

◆ 5.0에 추가된 문법

- ◆ 제네릭 타입 (Generic Type)
- ◆ 개선된 루프
- ◆ 오토박싱/언박싱
- ◆ Static import
- ◆ Varargs

◆ 제네릭 타입(Generic Type)

- ◆ Java.util.*에 있는 Collection API에 적용
- ◆ 5.0이전 버전에서는 Collection API에서 Collection 객체를 Object 클래스에 저장하기 때문에 컴파일 타임에 발생하는 Type 오류를 확인할 수 없다.
- ◆ 이러한 문제는 ClassCastException이 발생하여 확인이 가능하므로 실행시 때까지 기다려야 한다.

◆ 제네릭타입(Generic Type)

- ◆ 하지만 5.0에서 이런 번거로움을 피하기 위해서 일반화된 API를 사용하려면 컴파일 할 때 단지 <> 사이에 사용할 자료타입을 선언해주기만 하면 캐스팅을 더 이상 필요없다
- ◆ 따라서 사용되는 Collection에 삽입되는 객체의 자료Type이 잘못대입되면 런타임이 아닌 컴파일 할 때 확인할 수 있다.

예제 : VectorExam.java

◆ 개선된 루프

- ◆ 컴파일러는 필요한 루프코드를 생성하게 되고 Generic타입을 함께 사용하게 되면 더 이상 일일이 자동형변환을 해주지 않아도 되므로 프로그래밍을 하는데 편리해진다.

예제 : ForArr.java

◆ 개선된 루프

[이전버젼]

```
String [] str={"A","B","C"}  
for(int i=0; i<str.length; i++){  
    System.out.println(str[i]);  
}
```

[5.0 버젼]

```
String [] str={"A","B","C"}  
for(String n : str){  
    System.out.println(n);  
}
```

◆ 오토박싱/언박싱

- ◆ 기본형에 상응되는 Wapper 클래스 타입간의 자료 변환작업이 불필요할 정도로 많은 코드를 요구하는 경우가 있다.
- ◆ 5.0에서는 int와 Integer간의 변환작업을 컴파일러가 맡아서 처리하므로 불필요함을 크게 줄었다.
- ◆ 5.0에서 이런 귀찮은 변환작업을 자동으로 제공하는 이를 오토박싱/언박싱이라고 한다.

◆ 오토박싱/언박싱

[이전버젼]

```
Integer it=new Integer(50);  
int i=it.intValue();
```

[5.0 버젼]

```
Integer it=new Integer(50);  
int i=it;
```

예제 : AutoBoxing_UnBoxing.java

◆ static import

- ◆ Static import 기능은 import static의 형태로 사용되는 이를 통해 class로 부터 상속 받지 않고 또는 해당 class를 일일이 명시하지 않고도 static상수들에 접근할 수 있다.
- ◆ import static java.awt.FlowLayout.*;
- ◆ import static java.lang.System.out;

예제 : Staticimport.java

◆ Varargs (Variable Arguments)

- ◆ 5.0이전 버전에서는 특정 메소드를 정의 할때 인자의 타입과 수를 정해놓고 호출할 때 일치하지 않으면 오류발생한다.
- ◆ 이로 인해 개발자는 오버로딩 또는 배열 객체의 인자로 지정하였다. 이제 varargs 기능을 사용하여 인자로 받은 메서드에 ...이라고 명시하면 메서드를 수행하는데 필요한 인자의 수를 유연하게 구현 할수 있다.

◆ Varargs

[5.0 버전]

```
Public void test(Object ... ar){  
    for(int i=0; i< ar.length ; i++){  
        System.out.println(ar[i]);  
    }  
}
```

-실행

```
Test("java");  
Test("var","aa");  
Test("JAVA","잘하자","파이팅!");
```

예제 :Varargs_Overloading.java

◆ printf 메소드

```
System.out.printf("출력할 포맷지정", 인수, 인수 ,...);
```

◆ 포맷명세자(%문자와 조합 사용)

변환	설명	변환	설명
%d	정수10진법	%tY	년(4자리)
%o	정수8진법	%ty	년(2자리)
%x	정수 16진법	%tm	월
%f	실수형	%td	일
%s, %S	문자열	%tH,%tM,%tS	시,분,초

◆ printf 메소드

◆ 아규먼트인덱스(argument index)

- 아라비아 숫자와 \$문자로 구성
 - Ex) System.out.printf("%2\$d년 %3\$d월 %1\$d일", 1, 1919, 3);
결과=> 1919년 3월 1일 출력
- 자리 수 표현
 - %5d => 5자리 문자표현
 - %5.2f =>전체 5자리에 소수점 2자리까지 표현

예제 : PrintfExam.java