آموزش جاوا (Java) قسمت ۴۸ : معرفی و آموزش جنریک (Generic) (نسخه چاپی)

با سلام به همه دوستان و همراهان ltpro. آیا تا به حال فکر کرده اید که اگر یک متد مرتب سازی داشتیم که اگر داده int به متد می دادیم همان متد می توانست داده های int را مرتب کند و اگر داده string به آن می دادیم می توانست آنها را مرتب کند و یا هر نوع داده را به آن می دادیم می توانست آن را مرتب کند. دقت کنید که فقط یک متد باشد نه این که با اسـتفاده از overload کردن متدها برای هر نوع داده ای یک متد جدا بنویسـیم. فقط یک متد داشـته باشیم که بتواند با هر نوع داده ای کار کرده و عمل مرتب سازی را بر روی آن انجام دهد. این کار شدنی است. با استفاده از متدها و کلاس های جنریک (generic) در جاوا می توان این کار را انجام داد. با استفاده از امکان جنریک شـما می توانیـد یک متـد معرفی کنیـد و این متد برای چندین نوع داده کار خواهد کرد. یا می توانید یک کلاس جنریک تعریف کنید که داخل کلاس یک مجموعه متدها وجود داشته باشد که با انواع داده های مختلف کار کند. همچنین انواع داده جنریک را می توان در زمان کامپایل بررسی کرد تا اگر نوع داده ای نامعتبر است بتوان آن را کنترل کرد.

متدهای جنریک

شما با اسـتفاده از مفهوم جنریک در جاوا می توانیـد یک متـد جنریک تعریف کنید که آرگومان های آن می تواند از انواع مختلف داده ها باشد. با توجه به نوع داده ای که شـما به عنوان پارامترهای متد ارسال می کنید در هر بار فراخوانی کامپایلر نوع داده ارسالی را تشـخیص می دهد و با توجه به آن عمل می کند. قوانین تعریف متدهای جنریک به شرح زیر است:

- همه ی متدهای جنریک یک نوع داده برای پارامتر دارند که هنگام تعریف به جای این که از انواع داده که قبلا گفتیم اسـتفاده کنند از یک شاخص بین دو کاراکتر <> اسـتفاده می شود و اگر قرار باشد که متدی یک مقدار جنریک برگرداند از همین شاخص به جای انواع داده قبلی اسـتفاده خواهد شد. همچنین برای این که بخواهیم نوع داده پارامتر ها هم جنریک باشد به جای استفاده از انوع داده اولیه از شاخص ها استفاده می کنیم.
- به شاخصـی که به جای نوع داده اصـلی قرار می دهیم پارامتر نوع گفته می شود و اگر قرار باشـد که از دو نوع داده جـدا از هم در برنامه استفاده شود باید از چند پارامتر نوع متفاوت استفاده کرد که آنها را با استفاده از , جدا می کنیم.
- بدنه یک متد جنریک مانند بدنه بقیه متدها می باشد فقط به جای اسـتفاده از نوع داده ای که قبلا از آن استفاده می شد از پارامتر نوع استفاده می کنیم.

مثال زیر مشخص می کند که چگونه می توانیم یک آرایه از داده های مختلف را در خورجی چاپ کنیم.

```
public class GenericMethodTest
{
  // generic method printArray
  public static < E > void printArray( E[] inputArray )
  {
    // Display array elements
      for ( E element : inputArray ){
        System.out.printf( "%s ", element );
      }
      System.out.println();
  }

  public static void main( String args[] )
  {
    // Create arrays of Integer, Double and Character
    Integer[] intArray = { 1, 2, 3, 4, 5 };
```

```
        Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };
        Character[] charArray = { 'H', 'E', 'L', 'L', 'O' };

        System.out.println( "Array integerArray contains:" );
        printArray( intArray  ); // pass an Integer array

        System.out.println( "\nArray doubleArray contains:" );
        printArray( doubleArray ); // pass a Double array

        System.out.println( "\nArray characterArray contains:" );
        printArray( charArray ); // pass a Character array
    }
}
```

خروجی برنامه بالا به شکل زیر خواهد بود:

```
Array integerArray contains:
1 2 3 4 5 6

Array doubleArray contains:
1.1 2.2 3.3 4.4

Array characterArray contains:
H E L L O
```

در مثال بالا حرف E یک پارامتر نوع می باشد که به هنگام استفاده می توان از انواع داده مختلفی به جای این پارامتر نوع استفاده کرد.

محدود کردن پارامترهای نوع

ممکن است اوقاتی پیش بیایـد که بخواهیم پارامترهای نوع را محدود تر کنیم تا در اجرا متد ها و کدهایمان به مشـکل برنخوریم. در این صورت پارامتر نوع باید به گونه ای باشد که در حین فراخوانی اگر نوع داده مورد نظر شـروط مورد نظر را نداشته باشد اجازه اجرا به برنامه ندهـد. برای مثـال ممکن است که یک متـد فقط روی اعـداد کار کنـد. بنابراین نوع داده ای که بایـد به متـد داده شود یا بایـد از کلاس Number باشـد و یا یکی از زیرکلاس های آن باشـد. برای انجام این کار بایـد ابتـدا نام خود پارامتر نوع را بیاوریم و بعد از آن کلمه کلیدی Extends و بعـد از آن نام کلاس پدر را بیاوریم. برای محدود کردن پارامترهای نوع می توان از کلمات کلیدی extends , implements استفاده کرد. برای مثال به کد زیر دقت کنید. این مثال بین ۳ شئ بزرگترین آنها را برمی گرداند.

```
public class MaximumTest
{
  // determines the largest of three Comparable objects
  public static <T extends Comparable<T>> T maximum(T x, T y, T z)
  {
    T max = x; // assume x is initially the largest
```

```
    if ( y.compareTo( max ) > 0 ){
      max = y; // y is the largest so far
    }
    if ( z.compareTo( max ) > 0 ){
      max = z; // z is the largest now
    }
    return max; // returns the largest object
  }
  public static void main( String args[] )
  {
    System.out.printf( "Max of %d, %d and %d is %d\n\n",
              3, 4, 5, maximum( 3, 4, 5 ) );

    System.out.printf( "Maxm of %.1f,%.1f and %.1f is %.1f\n\n",
              6.6, 8.8, 7.7, maximum( 6.6, 8.8, 7.7 ) );

    System.out.printf( "Max of %s, %s and %s is %s\n","pear",
      "apple", "orange", maximum( "pear", "apple", "orange" ) );
  }
}
```

خروجی برنامه بالا به شکل زیر خواهد بود:

```
maximum of 3, 4 and 5 is 5

maximum of 6.6, 8.8 and 7.7 is 8.8

maximum of pear, apple and orange is pear
```

کلاس های جنریک

تعریف یـک کلاس جنریک بسـیار شبیه به تعریف یک کلاس معمولی است. تنها تفاوت آن این است که بعـد از نام کلاس پارامترهای نوع بین <> خواهد آمد. اگر بیش از یک پارامتر نوع داشته باشیم باید آنها را با استفاده کاما (,) جدا کنیم. برای مثال به کد زیر توجه کنید که چگونه یک کلاس جنریک تعریف کرده ایم:

```
public class Box<T> {

  private T t;

  public void add(T t) {
    this.t = t;
  }
```

```
  public T get() {
    return t;
  }

  public static void main(String[] args) {
     Box<Integer> integerBox = new Box<Integer>();
     Box<String> stringBox = new Box<String>();

     integerBox.add(new Integer(10));
     stringBox.add(new String("Hello World"));

     System.out.printf("Integer Value :%d\n\n", integerBox.get());
     System.out.printf("String Value :%s\n", stringBox.get());
  }
}
```

خروجی برنامه بالا به شکل زیر خواهد بود:

```
Integer Value :10
String Value :Hello World
```

در زبان جاوا از مفهوم جنریک بسیار استفاده شده است. برای مثال برای داده ساختار هایی مانند لیست هاو set ها از این مفهوم استفاده شـده است. نکته ای که باید به آن دقت داشـته باشـید این است که برای این که از کلاس یا متد جنریک اسـتفاده کنیم نمی توانیم انواع داده های اصلی مانند int و یا double را به آن بدهیم و باید کلاس wrapper آنها یعنی Integer و Double را به کلاس جنریک اختصاص می دهیم. همانطور که در مثال های بالا دیدیم.**Itpro باشید**

نویسنده: مهدی عادلی فر

منبع: جزیره برنامه نویسی و توسعه نرم افزار وب سایت توسینسو

مطلب اصلی