پارامتر و آرگومان

پارامترها، داده‌های خامی هستند که، متد آنها را پردازش می‌کند و سپس اطلاعاتی را که به دنبال آن هستید، در اختیار شما قرار می‌دهد. فرض کنید پارامترها مانند اطلاعاتی هستند که، شما به یک کارمند می‌دهید که بر طبق آنها کارش را به پایان برساند. یک متد می‌تواند هر تعداد پارامتر داشته باشد. هر پارامتر می‌تواند از انواع مختلف داده باشد. در زیر یک متد با N پارامتر نشان داده شده است :

```
returnType MethodName(datatype param1, datatype param2, ... datatype paramN)
{
   code to execute;
}
```

پارامترها، بعد از نام متد و بین پرانتزها قرار می‌گیرند. بر اساس کاری که متد انجام می‌دهد، می‌توان تعداد پارامترهای زیادی به متد اضافه کرد. بعد از فراخوانی یک متد باید آرگومانهای آن را نیز تأمین کنید. آرگومان‌ها، مقادیری هستند که به پارامترها اختصاص داده می‌شوند. ترتیب ارسال آرگومانها به پارامترها مهم است. عدم رعایت ترتیب در ارسال آرگومانها باعث به وجود آمدن خطای منطقی و خطای زمان اجرا می‌شود. اجازه بدهید که یک مثال بزنیم :

```
1  package myfirstprogram;
2
3  import java.util.Scanner;
4  import java.text.MessageFormat;
5
6  public class MyFirstProgram
7  {
8      static int CalculateSum(int number1, int number2)
9      {
10         return number1 + number2;
11     }
12
13     public static void main(String[] args)
14     {
15         Scanner input = new Scanner(System.in);
16         int num1, num2;
17
18         System.out.print("Enter the first number: ");
19         num1 = input.nextInt();
20         System.out.print("Enter the second number: ");
21         num2 = input.nextInt();
22
23         System.out.println(MessageFormat.format("Sum = {0}", CalculateSum(num1, num2)));
24     }
25 }
```

```
Enter the first number: 10
Enter the second number: 5
Sum = 15
```

در برنامه بالا یک متد به نام CalculateSum() (خطوط 8-11) تعریف شده است که وظیفه آن جمع مقدار دو عدد است. چون این متد مقدار دو عدد صحیح را با هم جمع می‌کند پس نوع برگشتی ما نیز باید int باشد. متد دارای دو پارامتر است که اعداد را به آنها ارسال می‌کنیم. به نوع داده‌ای پارامترها توجه کنید. هر دو پارامتر یعنی number1 و number2 مقادیری از نوع اعداد صحیح (int) دریافت می‌کنند. در بدنه متد دستور return نتیجه جمع دو عدد را بر می‌گرداند. در داخل متد main() برنامه از کاربر دو مقدار را درخواست می‌کند و آنها را داخل متغیرها قرار می‌دهد. حال متد را که آرگومانهای آن را آماده کرده‌ایم فراخوانی می‌کنیم. مقدار num1 به پارامتر اول و مقدار num2 به پارامتر دوم ارسال می‌شود. حال اگر مکان دو مقدار را هنگام ارسال به متد تغییر دهیم (یعنی مقدار num2 به پارامتر اول و مقدار num1 به پارامتر دوم ارسال شود) هیچ تغییری در نتیجه متد ندارد چون جمع خاصیت جابه جایی دارد.

فقط به یاد داشته باشید که باید ترتیب ارسال آرگومانها هنگام فراخوانی متد دقیقاً با ترتیب قرار گیری پارامترها تعریف شده در متد مطابقت داشته باشد. بعد از ارسال مقادیر 10 و 5 به پارامترها، پارامترها آنها را دریافت می‌کنند. به این نکته نیز توجه کنید که نام پارامترها طبق قرارداد به شیوه کوهان شتری یا camelCasing (حرف اول دومین کلمه بزرگ نوشته می‌شود) نوشته می‌شود. در داخل بدنه متد (خط 10) دو مقدار با هم جمع می‌شوند و نتیجه به متد فراخوان (متدی که متد CalculateSum() را فراخوانی می‌کند) ارسال می‌شود.

در درس آینده از یک متغیر برای ذخیره نتیجه محاسبات استفاده می‌کنیم ولی در اینجا مشاهده می‌کنید که می‌توان به سادگی نتیجه جمع را نشان داد (خط 10). در داخل متد main() از ما دو عدد که قرار است با هم جمع شوند درخواست می‌شود.

در خط 23 متد CalculateSum() را فراخوانی می‌کنیم و دو مقدار صحیح به آن ارسال می‌کنیم. دو عدد صحیح در داخل متد با هم جمع شده و نتیجه آنها برگردانده می‌شود. مقدار برگشت داده شده از متد به وسیله متد format() از کلاس MessageFormat نمایش داده می‌شود. (خط 23) در برنامه زیر یک متد تعریف شده است که دارای دو پارامتر از دو نوع داده‌ای مختلف است:

```
1  package myfirstprogram;
2
3  import java.text.MessageFormat;
4
5  public class MyFirstProgram
6  {
7      static void ShowMessageAndNumber(string message, int number)
8      {
9          System.out.println(message);
10         System.out.println(MessageFormat.format("Number = {0}", number));
11     }
12
13     public static void main(String[] args)
14     {
15         ShowMessageAndNumber("Hello World!", 100);
16     }
17 }
```

```
Hello World!
Number = 100
```

در مثال بالا یک متدی تعریف شده است که اولین پارامتر آن مقداری از نوع رشته و دومین پارامتر آن مقداری از نوع int دریافت می‌کند. متد به سادگی دو مقداری که به آن ارسال شده است را نشان می‌دهد. در خط 15 متد را اول با یک رشته و سپس یک عدد خاص فراخوانی می‌کنیم. حال اگر متد به صورت زیر فراخوانی می‌شد :

```
ShowMessageAndNumber(100, "Welcome to Gimme C#!");
```

در برنامه خطا به وجود می‌آمد چون عدد 100 به پارامتری از نوع رشته و رشته !Hello World به پارامتری از نوع اعداد صحیح ارسال می‌شد. این نشان می‌دهد که ترتیب ارسال آرگومانها به پارامترها هنگام فراخوانی متد مهم است.

به مثال 1 توجه کنید. در آن مثال دو عدد از نوع int به پارامترها ارسال کردیم که ترتیب ارسال آنها چون هردو پارامتر از یک نوع بودند مهم نبود. ولی اگر پارامترهای متد دارای اهداف خاصی باشند ترتیب ارسال آرگومانها مهم است.

```
void ShowPersonStats(int age, int height)
{
    System.out.println(MessageFormat.format("Age = {0}", age));
    System.out.println(MessageFormat.format("Height = {0}", height));
}

//Using the proper order of arguments
ShowPersonStats(20, 160);

//Acceptable, but produces odd results
ShowPersonStats(160, 20);
```

در مثال بالا، نشان داده شده است که حتی اگر متد دو آرگومان با یک نوع داده‌ای قبول کند باز هم بهتر است ترتیب بر اساس تعریف پارامترها رعایت شود. به عنوان مثال، در اولین فراخوانی متد بالا اشکالی به چشم نمی‌آید، چون سن شخص 20 و قد او 160 سانتی متر است. اگر آرگومانها را به ترتیب ارسال نکنیم سن شخص 160 و قد او 20 سانتی متر می‌شود که به واقعیت نزدیک نیست. دانستن مبانی مقادیر برگشتی و ارسال آرگومانها باعث می‌شود که شما متدهای کارآمدتری تعریف کنید. تکه کد زیر نشان می‌دهد که، شما حتی می‌توانید مقدار برگشتی از یک متد را به عنوان آرگومان به متد دیگر، ارسال کنید.

```
int MyMethod()
{
   return 5;
}

void AnotherMethod(int number)
{
   System.out.println(number);
}

// Codes skipped for demonstration

AnotherMethod(MyMethod());
```

چون مقدار برگشتی متد MyMethod() عدد 5 است و به عنوان آرگومان به متد AnotherMethod() ارسال می‌شود، خروجی کد بالا هم عدد 5 است.