

Java



레퍼런스 타입

레퍼런스 타입에 대하여
열거 타입의 선언과 이용

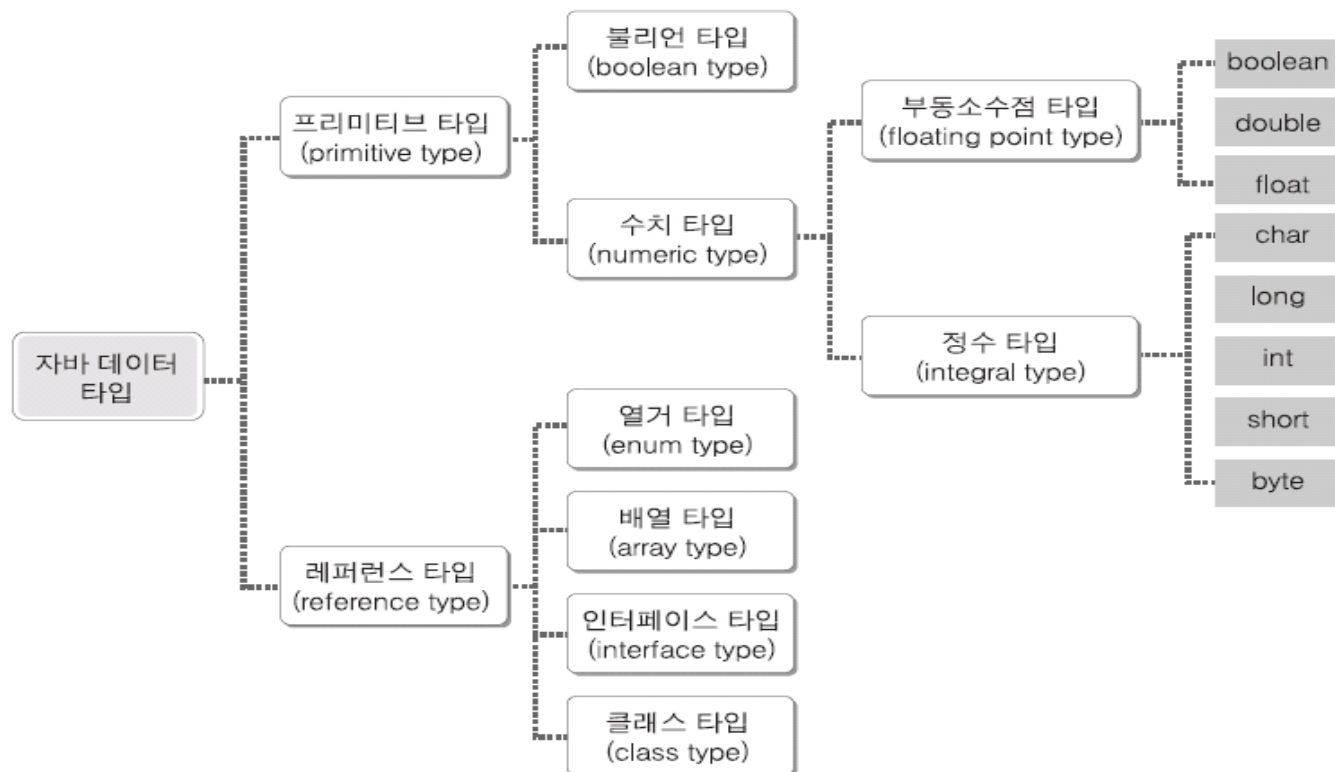


레퍼런스 타입

01. 레퍼런스 타입

레퍼런스 타입이란?

• 자바의 데이터 타입 분류 체계



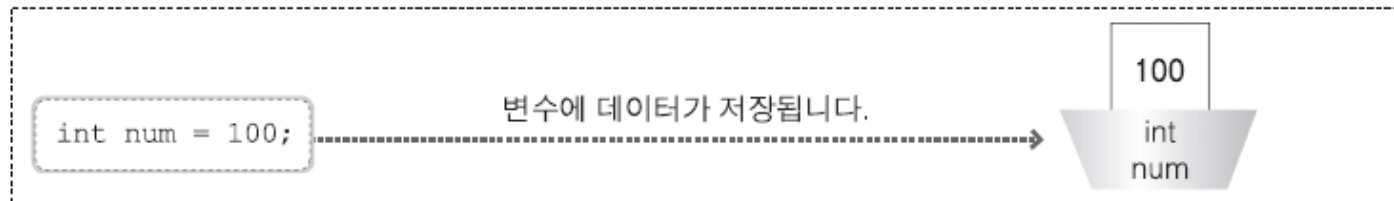
레퍼런스 타입

01. 레퍼런스 타입

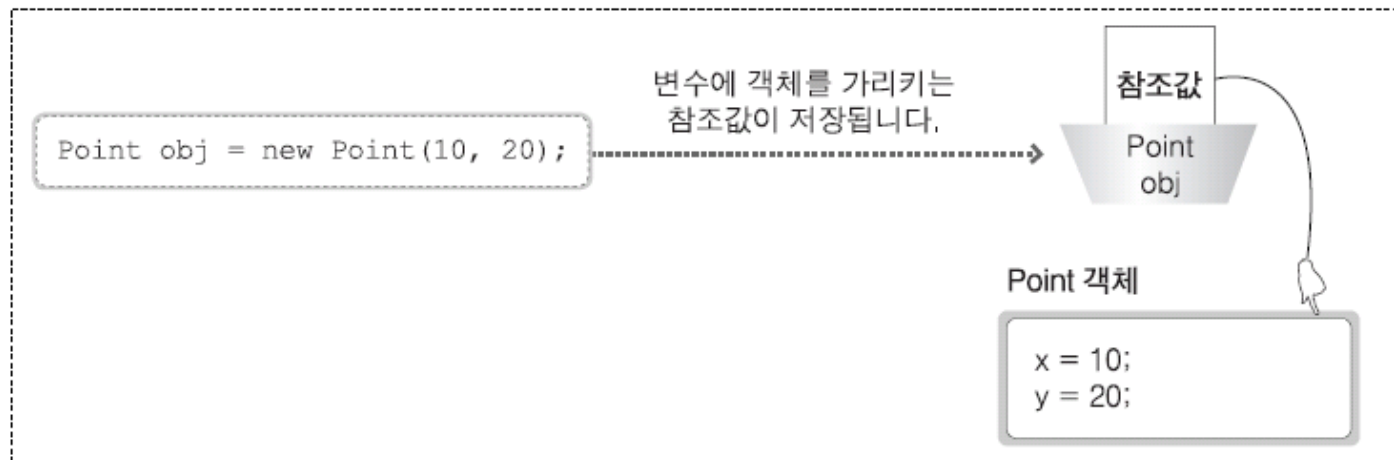
레퍼런스 값에 대하여

- 프리미티브 타입과 레퍼런스 타입의 대입문 처리

a) 프리미티브 타입의 대입문 처리



b) 레퍼런스 타입의 대입문 처리

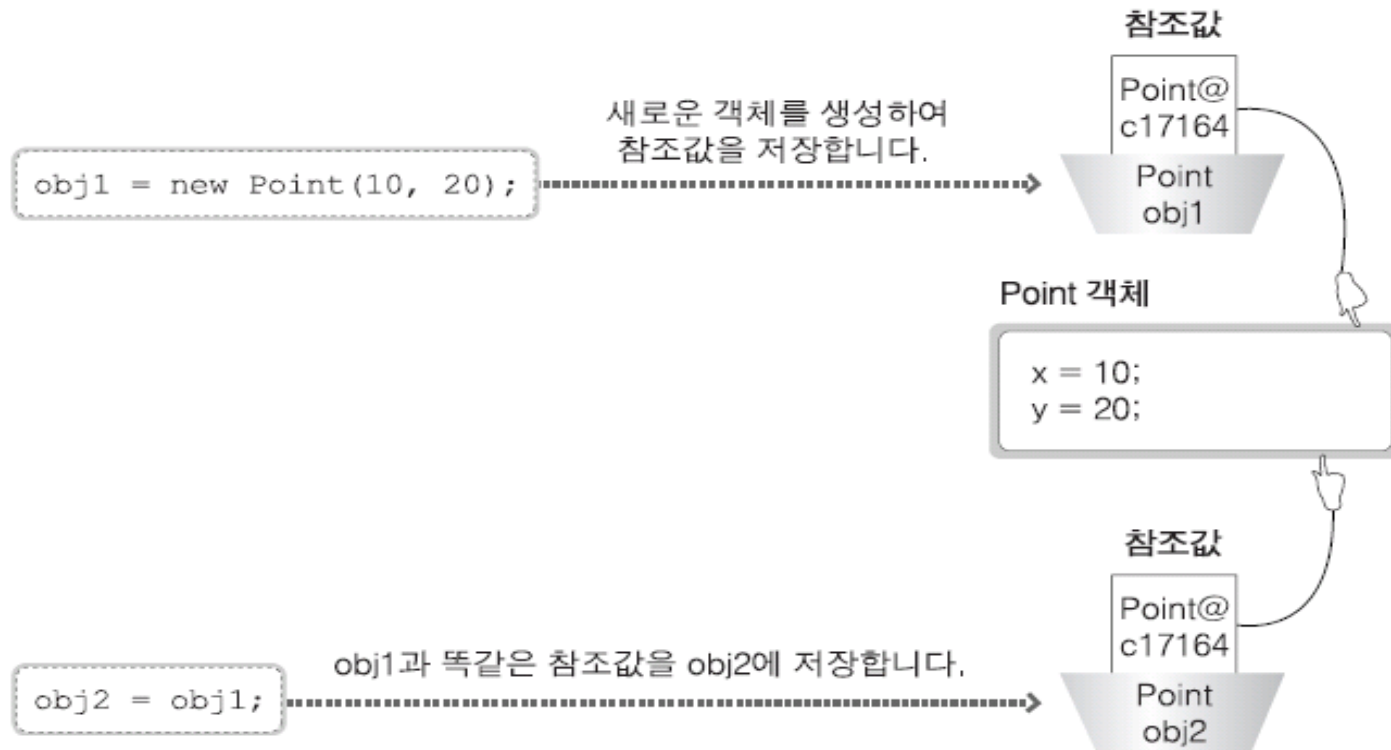


레퍼런스 타입

01. 레퍼런스 타입

참조 값과 데이터

- 레퍼런스 변수를 또 다른 레퍼런스 변수에 대입할 때 일어나는 일



레퍼런스 타입

01. 레퍼런스 타입

참조 값과 데이터

- [예제 7-1] 평면 위의 점을 표현하는 클래스

```
1  class Point {  
2      int x, y;           // 필드  
3      Point(int x, int y) { // 생성자  
4          this.x = x;  
5          this.y = y;  
6      }  
7  }
```

레퍼런스 타입

01. 레퍼런스 타입

참조값과 데이터

- [예제 7-2] 하나의 객체를 두 개의 변수에 대입하는 프로그램

```
1  class RefTypeExample1 {
2      public static void main(String args[]) {
3          Point obj1 = new Point(10, 20);
4          Point obj2 = obj1;
5          System.out.printf("obj1 = (%d, %d) %n", obj1.x, obj1.y);
6          System.out.printf("obj2 = (%d, %d) %n", obj2.x, obj2.y);
7          obj2.x = 30;
8          System.out.printf("obj1 = (%d, %d) %n", obj1.x, obj1.y);
9          System.out.printf("obj2 = (%d, %d) %n", obj2.x, obj2.y);
10     }
11 }
```

하나의 객체를 생성해서
두 개의 변수에 대입

obj2의 x 필드에 다른 값 대입

obj1, obj2의
x, y 필드 값 출력

```
명령 프롬프트
E:\work\chap7\7-1-1>java RefTypeExample1
obj1 = <10, 20>
obj2 = <10, 20>
obj1 = <30, 20>
obj2 = <30, 20>

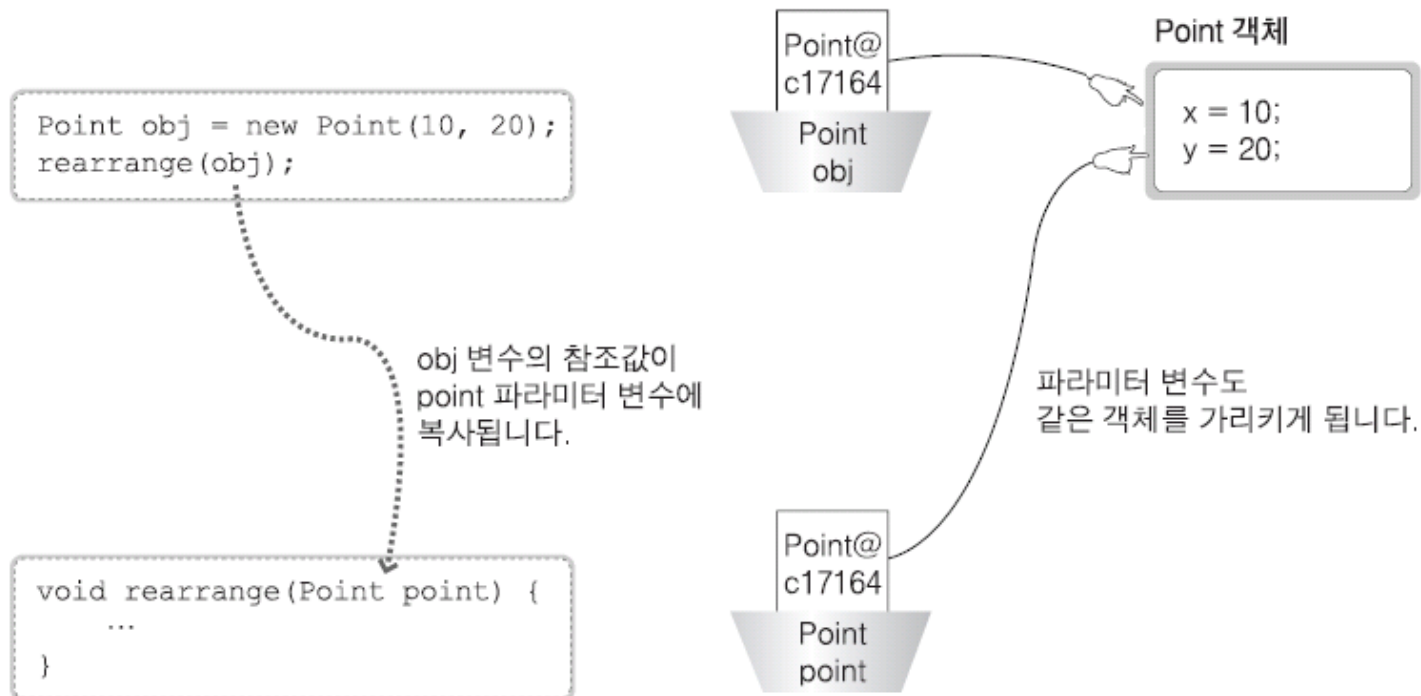
E:\work\chap7\7-1-1>
```

레퍼런스 타입

01. 레퍼런스 타입

레퍼런스 타입의 파라미터

- 레퍼런스 타입의 파라미터를 메소드에 넘겨줄 때 일어나는 일

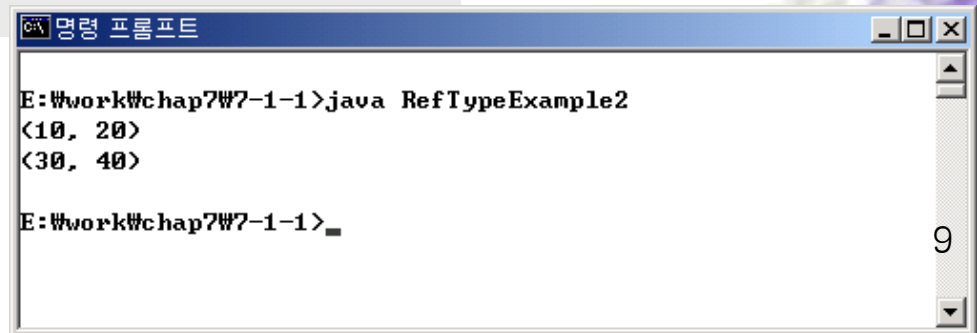


레퍼런스 타입

● 레퍼런스 타입의 파라미터

- [예제 7-3] 레퍼런스 타입 파라미터를 메소드 안에서 수정하는 예

```
1  class RefTypeExample2 {
2      public static void main(String args[]) {
3          Point obj = new Point(10, 20);
4          System.out.printf("(%d, %d) %n", obj.x, obj.y);
5          rearrange(obj); ----- 호출
6          System.out.printf("(%d, %d) %n", obj.x, obj.y);
7      }
8      static void rearrange(Point point) { <-----
9          point.x = 30; }
10         point.y = 40; } 파라미터 변수의 필드에
11     } 다른 값을 대입
12 }
```



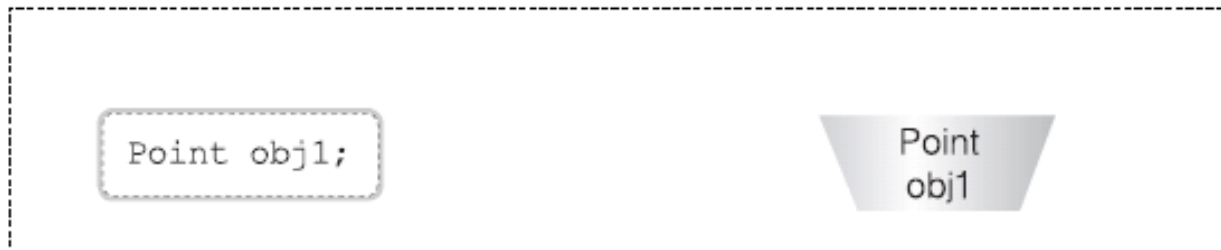
```
명령 프롬프트
E:\work\chap7\7-1-1>java RefTypeExample2
<10, 20>
<30, 40>
E:\work\chap7\7-1-1>
```

레퍼런스 타입

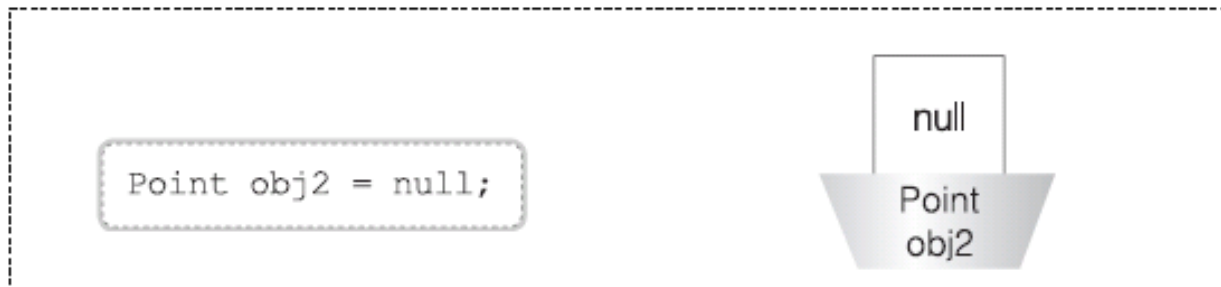
● null 참조값

- null 참조값 : 아무 데이터도 가리키지 않는 참조값

a) 아무 참조값도 갖지 않은 상태



b) null 참조값을 갖고 있는 상태



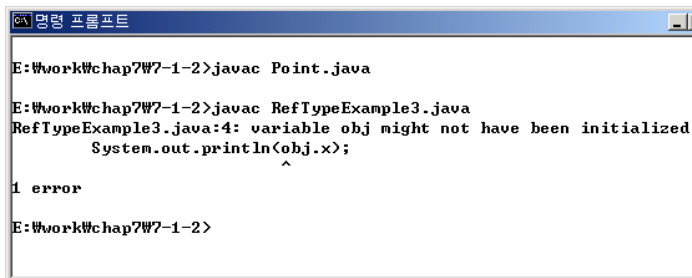
레퍼런스 타입

01. 레퍼런스 타입

● null 참조값

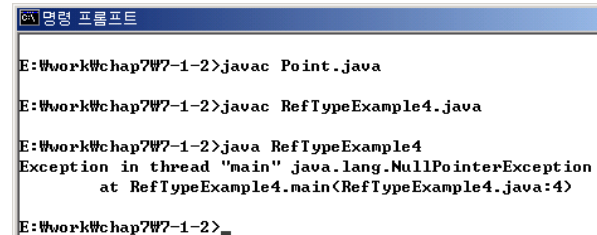
- [예제 7-4] 아무 값도 갖지 않는 레퍼런스 변수와 null 값을 갖는 레퍼런스 변수

```
class RefTypeExample3 {  
    public static void main(String args[]) {  
        Point obj;  
        System.out.println(obj.x);  
        System.out.println(obj.y);  
    }  
}
```



```
명령 프롬프트  
E:\work\chap7\7-1-2>javac Point.java  
E:\work\chap7\7-1-2>javac RefTypeExample3.java  
RefTypeExample3.java:4: variable obj might not have been initialized  
    System.out.println(obj.x);  
                        ^  
1 error  
E:\work\chap7\7-1-2>
```

```
class RefTypeExample4 {  
    public static void main(String args[]) {  
        Point obj = null;  
        System.out.println(obj.x);  
        System.out.println(obj.y);  
    }  
}
```



```
명령 프롬프트  
E:\work\chap7\7-1-2>javac Point.java  
E:\work\chap7\7-1-2>javac RefTypeExample4.java  
E:\work\chap7\7-1-2>java RefTypeExample4  
Exception in thread "main" java.lang.NullPointerException  
    at RefTypeExample4.main(RefTypeExample4.java:4)  
E:\work\chap7\7-1-2>
```

레퍼런스 타입

01. 레퍼런스 타입

● null 참조값

- [예제 7-5] 레퍼런스 변수를 null 참조값과 비교해서 처리하는 프로그램

```
1  class RefTypeExample5 {  
2      public static void main(String args[]) {  
3          Point obj = null;  
4          if (obj == null) {  
5              System.out.println("obj 변수가 가리키는 객체가 없습니다");  
6              return;  
7          }  
8          System.out.println("x = " + obj.x);  
9          System.out.println("y = " + obj.y);  
10     }  
11 }
```



```
명령 프롬프트  
E:\work\chap7\7-1-2>java RefTypeExample5  
obj 변수가 가리키는 객체가 없습니다  
E:\work\chap7\7-1-2>
```

레퍼런스 타입

01. 레퍼런스 타입

변수의 타입과 객체의 타입

- [예제 7-6] 상속 관계를 갖는 클래스들

```
class Account {  
    ...  
}
```

클래스 변수에는 서브클래스 객체도 대입 가능합니다.

상속

상속

```
class CheckingAccount extends Account {  
    ...  
}
```

```
class CreditLineAccount extends Account {  
    ...  
}
```

상속

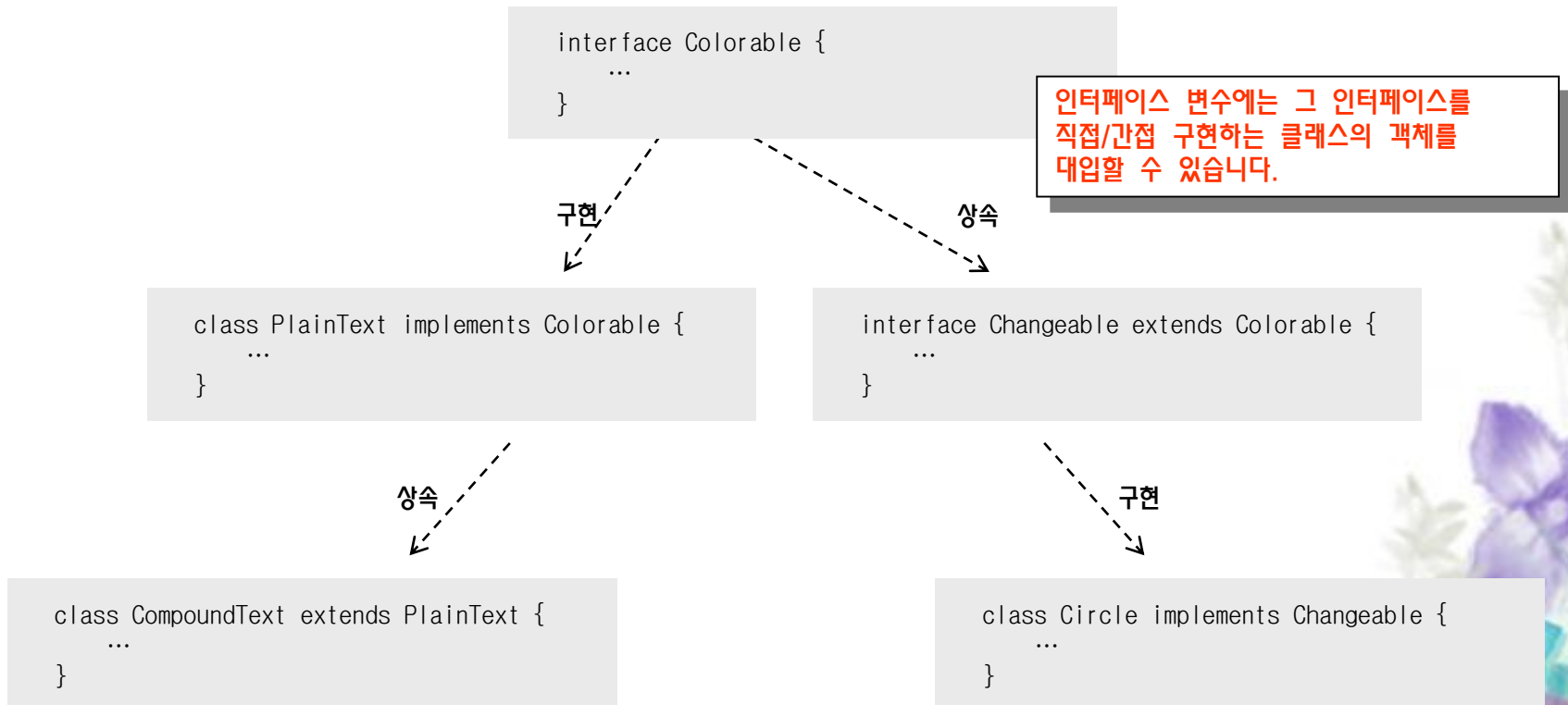
```
class CheckingTrafficCardAccount extends CheckingAccount {  
    ...  
}
```

레퍼런스 타입

01. 레퍼런스 타입

변수의 타입과 객체의 타입

- [예제 7-7] 구현/상속 관계를 갖는 인터페이스와 클래스들



레퍼런스 타입

● 변수에 의해 제한되는 객체의 사용 방법

- [예제 7-8] Account 클래스와 CheckingAccount 클래스

은행 계좌 클래스

```
1 class Account {  
2     String accountNo;  
3     String ownerName;  
4     int balance;  
5     Account(String accountNo, String ownerName, int balance) {  
6         this.accountNo = accountNo;  
7         this.ownerName = ownerName;  
8     }  
9 }
```

상속

직불 계좌 클래스

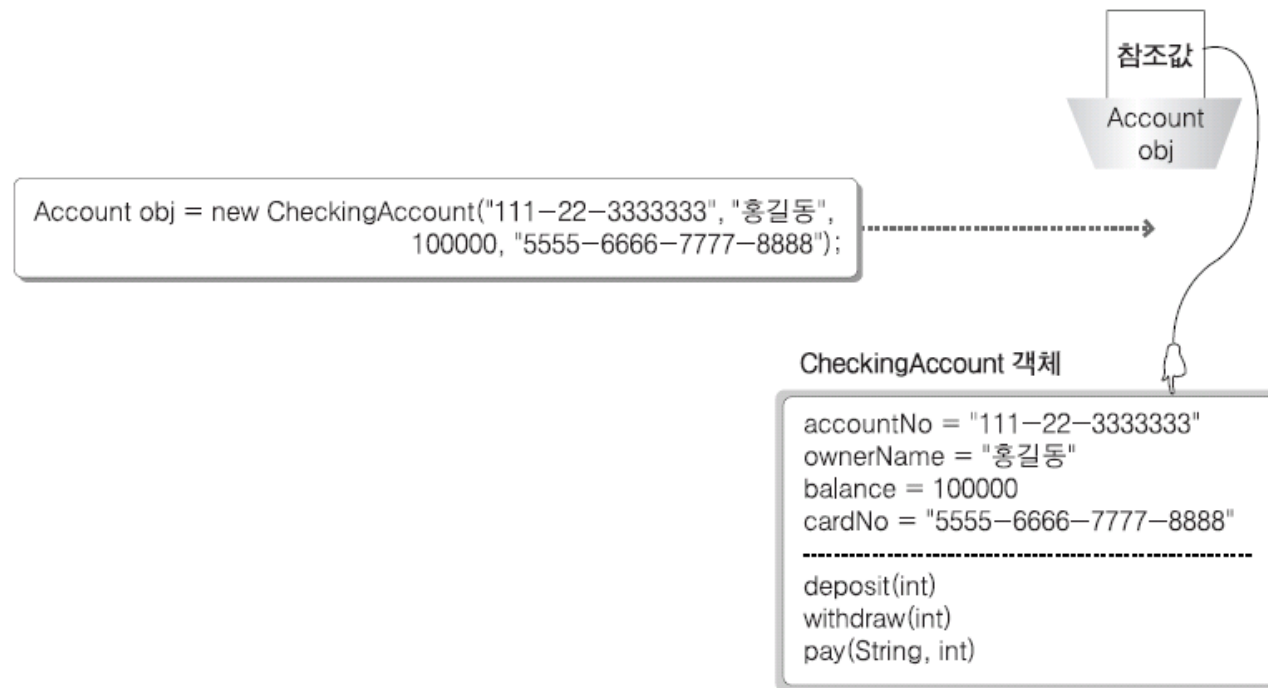
```
1 class CheckingAccount extends Account {  
2     String cardNo;  
3     CheckingAccount(String accountNo, String ownerName, int balance, String cardNo) {  
4         super(accountNo, ownerName, balance);  
5         this.cardNo = cardNo;  
6     }  
7     int pay(String cardNo, int amount) throws Exception {  
8         if (!cardNo.equals(this.cardNo) || (balance < amount))  
9             throw new Exception("지불이 불가능합니다.");  
10        return withdraw(amount);  
11    }  
12 }
```

레퍼런스 타입

01. 레퍼런스 타입

변수에 의해 제한되는 객체의 사용 방법

- 다른 타입의 객체를 가리키는 클래스 변수



레퍼런스 타입

● 변수에 의해 제한되는 객체의 사용 방법

- [예제 7-9] 변수 타입에 속하지 않는 필드와 메소드를 사용하는 잘못된 프로그램

```
1 class RefTypeExample6 {
2     public static void main(String args[]) {
3         Account obj = new CheckingAccount("111-22-33333333",
4             "홍길동", 10, "4444-5555-6666-7777");
5
6         try {
7             int amount = obj.pay("4444-5555-6666-7777", 47000);
8             System.out.println("인출액: " + amount);
9             System.out.println("카드번호: " + obj.cardNo);
10        }
11    }
12 }
13 }
```

Account 클래스에 없는
메소드와 필드를
사용하는 잘못된 명령문

명령 프롬프트

E:\work\chap7\7-1-3\example1>javac Account.java

E:\work\chap7\7-1-3\example1>javac CheckingAccount.java

E:\work\chap7\7-1-3\example1>javac RefTypeExample6.java

RefTypeExample6.java:5: cannot find symbol

symbol : method pay(java.lang.String,int)

location: class Account

int amount = obj.pay("4444-5555-6666-7777", 47000);
^

RefTypeExample6.java:7: cannot find symbol

symbol : variable cardNo

location: class Account

System.out.println("카드번호: " + obj.cardNo);
^

2 errors

레퍼런스 타입

● 변수에 의해 제한되는 객체의 사용 방법

- [예제 7-10] 슈퍼클래스 변수 값을 서브클래스 변수에 대입하는 잘못된 프로그램

```
1  class RefTypeExample7 {
2      public static void main(String args[]) {
3          Account obj1 = new CheckingAccount("111-22-33333333",
4              "홍길동", 100000, "5555-6666-7777-8888");
5          CheckingAccount obj2 = obj1;    // 잘못된 대입문
6          try {
7              int amount = obj2.pay("5555-6666-7777-8888", 47000);
8              System.out.println("인출액: " + amount);
9              System.out.println("카드번호: " + obj2.cardNo);
10         }
11         catch (Exception e) {
12             System.out.println(e.getMessage());
13         }
14     }
```

```
C:\ 명령 프롬프트
E:\work\chap7\7-1-3\example2>javac Account.java
E:\work\chap7\7-1-3\example2>javac CheckingAccount.java
E:\work\chap7\7-1-3\example2>javac RefTypeExample7.java
RefTypeExample7.java:4: incompatible types
found   : Account
required: CheckingAccount
    CheckingAccount obj2 = obj1;
                        ^
1 error
E:\work\chap7\7-1-3\example2>_
```

레퍼런스 타입

01. 레퍼런스 타입

서브타입

• 서브타입(subtype)이란?

- 어떤 타입의 객체가 사용되어야 할 위치에 다른 타입의 객체를 대신 끼워넣을 수 있으면 후자를 전자의 서브타입이라고 함

• 클래스와 인터페이스의 서브타입

타입	서브타입
A 클래스	A의 서브클래스
B 인터페이스	B의 서브인터페이스와 B를 직접 또는 간접으로 구현하는 클래스

레퍼런스 타입

01. 레퍼런스 타입

서브타입과 대입 연산

- 서브타입은 대입 연산의 성립 여부 결정의 기준이 됨

`obj2 = obj1;`

- 이 연산이 성립하기 위해서는 다음 둘 중 한 조건이 만족해야 함
- `obj1`의 타입 = `obj2`의 타입
- `obj1`의 타입 = `obj2`의 서브타입

하지만 때로는 이 중 한 조건이
만족하더라도 캐스트 연산이 필요

01. 레퍼런스 타입

캐스트 연산자

- ## • 레퍼런스 타입에 캐스트 연산자를 사용하는 방법

```
obj2 = (CheckingAccount) obj1;
```

CheckingAccount
타입의 변수

캐스트 연산자

Account 타입의 변수

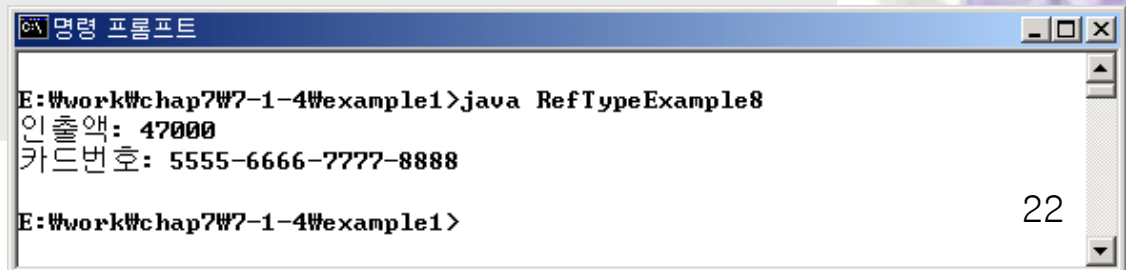
- * 문법적으로는 프리미티브 타입의 캐스트 연산과 동일

레퍼런스 타입

캐스트 연산자

- [예제 7-11] 슈퍼클래스 변수 값을 캐스트해서 서브클래스 변수에 대입하는 프로그램

```
1  class RefTypeExample8 {
2      public static void main(String args[]) {
3          Account obj1 = new CheckingAccount("111-22-33333333",
4                                              "홍길동", 100000, "5555-6666-7777-8888");
5          CheckingAccount obj2 = (CheckingAccount) obj1;
6          try {
7              int amount = obj2.pay("5555-6666-7777-8888", 47000);
8              System.out.println("인출액: " + amount);
9              System.out.println("카드번호: " + obj2.cardNo);
10         }
11         catch (Exception e) {
12             System.out.println(e.getMessage());
13         }
14     }
```



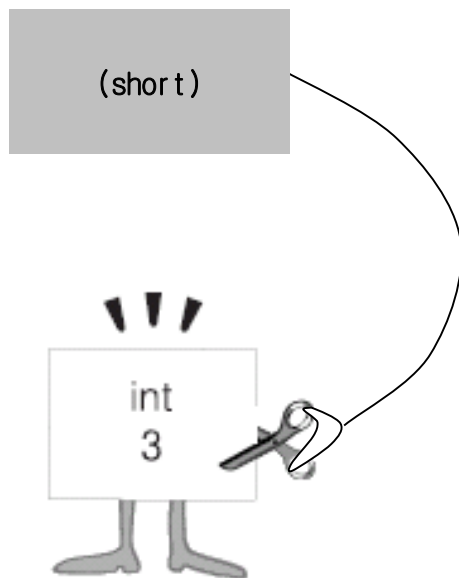
```
명령 프롬프트
E:\work\chap7\7-1-4\example1>java RefTypeExample8
인출액: 47000
카드번호: 5555-6666-7777-8888
E:\work\chap7\7-1-4\example1>
```

레퍼런스 타입

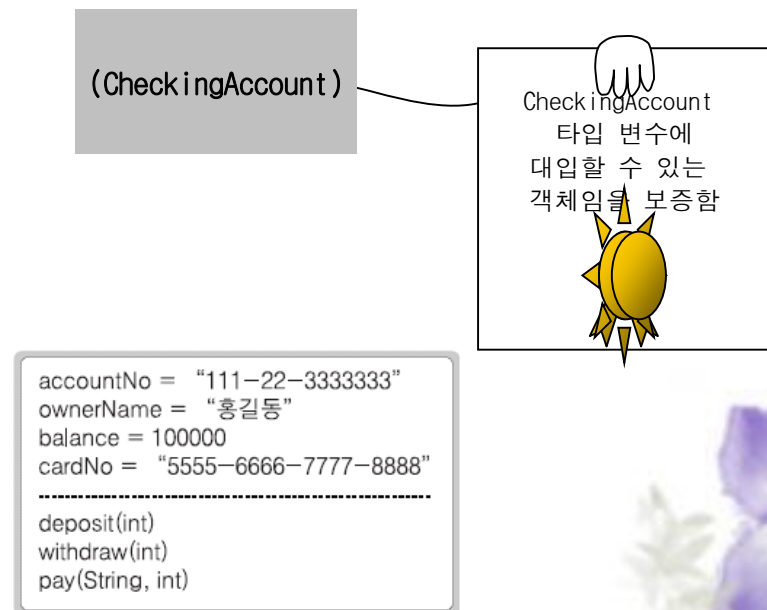
01. 레퍼런스 타입

캐스트 연산자

- 프리미티브 타입과 레퍼런스 타입의 캐스트 연산 차이



캐스트 연산자를 프리미티브 타입에 대해 사용하면 데이터 타입이 바뀝니다.



캐스트 연산자를 레퍼런스 타입에 대해 사용하면 객체의 타입을 보증하는 역할만 합니다.

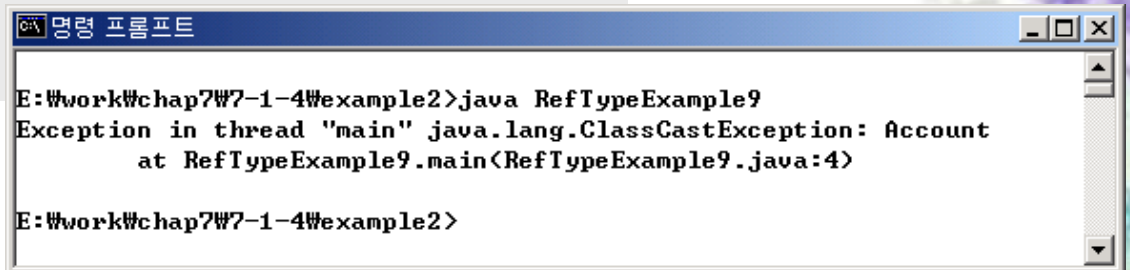
레퍼런스 타입

캐스트 연산자

- [예제 7-12] 객체를 그와 맞지 않는 타입으로 캐스트하는 잘못된 예

```
1 class RefTypeExample9 {
2     public static void main(String args[]) {
3         Account obj1 = new Account("111-22-33333333", " 홍 길 동 ",
4         100000);
5         CheckingAccount obj2 = (CheckingAccount) obj1;
6         try {
7             int amount = obj2.pay("5555-6666-7777-8888", 47000);
8             System.out.println("인출액: " + amount);
9             System.out.println("카드번호: " + obj2.cardNo);
10        }
11        catch (Exception e) {
12            System.out.println(e.getMessage());
13        }
14    }
}
```

----- 잘못된 캐스트 연산



```
명령 프롬프트
E:\work\chap7\7-1-4\example2>java RefTypeExample9
Exception in thread "main" java.lang.ClassCastException: Account
    at RefTypeExample9.main<RefTypeExample9.java:4>

E:\work\chap7\7-1-4\example2>
```


레퍼런스 타입

01. 레퍼런스 타입

instanceof 연산자

- `instanceof` : 캐스트 연산 가능성을 검사하는 연산자, 자바 키워드
- `instanceof` 연산자의 사용 방법



레퍼런스 타입

instanceof 연산자

- [예제 7-13] 객체를 캐스트하기 전에 instanceof 연산자로 검사하는 예

```
1 class RefTypeExample10 {
2     public static void main(String args[]) {
3         Account obj = new Account("111-22-33333333", "홍길동", 100000);
4         if (obj instanceof CheckingAccount)
5             pay((CheckingAccount) obj);
6         else
7             System.out.println("캐스트할 수 없는 타입입니다.");
8     }
9     static void pay(CheckingAccount obj) {
10        try {
11            int amount = obj.pay("5555-6666-7777-8888", 47000);
12            System.out.println("인출액: " + amount);
13            System.out.println("카드번호: " + obj.cardNo);
14        }
15        catch (Exception e) {
16            System.out.println(e.getMessage());
17        }
18    }
19 }
```

} obj를 CheckingAccount로
캐스트할 수 있을 때만
pay 메소드 호출

```
명령 프롬프트
E:\work\chap7\7-1-4\example3>java RefTypeExample10
캐스트할 수 없는 타입입니다.

E:\work\chap7\7-1-4\example3>
```

02. 열거 타입

열거 타입의 필요성

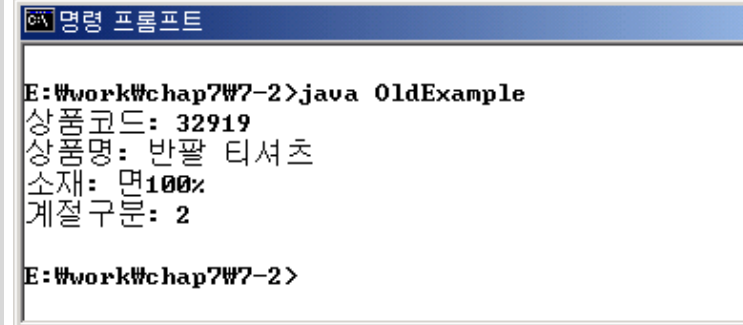
- 한정된 수의 값을 표현하기 위해 필요
 - [예1] 월, 화, 수, 목, 금, 토, 일
 - [예2] 봄, 여름, 가을, 겨울
- 자바에서 열거 타입이 있기 전에는;
 - 코드화된 정수나 문자를 이용하여 해당 값을 표현했음

레퍼런스 타입

의류 정보 클래스

```
1 class ClothingInfo {
2     String code;
3     String name;
4     String material;
5     int season;
6     static final int SPRING = 1;
7     static final int SUMMER = 2;
8     static final int FALL = 3;
9     static final int WINTER = 4;
10    ClothingInfo(String code, String name, String material, int season) {
11        this.code = code;
12        this.name = name;
13        this.material = material;
14        this.season = season;
15    }
16 }
```

사계절을 표현하는 상수 필드



```
E:\work\chap7\7-2>java OldExample
상품코드: 32919
상품명: 반팔 티셔츠
소재: 면100%
계절구분: 2

E:\work\chap7\7-2>
```

의류 정보 클래스를 사용하는 프로그램

```
1 class OldExample {
2     public static void main(String args[]) {
3         ClothingInfo obj = new ClothingInfo("32919", "반팔 티셔츠", "면100%", ClothingInfo.SUMMER);
4         System.out.println("상품코드: " + obj.code);
5         System.out.println("상품명: " + obj.name);
6         System.out.println("소재: " + obj.material);
7         System.out.println("계절구분: " + obj.season);
8     }
9 }
```

ClothingInfo 클래스에
선언된 상수 필드를 사용

레퍼런스 타입

02. 열거 타입

열거 타입의 선언과 이용

• 열거 타입의 정의

- 열거 타입에 속하는 값들을 파악하여 정의

[열거 타입] 사계절

[열거 값]

봄

여름

가을

겨울

레퍼런스 타입

02. 열거 타입

열거 타입의 선언과 이용

• 열거 타입의 선언

- 제일 먼저 해야할 일은 열거 타입 이름과 열거 값 이름 정하기

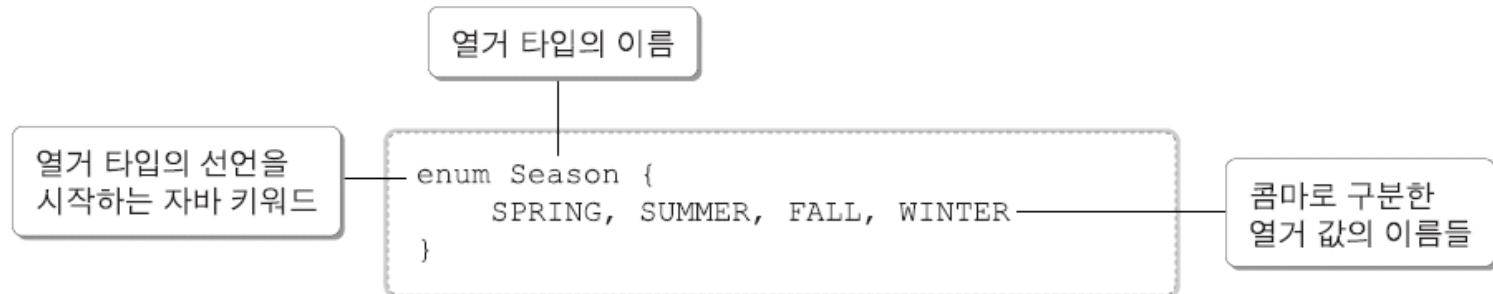
[열거 타입] 사계절>	Season	열거 타입의 이름과
[열거 값]			
봄>	SPRING	열거 타입에 속하는 값의이름들을 정해야 합니다.
여름>	SUMMER	
가을>	FALL	
겨울>	WINTER	

레퍼런스 타입

02. 열거 타입

열거 타입의 선언과 이용

• 열거 타입의 선언



레퍼런스 타입

02. 열거 타입

열거 타입의 선언과 이용

• 열거 타입의 변수 선언

Season season;

↑ ↑

열거 타입 변수 이름

레퍼런스 타입

02. 열거 타입

열거 타입의 선언과 이용

• 열거 타입의 변수의 사용

```
season = Season.SPRING;
```

↑
열거 타입 이름

↑
열거 상수 이름

- * 열거 타입 변수에는 그 열거 타입에 속하는 열거 값(열거 상수)만 대입가능

레퍼런스 타입

열거 타입의 선언과 이용

- [예제 7-15] 사계절을 열거 타입으로 선언해서 사용하는 예

사계절 열거 타입

```
1 enum Season {  
2     SPRING, SUMMER, FALL, WINTER  
3 }
```

의류 정보 클래스

```
1 class ClothingInfo {  
2     String code;  
3     String name;  
4     String material;  
5     Season season;----- 열거 타입의 필드  
6     ClothingInfo(String code, String name, String material, Season season) {  
7         this.code = code;  
8         this.name = name;  
9         this.material = material;  
10        this.season = season;  
11    }  
12 }
```

열거 타입의
파라미터 변수

의류 정보 클래스를 사용하는 프로그램

```
1 class NewExample {  
2     public static void main(String args[]) {  
3         ClothingInfo obj = new ClothingInfo("32919", "반팔 티셔츠", "면100%", Season.SUMMER);  
4         System.out.println("상품코드: " + obj.code);  
5         System.out.println("상품명: " + obj.name);  
6         System.out.println("소재: " + obj.material);  
7         System.out.println("계절구분: " + obj.season);  
8     }  
9 }
```

열거 상수

명령 프롬프트

```
E:\work\chap7\7-2-1\example1>java NewExample  
상품코드: 32919  
상품명: 반팔 티셔츠  
소재: 면100%  
계절구분: SUMMER  
E:\work\chap7\7-2-1\example1>
```

레퍼런스 타입

02. 열거 타입

클래스에 종속된 열거 타입

- 클래스/인터페이스 안에 선언된 열거 타입은 그 클래스/인터페이스에 종속됨
- 클래스/인터페이스에 종속된 열거 타입의 사용 방법

```
season = ClothingInfo.Season.SPRING;
```

열거 타입이 속하는
클래스 이름

열거 타입 이름

열거 상수 이름

레퍼런스 타입

클래스에 종속된 열거 타입

- [예제 7-16] 사계절 열거 타입을 클래스 안에 선언해서 사용하는 예

의류 정보 클래스

```
1 class ClothingInfo {
2     enum Season {
3         SPRING, SUMMER, FALL, WINTER } --- 사계절 열거 타입
4     }
5     String code;
6     String name;
7     String material;
8     Season season; ----- 열거 타입 필드
9     ClothingInfo(String code, String name, String material, Season season) {
10         this.code = code;
11         this.name = name;
12         this.material = material;
13         this.season = season;
14     }
15 }
```

의류 정보 클래스를 사용하는 프로그램

```
1 class NewExample2 {
2     public static void main(String args[]) {
3         ClothingInfo obj = new ClothingInfo("32919", "반팔 티셔츠", "면100%", ClothingInfo.Season.SUMMER);
4         System.out.println("상품코드: " + obj.code);
5         System.out.println("상품명: " + obj.name);
6         System.out.println("소재: " + obj.material);
7         System.out.println("계절구분: " + obj.season);
8     }
9 }
```

명령 프롬프트

```
E:\work\chap7\7-2-1\example2>java NewExample2
상품코드: 32919
상품명: 반팔 티셔츠
소재: 면100%
계절구분: SUMMER

E:\work\chap7\7-2-1\example2>
```

열거 상수

레퍼런스 타입

02. 열거 타입

values 메소드와 valueOf 메소드

- 열거 타입은 컴파일하고 나면 내부적으로 클래스가 됨
 - -> 그 클래스에는 `values`와 `valueOf`라는 정적 메소드가 자동으로 추가됨
- `values` 메소드 : 모든 열거 상수를 리턴하는 메소드
- `valueOf` 메소드 : 주어진 문자열에 해당하는 열거 상수를 리턴하는 메소드

레퍼런스 타입

02. 열거 타입

values 메소드와 valueOf 메소드

- [예제 7-17] 요일을 표현하는 열거 타입

```
1    enum Day {  
2        MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY  
3    }
```


레퍼런스 타입

02. 열거 타입

values 메소드와 valueOf 메소드

- [예제 7-18] 열거 타입에 대해 values 메소드를 호출하는 예

```
1  class EnumExample1 {  
2      public static void main(String args[]) {  
3          Day days[] = Day.values();  
4          for (Day day : days)  
5              System.out.println(day);  
6      }  
7  }
```



```
C:\>명령 프롬프트  
E:\work\chap7\7-2>java EnumExample1  
MONDAY  
TUESDAY  
WEDNESDAY  
THURSDAY  
FRIDAY  
SATURDAY  
SUNDAY  
E:\work\chap7\7-2>
```

레퍼런스 타입

02. 열거 타입

values 메소드와 valueOf 메소드

- [예제 7-19] 열거 타입에 대해 valueOf 메소드를 호출하는 예

```
1  class EnumExample2 {  
2      public static void main(String args[]) {  
3          printDay("MONDAY");  
4          printDay("WEDNESDAY");  
5          printDay("FRIDAY");  
6      }  
7      static void printDay(String name) {  
8          Day day = Day.valueOf(name);  
9          System.out.println(day);  
10     }  
11 }
```



```
명령 프롬프트  
E:\work\chap7\7-2>java EnumExample2  
MONDAY  
WEDNESDAY  
FRIDAY  
E:\work\chap7\7-2>
```


레퍼런스 타입

02. 열거 타입

열거 상수를 다른 값과 연관짓기

- 다음의 열거 상수들을 "봄", "여름", "가을", "겨울" 이라는 이름과 연관시킬 수 있음

```
enum Season {  
    SPRING, SUMMER, FALL, WINTER  
}
```

레퍼런스 타입

02. 열거 타입

열거 상수를 다른 값과 연관짓기

- 1단계 : 각각의 열거 상수 뒤에 연관 값을 기술

```
enum Season {  
    SPRING("봄"), SUMMER("여름"), FALL("가을"), WINTER("겨울") ;  
      
}  
    
```

필드, 생성자, 메서드를
선언할 수 있는 위치

열거 상수와 다른 구성요소를
구분하는 세미콜론

레퍼런스 타입

02. 열거 타입

열거 상수를 다른 값과 연관짓기

- 2단계 : 연관 값을 저장할 필드 선언

```
enum Season {  
    SPRING("봄"), SUMMER("여름"), FALL("가을"), WINTER("겨울") ;  
    final private String name;  
}
```

반드시 써야하는 키워드

레퍼런스 타입

02. 열거 타입

열거 상수를 다른 값과 연관짓기

• 3단계 : 생성자의 선언

```
enum Season {  
    SPRING("봄"), SUMMER("여름"), FALL("가을"), WINTER("겨울") ;  
    final private String name;  
    Season(String name) {  
        this.name = name;  
    }  
}
```

이런 값이 생성자 파라미터로 넘어옵니다.

그 파라미터 값으로
필드를 초기화 해야 합니다.

레퍼런스 타입

02. 열거 타입

열거 상수를 다른 값과 연관짓기

•• 4단계 : 메소드의 선언

```
enum Season {  
    SPRING("봄"), SUMMER("여름"), FALL("가을"), WINTER("겨울") ;  
    final private String name;  
    Season(String name) {  
        this.name = name;  
    }  
    String value() {  
        return name;  
    }  
}
```

열거 상수에 연관된 값을
리턴하는 메소드

레퍼런스 타입

열거 상수를 다른 값과 연관짓기

- [예제 7-20] 열거 상수에 연관된 값을 이용하는 프로그램

요일 열거 타입

```
1    enum Season {  
2        SPRING("봄"), SUMMER("여름"), FALL("가을"), WINTER("겨울") ;  
3        final private String name;  
4        Season(String name) {  
5            this.name = name;  
6        }  
7        String value() {  
8            return name;  
9        }  
10    }
```

요일 열거 타입을 사용하는 프로그램

```
1    class EnumExample3 {  
2        public static void main(String args[]) {  
3            printSeason(Season.SPRING);  
4            printSeason(Season.SUMMER);  
5            printSeason(Season.FALL);  
6            printSeason(Season.WINTER);  
7        }  
8        static void printSeason(Season season) {  
9            System.out.println(season.value());  
10       }  
11    }
```

메소드 호출

명 프롬프트

ork\chap7\7-2-3>java EnumExample3

ork\chap7\7-2-3>

어제

많은

일을

이루고

싶습니다~