آموزش اولویت عملگرها در جاوا

البته که شما می توانید با بیشتر از دو عدد در جاوا محاسبه کنید. اما باید مراقب آنچه قرار است محاسبه شود، باشید. مورد زیر را به عنوان مثال در نظر بگیرید:

```
first_number = 100;

second_number = 75;

third_number = 25;

answer = first_number - second_number + third_number;
```

اگر محاسبه را از سمت چپ به راست انجام داده باشید، 75-100 می شود که پاسخ 25 است. سپس عدد سوم را که 25 است اضافه کنید. کجکوع 50 خواهد بود. به هرحال اگر مد نظر شما این نباشد چطور؟ اگر تمایل داشته باشید اعداد دوم و سوم را با هم اضافه کنید و سپس مجموع را از اولین عدد کسر کنید، چطور؟ بنابراین 25+75 است که پاسخ 100 می باشد. سپس آن را از اولین عدد کسر کنید که 100 می باشد. اکنون مجموع 0 خواهد بود.

برای اطمینان از اینکه جاوا کاری را انجام می دهد که شما می خواهید، می توانید از آکولا استفاده کنید. بنابراین اولین محاسبه مانند زیر خواهد بود:

```
answer = (first_number - second_number) + third_number;
```

این پنجره ی برنامه نویسی می باشد، بنابراین می توانید آن را امتحان کنید:

```
public static void main(String[] args) {

    int first_number, second_number, third_number, answer;

    first_number = 100;
    second_number = 75;
    third_number = 25;
    answer = (first_number - second_number) + third_number;

    System.out.println("Total = " + answer );
}
```

محاسبه دوم نیز به این شکل می باشد:

```
answer = first_number - (second_number + third_number);
```

پنجره ی کد آن را نیز در اینجا مشاهده می کنید:

```
public static void main(String[] args) {

    int first_number, second_number, third_number, answer;

    first_number = 100;
    second_number = 75;
    third_number = 25;
    answer = first_number - (second_number + third_number);

    System.out.println("Total = " + answer );
}
```

اکنون اجازه بدهید چند عمل ضرب و جمع را امتحان کنیم.

نمادهای ریاضی خود را به (که اپراتور نامیده می شوند) به جمع و ضرب تبدیل کنید:

```
answer = first_number + second_number * third_number;
```

تمام آکولادها را حذف کرده و سپس برنامه ی خود را اجرا کنید.

بدون آکولا تصور می کنید که Java از چپ به راست محاسبه را انجام می دهد .بنابراین تصور می کنید که عدد اول را به عدد دوم اضافه می کند تا 175 به دست آورد. سپس تصور می کنید که در عدد سوم ضرب می شود که 25 می باشد . بنابراین پاسخ 4375 خواهد بود. سپس برنامه را اجرا کنید. پاسخ حقیقی را که شما به دست می آورید 1975 می باشد. پس جریان چیست؟

دلیل اینکه جاوا پاسخ اشتباه ارائه می دهد Operator Precedence است .جاوا برخی از نمادهای ریاضی را مهم تر از بقیه در نظر می گیرد. این برنامه ضرب را مقدم به جمع می داند، بنابراین عملیات ضرب را قبل از جمع انجام می دهد، سپس جمع را انجام می دهد. بنابراین جاوا در حال انجام عملیات زیر می باشد:

```
answer = first_number + (second_number * third_number);
```

با قرار دادن آکولادها در جای درست مشاهده می کنید که عدد دوم در عدد سوم ضرب شده است. سپس مجموع به اولین عدد اضافه می شود. بنابراین حاصل 75 در 25 عدد 1875 می باشد. عدد 100 را اضافه کنید، که 1975می باشد.

اگر آن را به روش دیگری می خواهید، فراموش نکنید که با استفاده از آکولادها به جاوا اعلام کنید:

```
answer = (first_number + second_number) * third_number;
```

تقسیم مشابه ضرب می باشد: جاوا ابتدا تقسیم را انجام می دهد وسپس جمع و یا تفریق را. خط پاسخ خود را به شکل زیر تغییر دهید:

```
answer = first_number + second_number / third_number;
```

پاسخی که به دست می آورید 103 می باشد. اکنون چند آکولاد اضافه کنید:

```
answer = (first_number + second_number) / third_number;
```

پاسخ این بار 7 خواهد بود. بنابراین بدون آکولادها، جاوا ابتدا تقسیم را انجام می دهد و سپس 100 را به مجموع اضافه می کند – این عملکرد از چپ به راست کار نمی کند.

در اینجا لیستی از Operator Precedence را مشاهده می کنید:

ضرب و تقسیم – به طور مساوی رفتار می شوند، اما نسبت به جمع و تفریق دارای اولویت هستند .جمع و تفریق – به طور مساوی رفتار می شوند اما نسبت به ضرب و تقسیم اولویت پایین تری دارند .بنابراین اگر فکر می کنید که جاوا پاسخ اشتباه به شما می دهد، به یاد داشته باشید که Operator Precedence مهم می باشد و چند آکولاد اضافه می کند .در قسمت بعدی به چگونگی ذخیره ی مقادیر با استفاده از Java نگاهی خواهیم داشت.