제4장 : 제네릭, 열거형, 이너클래스

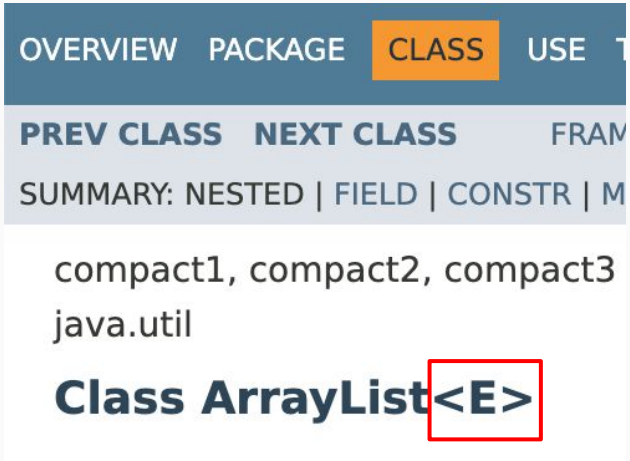
타입이 없을때의 문제점

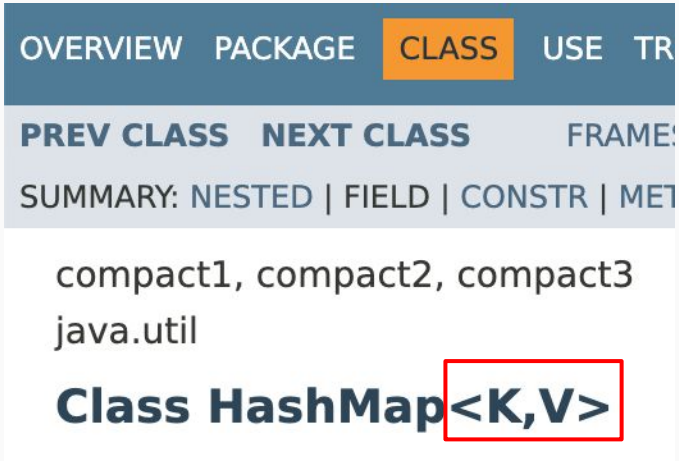
1) 런타임 에러가 나기 쉽다.
2) IDE가 컴파일 에러를 미리 찾을 수 없다.

제네릭 (Generic)

타입을 나중에 원하는 형태로 정의할 수 있음

타입 안전(type safe) 효과





제네릭을 사용하지 않은 Pocket 클래스 (Ver. 1)


```
public class Pocket {
  private Object data;
  public void put(Object data) {
    this.data = data;
  }
  public Object get() {
    return this.data;
  }
}
```

제네릭을 사용한 Pocket 클래스 (Ver. 2)


```
public class Pocket<E> {
  private E data;
  public void put(E data) {
    this.data = data;
  }
  public E get() {
    return this.data;
  }
}
```

이용 가능한 타입에 제약을 추가한 Pocket 클래스 (Ver. 3)


```
public class Pocket<E extends Character> {
  private E data;
  public void put(E data) {
    this.data = data;
  }
  public E get() {
    return this.data;
  }
}
```

열거형 (enum)

정해 둔 값만 넣어둘 수 있는 타입

로그인 상태를 나타내는 상수를 정의한 예


```
public class AuthState {
  public static final int AUTHENTICATED = 1;
  public static final int UNAUTHENTICATED = 2;
  public static final int AUTHENTICATING = 3;
}
```

```
int authState = AuthState.UNAUTHENTICATED;

if (authState == AuthState.AUTHENTICATED) {
  System.out.println("로그인 됨");
} else if (authState == AuthState.UNAUTHENTICATED) {
  System.out.println("로그아웃 됨");
} else {
  System.out.println("로그인 중");
}
```

Quiz 안전한 코드인가?

enum을 정의하여 세 가지 상태로 제약을 건 코드


```
enum AuthState {
  AUTHENTICATED, UNAUTHENTICATED, AUTHENTICATING;
}
```

```
AuthState authState = AuthState.UNAUTHENTICATED;

if (authState == AuthState.AUTHENTICATED) {
  System.out.println("로그인 됨");
} else if (authState == AuthState.UNAUTHENTICATED) {
  System.out.println("로그아웃 됨");
} else {
  System.out.println("로그인 중");
}
```

이너클래스 (Inner class)

클래스 안에 정의하는 클래스

static 을 붙이고 안 붙이고의 차이


```
class Outer {
    // 생략...

    class Inner {
        // 생략...
    }
}
```

```
class Outer {
    // 생략...

    static class Inner {
        // 생략...
    }
}
```

new Outer.Inner() 의 사용 가능 여부

익명클래스 (anonymous class)


```
Pocket<Object> Pocket = new Pocket<>();
Pocket.put(new Object() {
  String field;
  void method() {

  }
});
```

메서드 호출 도중에 갑자기 필요한 클래스를 정의하여 사용하는 방법

연습문제 4-1

다음 조건을 만족하는 금고인 StrongBox 클래스를 정의하시오.

1) 금고 클래스에 담는 인스턴스의 타입은 미정
2) 금고에는 1개의 인스턴스를 담을 수 있음
3) put() 메서드로 인스턴스를 저장하고 get() 메서드로 인스턴스를 얻을 있음
4) get() 으로 얻을 때는 별도의 타입 캐스팅을 사용하지 않아도 됨

연습문제 4-2

연습문제 4-1에서 작성한 StrongBox 클래스에 열쇠의 종류를 나타내는 열거형 KeyType을 정의하고, 다음 내용을 반영하여 StrongBox 클래스를 수정하시오.

- 열쇠의 종류를 나타내는 필드 변수
- 열쇠의 종류를 받는 생성자

단, 열쇠의 종류는 다음 4종류로 한정한다. 각 열쇠는 사용횟수의 한도가 정해져 있다.

1) PADLOCK (1,024회)
2) BUTTON (10,000회)
3) DIAL (30,000회)
4) FINGER (1,000,000회)

금고에서 get() 메서드를 호출할 때 마다 사용횟수를 카운트하고 각 열쇠의 사용횟수에 도달하기 전에는 null을 리턴한다.