물건의 종류가 늘어날 때마다 멤버변수를 계속 추가해줘야 한다는 것이다.

이 문제는 TV, Sofa, Bicycle 클래스가 Item 이라는 클래스를 부모로 설정하고 장바구니 클래스는 List<Item> 타입의 멤버변수 1개 만을 가진다면 해결된다.

해당 멤버변수는 어떠한 타입의 객체이건 Item의 자식이기만 하면 다형성에 의해 List의 요소로 추가될 수 있다.

- 클래스 : Item

```
public class Main {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        ShoppingBasket basket = new ShoppingBasket();  
  
        //장바구니에는 다형성에 의해 Item의 자식타입이면 어떤 객체든 들어간다.  
        basket.itemList.add(new TV());  
        basket.itemList.add(new Sofa());  
    }  
}
```

## 1-2 문제풀이 (normal)

- ShoppingBasket은 여러 개의 Item을 가진다. 따라서 포함관계인 멤버변수로 표현을 한다.
- Item 타입은 다형성으로 인해 Item의 자식들 모두를 가질 수 있다. 따라서 itemList배열은 item의 자식이면 어떤 객체든 넣을 수 있다.

```
1 public class ShoppingBasket {  
2  
3     List<Item> itemList;  
4  
5 }  
6
```

```
1 public class Item {  
2  
3  
4 }  
5
```

```
1  
2  
3 public class TV extends Item{  
4  
5 }  
6
```

```
1  
2  
3 public class Sofa extends Item{  
4  
5 }  
6
```

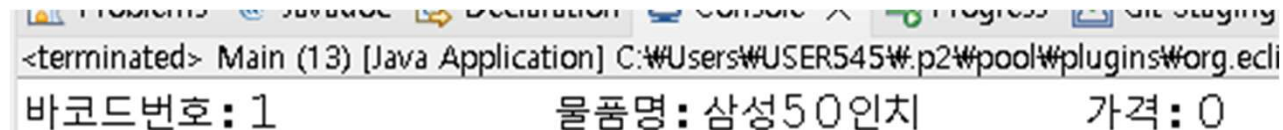
```
1  
2  
3 public class Bicycle extends Item{  
4  
5 }  
6
```

## 1-3 실습문제 (normal)

클래스들의 내부를 구현해보자.

- 모든 물품들은 int barcodeNumber, String name, int price를 가지고 있어야 한다.  
(이 멤버변수들은 어떤 클래스에 있어야 할까?)
- 멤버변수는 생성자를 통해 초기화 하자(생성자는 상속되지 않는다!)
- Item 클래스는 객체화 될 필요가 있는가?

```
public class Main {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        ShoppingBasket sb = new ShoppingBasket();  
  
        sb.itemList.add(new TV(1, "삼성50인치", 0));  
        sb.itemList.add(new Sofa(2, "폭신한쇼파", 23));  
        sb.itemList.add(new Bicycle(3, "천리마 자전거", 100));  
  
        Item item = sb.itemList.get(0);  
  
        System.out.println("바코드번호: "+item.barcodeNumber  
            +"\t\t물품명: "+item.name  
            +"\t가격: "+item.price  
            );  
    }  
}
```



<terminated> Main (13) [Java Application] C:\Users\USER545\p2\pool\plugins\org.eclipse.jdt.launcher\org.eclipse.jdt.launcher\_3.10.0.v20160510-1900.jar -x -console -data C:\Users\USER545\p2\pool\workspace\1-3 실습문제 (normal)\src\Main.java -e -vm C:\Program Files\Java\jdk-8.0.600\bin\java.exe.  
바코드번호 : 1                      물품명 : 삼성50인치                      가격 : 0

다형성에 의해 장바구니에 어떤 객체가 들어있든 꺼내서 Item타입의 변수에 넣을 수 있다.



## 1-3 문제풀이 (normal)

```
public abstract class Item {
```

```
    public int barcodNumber;  
    public String name;  
    public int price;
```

```
    public Item()  
    {
```

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

```
public class TV extends Item{
```

```
    public TV()  
    {
```

```
    public TV(int barcodNumber, String name, int price)  
    {  
        super(barcodNumber, name, price);  
    }
```

Item 클래스는 객체화될 필요가 없다.  
오직 상속의 목적만을 가진다.

부모의 생성자는 상속이 안되지만  
자식에서 super를 이용해 사용 할 수 있다.  
따라서 모든 자식마다 생성자를 만들지 않고  
부모에서 만든 생성자를 재활용 하도록 한다.

자식에서 부모의 생성자를 이용해  
멤버변수를 초기화 하고 있다. 멤버변수  
역시 부모가 상속시켜준것이기에 가능하다.

Park Ju B, Jong

## 1-4 실습문제 (normal)

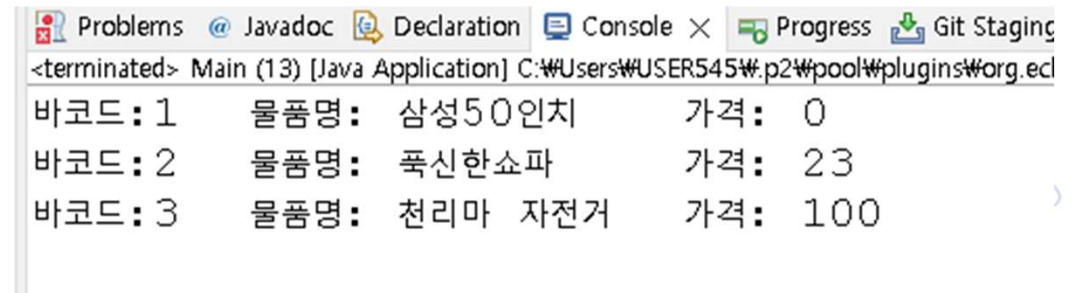
toString()메서드를 오버라이딩 하자

- toString()메서드는 Object가 상속해준 메서드이다.
- 멤버변수의 값들을 문자열로 반환하는 기능으로 오버라이딩 해보자.
- Item 클래스에서 toString()을 오버라이딩하면 자식클래스에 상속이 되므로 Item클래스에서만 오버라이딩하면 된다.

```
ShoppingBasket sb = new ShoppingBasket();
```

```
sb.itemList.add(new TV(1, "삼성50인치", 0));  
sb.itemList.add(new Sofa(2, "폭신한쇼파", 23));  
sb.itemList.add(new Bicycle(3, "천리마 자전거", 100));
```

```
for(Item item : sb.itemList)  
{  
    String result = item.toString();  
  
    System.out.println(result);  
}
```



## 1-4 문제풀이 (normal)

```
public abstract class Item {
```

```
    public int barcodNumber;  
    public String name;  
    public int price;  
    int mileagePercent;  
    public Item()  
    {  
  
    }
```

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

```
    public String toString()  
    {  
        return "바코드:" + barcodNumber + "\t물품명: " + name + "\t가격: " + price;  
    }
```

Object가 상속해준 toString()을 오버라이딩한다.  
이클립스의 자동완성기능(ctrl+스페이스)를  
활용해도 된다.

문자열을 만들어 반환한다. 절대 직접 출력하지 말자.

## 1-5 실습문제 (normal)

ShoppingBasket 클래스의 기능을 추가하자.

- ShoppingBasket 클래스에 현재 장바구니에 있는 물품이름과 가격을 문자열로 반환하는 String getInfoList() 메서드를 만들자
- 앞서 만든 Item 클래스의 toString() 기능을 이용하면 편리하다.

**\*문자열에 줄넘김 문자를 포함하는 방법**

```
// TODO Auto-generated method stub  
System.out.println("동해물과" + System.lineSeparator() + "백두산이");
```

Problems Javadoc  
<terminated> Main (3) [Java]  
동해물과  
백두산이

```
ShoppingBasket shopping = new ShoppingBasket();
```

```
shopping.itemList.add(new Sofa(1, "샤넬소파", 25000));  
shopping.itemList.add(new Sofa(2, "폭신한소파", 25000));  
shopping.itemList.add(new TV(3, "삼성 QLED", 10000));  
shopping.itemList.add(new Bicycle(1, "빠른 자전거", 5000));  
System.out.println(shopping.getInfoList());
```

Problems Javadoc Declaration Console × Progress Git Sta  
<terminated> Main (13) [Java Application] C:\Users\USER545\p2\pool\plugins\wor  
바코드: 1 물품명: 샤넬소파 가격: 25000  
바코드: 2 물품명: 폭신한소파 가격: 25000  
바코드: 3 물품명: 삼성 QLED 가격: 10000  
바코드: 1 물품명: 빠른 자전거 가격: 5000

Park Ju Byeol

## 1-5 문제풀이 (normal)

```
public class ShoppingBasket {  
  
    //현재 장바구니에 담긴 물품 리스트  
    public List<Item> itemList = new ArrayList();  
  
    public String getInfoList()  
    {  
        String infoList="";  
  
        for(Item item: itemList)  
            infoList += item.toString() + System.lineSeparator();  
  
        return infoList;  
    }  
}
```

Item의 toString()은 객체의 값을 표현하도록 이미 만들어져 있으므로 재활용한다.

## 1-6 실습문제 (normal)

ShoppingBasket 클래스의 기능을 추가하자.

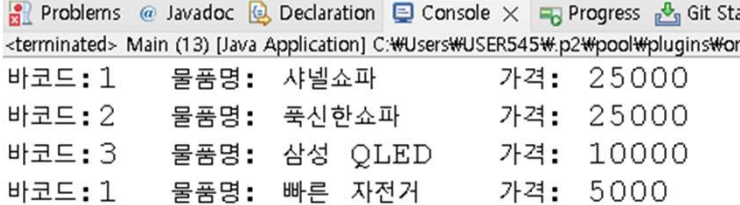
- ShoppingBasket 클래스에 현재 장바구니에 있는 물품 가격의 합계를 구하는 getTotalPrice() 메서드를 만들어보자.

```
ShoppingBasket shopping = new ShoppingBasket();

shopping.itemList.add(new Sofa(1, "샤넬쇼파", 25000));
shopping.itemList.add(new Sofa(2, "폭신한쇼파", 25000));
shopping.itemList.add(new TV(3, "삼성 QLED", 10000));
shopping.itemList.add(new Bicycle(1, "빠른 자전거", 5000));

System.out.println(shopping.getInfoList());

System.out.println("합계: " + shopping.getTotalPrice());
```



바코드	물품명	가격
1	샤넬쇼파	25000
2	폭신한쇼파	25000
3	삼성 QLED	10000
1	빠른 자전거	5000

합계: 65000

## 1-6 문제풀이 (normal)

```
public class ShoppingBasket {  
  
    //현재 장바구니에 담긴 물품 리스트  
    public List<Item> itemList = new ArrayList();  
  
    public String getInfoList()  
    {  
        String infoList="";  
  
        for(Item item: itemList)  
            infoList += item.toString() + System.lineSeparator();  
  
        return infoList;  
    }  
  
    public int getTotalPrice()  
    {  
        int sum =0;  
  
        for(Item item: itemList)  
            sum +=item.price;  
  
        return sum;  
    }  
}
```

foreach문을 이용해 장바구니에 들어있는  
물품리스트를 하나씩 가져와 가격의 합계를  
구한다.



## 1-7 실습문제 (hard)

**현재 자전거 재고가 없어 장바구니에 자전거를 담지 못하게 해보자.**

- ShoppingBasket 클래스의 멤버변수인 itemList는 현재 외부에서 사용 할 수 있게 열려 있다.
- itemList 가 외부에 열려 있으면 리스트에 특정 물건을 add 하는걸 막을 방법이 없다.  
따라서 private으로 itemList를 외부에서 직접 사용하지 못하게 막아야 한다.
- 대신 물건을 넣을 다른 방법을 제공해줘야 한다. ShoppingBasket 클래스에서 멤버변수인 itemList에 물건을 넣을수 있는 public void addItem(Item item) 메서드를 만들어야 한다.
- **addItem메서드 내부에서 instanceof 활용해서 item의 객체가 자전거인지 확인해야 한다.**

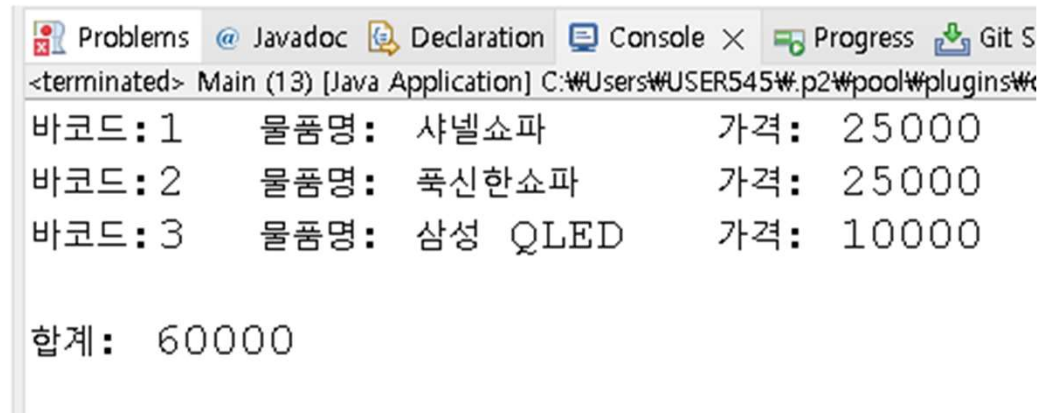
```
ShoppingBasket shopping = new ShoppingBasket();
```

```
//itemList를 외부에서 직접 사용하지 못한다.  
//shopping.itemList.add(new Sofa(1, "샤넬쇼파", 25000));  
//shopping.itemList.add(new Sofa(2, "폭신한쇼파", 25000));  
//shopping.itemList.add(new TV(3, "삼성 QLED", 10000));  
//shopping.itemList.add(new Bicycle(1, "빠른 자전거", 5000));
```

```
//필터링 기능이 있는 addItem을 사용한다.  
shopping.addItem(new Sofa(1, "샤넬쇼파", 25000));  
shopping.addItem(new Sofa(2, "폭신한쇼파", 25000));  
shopping.addItem(new TV(3, "삼성 QLED", 10000));  
shopping.addItem(new Bicycle(1, "빠른 자전거", 5000));
```

```
System.out.println(shopping.getInfoList());
```

```
System.out.println("합계: "+shopping.getTotalPrice());
```



The screenshot shows the IDE's console window with the following output:

바코드	물품명	가격
1	샤넬쇼파	25000
2	폭신한쇼파	25000
3	삼성 QLED	10000

합계: 60000



## 1-7 문제풀이 (hard)

```
public class ShoppingBasket {
```

```
//현재 장바구니에 담긴 물품 리스트
```

```
private List<Item> itemList = new ArrayList();
```

```
public String getInfoList()
```

```
{
```

```
    String infoList="";
```

```
    for(Item item: itemList)
```

List를 밖에서 직접 사용하면  
필터링 할 수 없다.

```
public void addItem(Item item)
```

```
{
```

```
    if(item instanceof Bicycle)
```

```
        return;
```

```
    itemList.add(item);
```

```
}
```

장바구니에 넣으려는 물건이  
자전거 인지 판단한다.

void 이더라도 return으로 메서드를  
종료하는것은 가능하다.

내부에서는 List의 add메서드를  
이용한다.

# — 04

## 실습문제

## 2-1 실습문제(normal)

User, Weapon, Repairable, Sword, Gun, Punch 클래스를 만들고 각각의 관계에 맞게 클래스 설계를 하자.

Weapon: 추상 클래스   Repairable : 인터페이스

- User는 Weapon을 가진다.(무기는 1개만 가질수 있다)
- Sword, Gun은 수리가 가능하다.

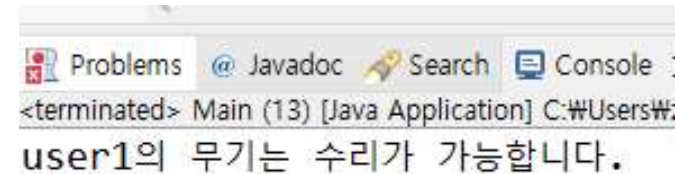
```
User user1 = new User();  
user1.Weapon = new Gun();
```

```
User user2 = new User();  
user2.Weapon = new Punch();
```

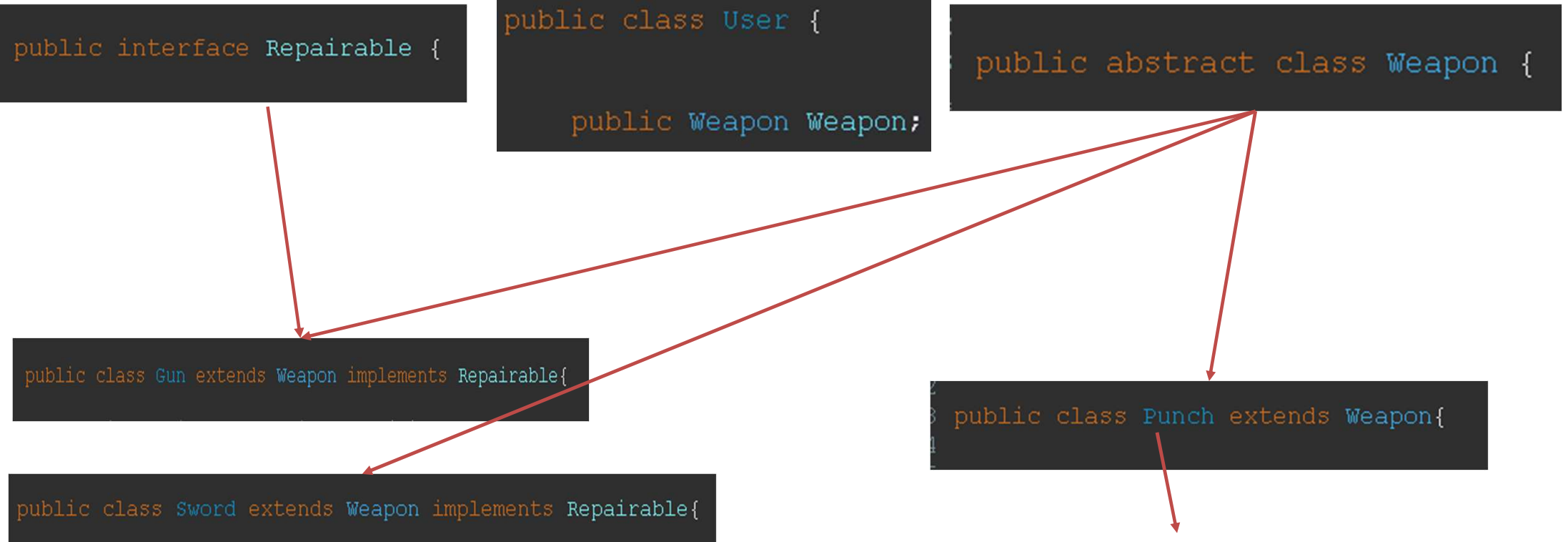
```
User user3 = new User();  
user3.Weapon = new Sword();
```

```
if(user1.Weapon instanceof Repairable)  
    System.out.println("user1의 무기는 수리가 가능합니다.");
```

```
if(user2.Weapon instanceof Repairable)  
    System.out.println("user2의 무기는 수리가 가능합니다.");
```



## 2-1 문제풀이(normal)



Punch는 수리가 불가능하므로  
Repairable 인터페이스를 구현하지 않는다.

## 2-1 문제풀이(normal)

```
User user1 = new User();  
user1.Weapon = new Gun();
```

다형성으로 인해 Gun클래스의 부모는 Weapon이고  
Weapon타입의 변수에 들어갈수 있다.

```
User user2 = new User();  
user2.Weapon = new Punch();
```

```
User user3 = new User();  
user3.Weapon = new Sword();
```

참조변수에 들어가 있는 실질적인 객체가  
Repairable 인터페이스를 구현한 객체라면  
true가 된다.

```
if(user1.Weapon instanceof Repairable)  
    System.out.println("user1의 무기는 수리가 가능합니다.");
```

```
if(user2.Weapon instanceof Repairable)  
    System.out.println("user2의 무기는 수리가 가능합니다.");
```

## 2-2 실습문제(normal)

2-1 문제에서 만든 클래스에 기능을 추가해보자.

- User에 멤버변수 String id, int hp 를 추가 한다.
- User는 생성자를 이용해 멤버변수를 초기화 한다.
- 모든 무기는 멤버변수 int damage, int durability 를 가진다.어느 클래스에 추가할지 생각해보자.
- 모든 무기들은 생성자로 멤버변수를 초기화 한다.
- User 클래스와 모든 무기 클래스들은 toString()을 오버라이딩하여 멤버변수의 값을 문자열로 반환한다.

```
User user1 = new User("랭킹1위가자",100,new Gun(10,5));
```

```
User user2 = new User("똥겜망해라",70,new Sword(15,10));
```

```
System.out.println(user1.toString());
```

```
System.out.println(user2.toString());
```

```
<terminated> Main (13) [Java Application] C:\Users\zest1\p2\pool\plugins\org.eclipse.justj.o  
아이디: 랭킹1위가자 체력: 100 무기 : {데미지: 10 내구도: 5}  
아이디: 똥겜망해라 체력: 70 무기 : {데미지: 15 내구도: 10}
```

## 2-2 문제풀이(normal)

```
public class User {  
    String id;  
    int hp;  
    public Weapon Weapon;
```

```
    public User()  
    {  
    }
```

생성자를 통해 멤버변수를 초기화 한다.  
무기 객체도 생성자를 통해 받는다.

```
    public User(String id, int hp, Weapon Weapon)  
    {  
        this.id = id;  
        this.hp = hp;  
        this.Weapon = Weapon;  
    }
```

Object클래스가 상속해준 메서드를 오버라이딩

```
    public String toString()  
    {  
        return "아이디: " + id + " 체력: " + hp + " 무기 : {" + Weapon.toString() + "}";  
    }
```

Weapon의 toString()을 활용한다.



## 2-2 문제풀이(normal)

- 모든 무기들이 공통으로 가져야 하는 멤버변수와 메서드를 만든다.

```
public abstract class Weapon {  
  
    int damage;  
    int durability;  
  
    public Weapon()  
    {  
    }  
  
    public Weapon(int damage, int durability)  
    {  
        this.damage = damage;  
        this.durability = durability;  
    }  
  
    public String toString()  
    {  
        return "데미지: " + damage + " 내구도: " + durability;  
    }  
}
```

```
public class Gun extends Weapon implements Repairable{  
  
    public Gun()  
    {  
    }  
  
    public Gun(int damage, int durability)  
    {  
        super( damage, durability);  
    }  
    public class Punch extends Weapon{  
  
        public Punch()  
        {  
        }  
        public Punch(int damage, int durability)  
        {  
            super( damage, durability);  
        }  
    }  
    public class Sword extends Weapon implements Repairable{  
  
        public Sword()  
        {  
        }  
        public Sword(int damage, int durability)  
        {  
            super( damage, durability);  
        }  
    }  
}
```



## 2-3 실습문제(hard)

2.User클래스에 공격할수 있는기능을 추가하자.

- User는 void attack(User) 메서드를 추가하고 매개변수로 들어온 User의 체력을 this의공격력 만큼 감소 시키자.
- User는 attack 메서드 내부에서 누가 누구를 공격했는지 출력하자.
- 모든 무기들은 한번 공격시 durability가 1씩 감소된다.
- 내구도가 없다면 공격할 수 없고 그럴 경우 sysout으로 내구도가 없음을 알려주자.

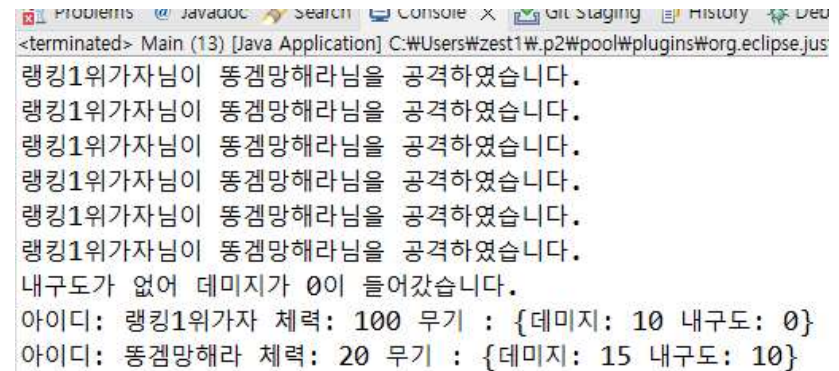
```
User user1 = new User("랭킹1위가자",100,new Gun(10,5));
```

```
User user2 = new User("동검망해라",70,new Sword(15,10));
```

```
user1.attack(user2);
user1.attack(user2);
user1.attack(user2);
user1.attack(user2);
user1.attack(user2);
user1.attack(user2);
user1.attack(user2);
```

→ 내구도가 없어 공격할수없다

```
System.out.println(user1.toString());
System.out.println(user2.toString());
```



The screenshot shows the Eclipse IDE console with the following output:

```
<terminated> Main (13) [Java Application] C:\Users\wzest1\p2\pool\plugins\org.eclipse.jdt
랭킹1위가자님이 동검망해라님을 공격하였습니다.
랭킹1위가자님이 동검망해라님을 공격하였습니다.
랭킹1위가자님이 동검망해라님을 공격하였습니다.
랭킹1위가자님이 동검망해라님을 공격하였습니다.
랭킹1위가자님이 동검망해라님을 공격하였습니다.
랭킹1위가자님이 동검망해라님을 공격하였습니다.
내구도가 없어 데미지가 0이 들어갔습니다.
아이디: 랭킹1위가자 체력: 100 무기 : {데미지: 10 내구도: 0}
아이디: 동검망해라 체력: 20 무기 : {데미지: 15 내구도: 10}
```

## 2-3 문제풀이(hard)

```
public void attack(User target)
{
    System.out.println(id+"님이 " + target.id+"님을 공격하였습니다.");

    if(Weapon.durability <= 0)
    {
        System.out.println("내구도가 없어 데미지가 0이 들어갔습니다.");
        return;
    }
```

target.hp -=Weapon.damage; → 자신의 무기의 공격력만큼  
상대방의 체력을 감소시킨다.

```
//내구도 차감
Weapon.durability--;
```

→ 공격에 성공했으니 자신의 내구도를  
감소시킨다.

## 2-4 실습문제(hard)

BlackSmith(대장장이) 클래스를 추가하여 무기를 수리해보자.

BlackSmith 클래스에 void repaire(Weapon) 메서드를 구현하자.

- 매개변수로 받은 무기의 durability를 10 회복시킨다.
- Sword,Gun은 수리가 가능하다.
- 수리가 불가능한 무기가 매개변수로 오면 sysout으로 거부 메시지를 출력하자
- Repairable 인터페이스를 활용하자.

//무기 내구도를 0으로 준다.

```
User user1 = new User("랭킹1위가자",100,new Gun(10,0));
```

```
User user2 = new User("통검망해라",70,new Punch(15,0));
```

//대장장이 객체 생성

```
BlackSmith bm = new BlackSmith();
```

//유저1의 무기를 대장장이에게 맡긴다.

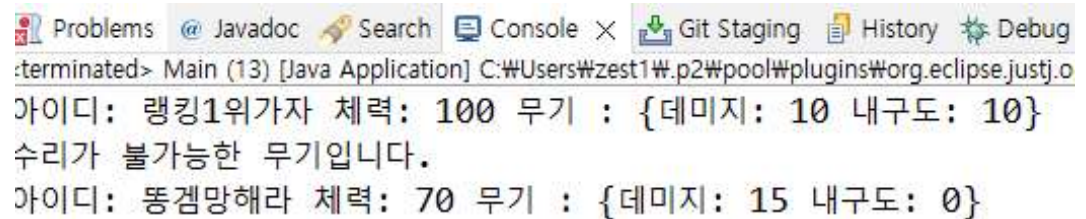
```
bm.repaire(user1.Weapon);
```

```
System.out.println(user1.toString());
```

//유저2의 무기를 대장장이에게 맡긴다.Punch는 수리가 불가능하다.

```
bm.repaire(user2.Weapon);
```

```
System.out.println(user2.toString());
```



Problems Javadoc Search Console × Git Staging History Debug  
:terminated> Main (13) [Java Application] C:\Users\zest1\p2\pool\plugins\org.eclipse.justj.o  
아이디: 랭킹1위가자 체력: 100 무기 : {데미지: 10 내구도: 10}  
수리가 불가능한 무기입니다.  
아이디: 통검망해라 체력: 70 무기 : {데미지: 15 내구도: 0}

## 2-4 문제풀이(hard)

```
public class BlackSmith {
```

```
    public void repaire(Weapon weapon)  
    {
```

```
        if(weapon instanceof Repairable)
```

```
            weapon.durability+=10;
```

```
        else
```

```
            System.out.println("수리가 불가능한 무기입니다.");
```

```
    }
```

→ 인터페이스를 구현한  
클래스만 수리를 한다.

## 2-5 실습문제(expert)

### BlackSmith(대장장이) 를 개선해보자

현재 BlackSmith는 repaire메서드 내부에서 직접 무기를 수리하고 있다.

그러나 무기마다 수리하는 방법이 다를것이고 내구도 또한 한번에 올라가는 양이 다를수 있다.

대장장이의 repaire() 메서드 내부에서 if문을 통해 무기마다 수리되는 방법을 다르게 준다면 기능구현은 가능하나 유지보수 측면에서 좋지 않다.(무기가 추가될때마다 대장장이 클래스가 수정되어야 하기때문)  
그러므로 해당 코드는 다른곳으로 옮겨야 한다.

Hint : Repairable 인터페이스에 repaired() 추상메서드를 추가한다.

각각의 무기 클래스마다 자신이 어떻게 수리 되어야 할지를 repaired() 추상메서드를 구현함으로써 정의한다. 그리고 대장장이의 repaire() 메서드는 매개변수로 받은 무기 객체의 repaired()를 호출함으로써 수리되는 타이밍만 결정하고 어떻게 수리 할지는 무기 스스로에게 위임한다.

제어역전(IOC)에 대해 공부해보면 방법을 알 수 있다.

```
User user1 = new User("랭킹1위가자",100,new Gun(10,0));
```

```
User user2 = new User("똥겜망해라",70,new Sword(15,0));
```

```
BlackSmith bm = new BlackSmith();
```

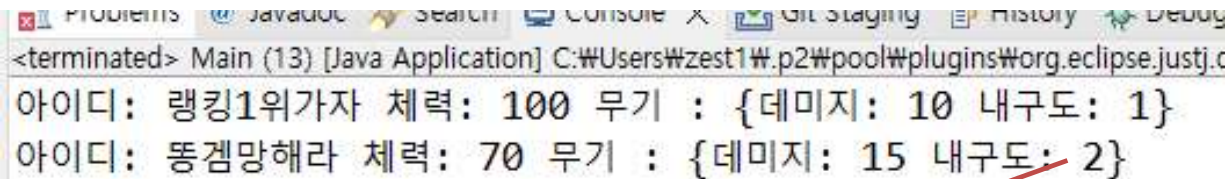
```
//유저1의 무기를 대장장에게 맡긴다.
```

```
bm.repaire(user1.Weapon);
```

```
System.out.println(user1.toString());
```

```
bm.repaire(user2.Weapon);
```

```
System.out.println(user2.toString());
```



```
<terminated> Main (13) [Java Application] C:\Users\zest1\p2\pool\plugins\org.eclipse.justj.c  
아이디: 랭킹1위가자 체력: 100 무기 : {데미지: 10 내구도: 1}  
아이디: 똥겜망해라 체력: 70 무기 : {데미지: 15 내구도: 2}
```

무기마다 회복되는양이  
다르다.

Park Ju Byeong



## 2-5 문제풀이(expert)

```
public void repaire(Weapon weapon)
{
    if(weapon instanceof Repairable)
    {
        Repairable temp = (Repairable)weapon;

        temp.repaired();
        //weapon.durability+=10;
    }
    else
        System.out.println("수리가 불가능한 무기입니다.");
}
```

직접 수리하지 않고 무기가  
구현해놓은 메서드를  
호출한다.

```
public class Gun extends Weapon implements Repairable{

    public Gun()
    {
    }

    public Gun(int damage, int durability)
    {
        super( damage, durability);
    }

    @Override
    public void repaired() {
        this.durability++;
    }
}
```

무기마다 자신이 어떻게  
수리될지를 직접 구현한다.



# THANK YOU



강사 박주병