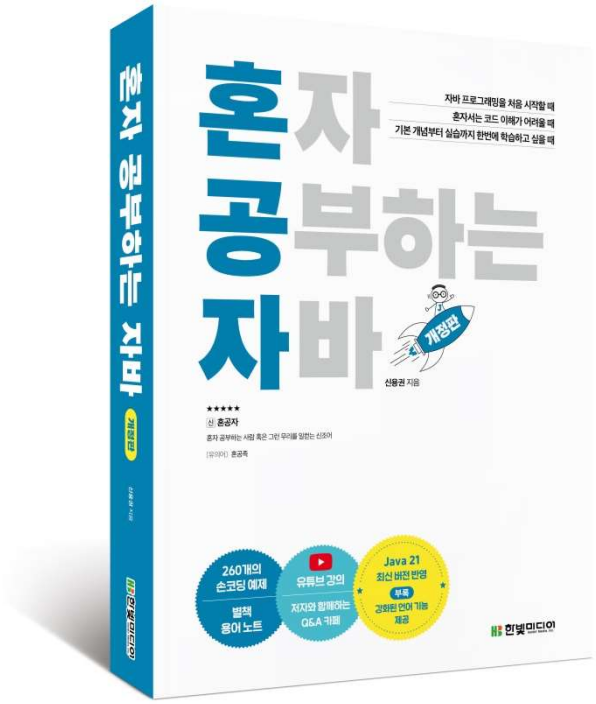


Generic

혼자 공부하는 자바 (신용권 저)

한빛미디어 Hanbit Media, Inc.

제네릭 타입

❖ 필요성

- 자바는 다양한 종류의 객체를 관리하는 컬렉션이라는 자료구조를 제공
- 초기에는 Object 타입의 컬렉션을 사용
- Object 타입의 컬렉션은 실행하기 전에는 어떤 객체인지?

❖ 소개

- 제네릭 타입의 의미

  하나의 코드를 다양한 타입의 객체에 재사용하는 객체 지향 기법

  클래스, 인터페이스, 메서드를 정의할 때 타입을 변수로 사용

- 제네릭 타입의 장점

  컴파일할 때 타입을 점검하기 때문에 실행 도중 발생할 오류 사전 방지

  불필요한 타입 변환이 없어 프로그램 성능 향상

❖ 제네릭 타입 선언

```
class 클래스이름<타입매개변수> {
    필드;
    메서드;
}
```

메서드나 필드에 필요한 타입을 타입 매개변수로 나타낸다.

- 타입 매개변수는 객체를 생성할 때 구체적인 타입으로 대체

- 전형적인 타입 매개변수

타입 매개변수	설명
E	원소(Element)
K	키(Key)
N	숫자(Number)
T	타입(Type)
V	값(Value)

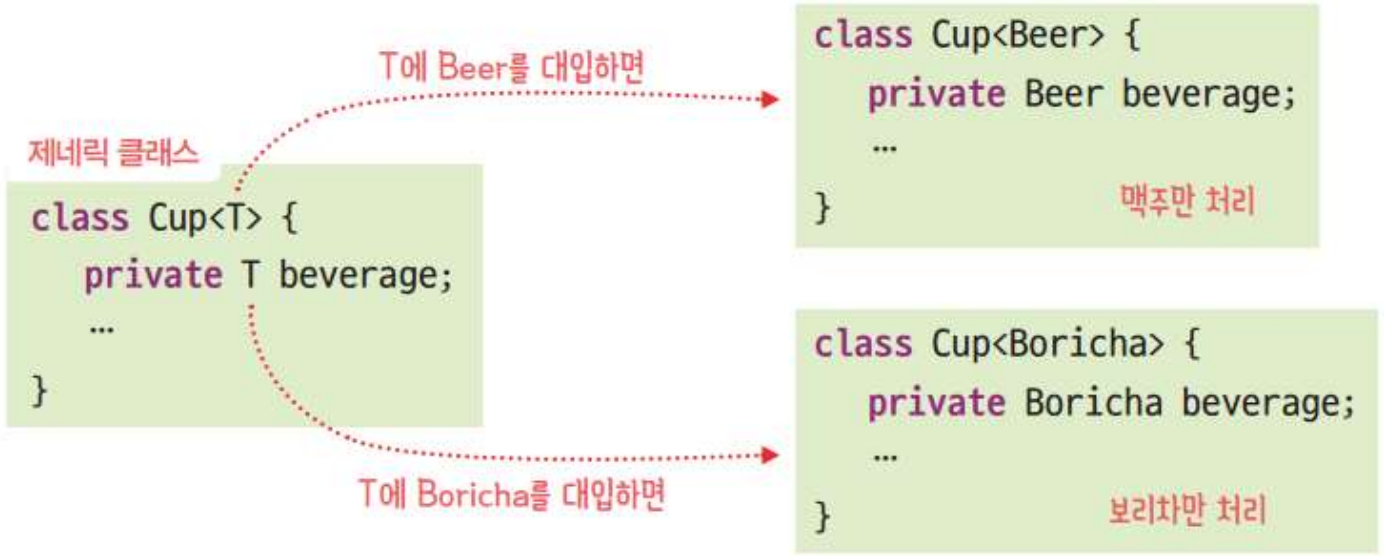
❖ 제네릭 객체 생성

```
제네릭클래스 <적용할타입> 변수 = new 제네릭클래스<적용할타입>();
```

생략할 수 있다.

- <적용할타입>에서 적용할 타입을 생략할 경우 <>를 다이어몬드 연산자라고 함

- 제네릭 클래스의 적용



```java
public class Beverage {
}
```

```java
public class Boricha extends Beverage {
}
```

```java
public class Beer extends Beverage {
}
```

```java
public class Cup<T> {
    private T beverage;

    public T getBeverage() {
        return beverage;
    }

    public void setBeverage(T beverage) {
        this.beverage = beverage;
    }
}
```

```java
public class GenericClass3Demo {
    public static void main(String[] args) {
        Cup c = new Cup();

        c.setBeverage(new Beer());

//        Beer beer = c.getBeverage();
        Beer beer = (Beer) c.getBeverage();
    }
}
```

❖ 제네릭 타입 응용

❖ Raw 타입의 필요성 및 의미

- 이전 버전과 호환성을 유지하려고 Raw 타입을 지원
- 제네릭 클래스를 Raw 타입으로 사용하면 타입 매개변수를 쓰지 않기 때문에 Object 타입이 적용

❖ 제네릭 타입의 상속 관계

- 예를 들어

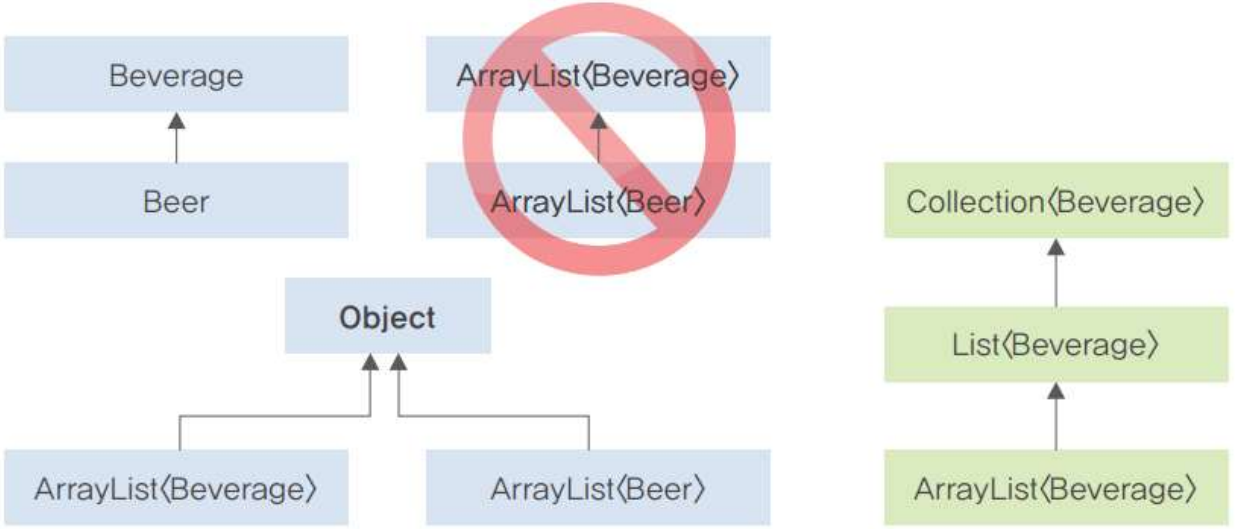
```
ArrayList<Beverage> list = new ArrayList<>();
list.add(new Beer());                    // OK
list.add(new Boricha());                 // OK
```

- 그러나 ArrayList<Beverage> 타입과 ArrayList<Beer>의 경우는 상속 관계가 없다.

제네릭 상속 및 타입 한정

❖ 제네릭의 제약

- 기초 타입을 제네릭 인수로 사용 불가
- 정적 제네릭 타입 금지
- 제네릭 타입의 인스턴스화 금지. 즉, new T( ) 등 금지
- 제네릭 타입의 배열 생성 금지
- 실행 중에 제네릭 타입 점검 금지. 예를 들어, a instanceof ArrayList<String>
- 제네릭 클래스의 객체는 예외로 던지거나 잡을 수 없다
- 제네릭의 서브 타입 허용 않음



제네릭 상속 및 타입 한정

❖ 타입 한정

```
<T extends 특정클래스> 반환타입 메서드이름(…) { … }
<T extends 인터페이스> 반환타입 메서드이름(…) { … }
```

부모가 인터페이스라도 extends를 사용한다.

CupCup.java

```java
package generic_package;

public class CupCup<T extends Beverage> {
    private T beverage;
    public T getBeverage() {
        return beverage;
    }

    public void setBeverage(T beverage) {
        this.beverage = beverage;
    }
}

```

*BoundedTypeDemo.java

```java
package generic_package;

public class BoundedTypeDemo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Cup<String> strCup = new Cup<>();
        //CupCup<String> strCup1 = new CupCup<>();

        Cup<Beer> beer = new Cup<>();
        Cup<Boricha> cup2 = new Cup<>();
        Cup<Beverage> cup3 = new Cup<>();

    }

}

```

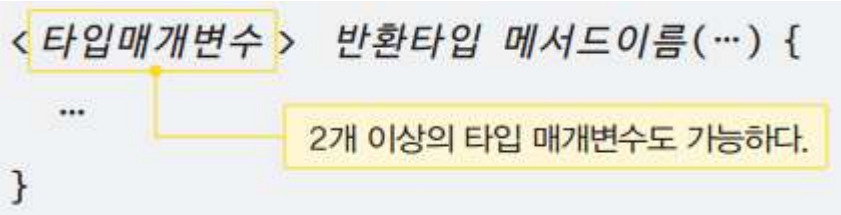
Cup.java

```java
package generic_package;

public class Cup<T> {
    private T beverage;
    public T getBeverage() {
        return beverage;
    }

    public void setBeverage(T beverage) {
        this.beverage = beverage;
    }
}

```

제네릭 메서드

❖ 의미와 선언 방법

- 타입 매개변수를 사용하는 메서드
- 제네릭 클래스뿐만 아니라 일반 클래스의 멤버도 될 수 있음
- 제네릭 메서드를 정의할 때는 타입 매개변수를 반환 타입 앞에 위치



- 제네릭 메서드를 호출할 때는 구체적인 타입 생략 가능

- JDK 7과 JDK 8의 경우 익명 내부 클래스에서는 다이어몬드 연산자 사용 불허
- JDK 9부터는 익명 내부 클래스에서도 다이어몬드 연산자 사용 가능

제네릭 메서드

❖ 예제

- 배열의 타입에 상관없이 모든 원소 출력

```
3   public class GenMethod1Demo {
4       static class Utils {
5           public static <T> void showArray(T[] a) {
6               for (T t : a)
7                   System.out.printf("%s ", t);
8               System.out.println();
9           }
10
11          public static <T> T getLast(T[] a) {
12              return a[a.length - 1];
13          }
14      }
15
16      public static void main(String[] args) {
17          Integer[] ia = { 1, 2, 3, 4, 5 };
18          Character[] ca = { 'H', 'E', 'L', 'L', 'O' };
19
20          Utils.showArray(ia);
21          Utils.<Character>showArray(ca);
22
23          System.out.println(Utils.getLast(ia));
24      }
25  }
```

제네릭 메서드

❖ 제네릭 타입에 대한 범위 제한

- 사용 방법

```
<T extends 특정클래스> 반환타입 메서드이름(…) { … }
<T extends 인터페이스> 반환타입 메서드이름(…) { … }
```

부모가 인터페이스라도 extends를 사용한다.

- 예제

```java
public class GenMethod2Demo {
    static class Utils {
        public static <T extends Number> void showArray(T[] a) {
            for (T t : a)
                System.out.printf("%s ", t);
            System.out.println();
        }
    }

    public static void main(String[] args) {
        Integer[] ia = { 1, 2, 3, 4, 5 };
        Double[] da = { 1.0, 2.0, 3.0, 4.0, 5.0 };
        Character[] ca = { 'H', 'E', 'L', 'L', 'O' };

        Utils.showArray(ia);
        Utils.showArray(da);
        // Utils.<Character>showArray(ca);
    }
}
```

```java
class Ticket implements Comparable {
    int no;

    public Ticket(int no) {
        this.no = no;
    }

    public int compareTo(Object o) {
        Ticket t = (Ticket) o;
        return no < t.no ? -1 : (no > t.no ? 1 : 0);
    }
}

public class GenMethod3Demo {
    public static <T extends Comparable> int countGT(T[] a, T elem) {
        int count = 0;
        for (T e : a)
            if (e.compareTo(elem) > 0)
                ++count;
        return count;
    }

    public static void main(String[] args) {
        Ticket[] a = { new Ticket(5), new Ticket(3), new Ticket(10), new Ticket(7), new Ticket(4) };

        System.out.println(countGT(a, a[4]));
    }
}
```

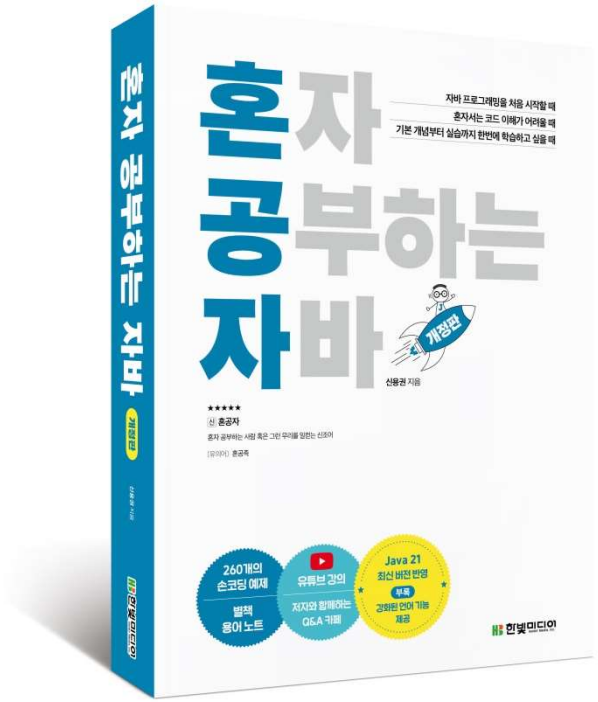
```java
package chap09;

import java.util.NoSuchElementException;
import java.util.StringTokenizer;

public class TokenPrintTest {
    public static void main(String[] args) {
        String s = "of the people, by the people, for the people";

        try {
            showTokens(s, ", ");
        } catch (NoSuchElementException e) {
            System.out.println("끝");
        }
    }
    // while(true) {}을 사용하는 showTokens() 메서드를 추가
    public static void showTokens(String s, String delim) {

    }
}

```

```
of
the
people
by
the
people
for
the
people
```

❖ Max 클래스를 구현해 보자

- 제네릭 클래스로 인수가 숫자라면 큰수를 반환
- 문자라면 긴 문자를 반환하는 클래스를 만들어 봅시다.

```
3   public class MaxTest {
4       public static void main(String[] args) {
5           Max<Number> n = new Max<>();
6           System.out.println(n.max(10.0, 8.0));
7           System.out.println(n.max(5, 8.0));
8
9           Max<String> s = new Max<>();
10          System.out.println(s.max("Hello", "Hi"));
11          System.out.println(s.max("Good", "morning"));
12      }
13  }
14
15  class Max<T> {
16      T max(T t1, T t2) {
17
18      }
19  }
```



Thank You!

한빛미디어 Hanbit Media, Inc.