Operators

(عملگرها)

ساده ترین دستورات چاپ و نمایش

- مبحث کنترل دسترسی ها مفهومی به نام "**import static**"
را معرفی می کند که به جاوا SE5 اضافه شده و کتابخانه کوچکی
را برای ساده تر نوشتن عبارات نمایشی و چاپ،ایجاد کرده است.

```
import static java.lang.System.*;
```

حال به همه فیلدها و متدهای استاتیک موجود در کلاس سیستم
دسترسی خواهیم داشت.

استفاده از اپراتورهای جاوا

- یک اپراتور یک یا چند آرگومان را می گیرد و یه مقدار جدید تولید می کند. جمع(+) و تفریق (-)یگانی،ضرب(*)،تقسیم(/) و تساوی(=) همه در اکثر زبان های برنامه نویسی به صورت مشابه عمل می کنند.
- تمام اپراتور ها مقداری را از طریق عملوندهای(operand های) خود تولید می کنند.به علاوه بعضی از اپراتورها می توانند مقدار operandای را تغییر دهند.به این حالت اثر جانبی (Side effect) می گویند.
- تقریبا تمام اپراتورها تنها با نوع داده های اولیه کار می کنند.

تقدم

- در مواردی که عبارتی با چندین اپراتور داریم،تقدم عملگرها نحوه محاسبه عبارت را توضیح می دهد.ساده ترین راه حفظ تقدم عملگرها این است که بدانیم ضرب و تقسیم تقدم بیشتری از جمع و تفریق دارند.
-

```
int x = 1, y = 2, z = 3;
int a = x + y - 2/2 + z;  // (1)
int b = x + (y - 2)/(2 + z); // (2)
System.out.println("a = " + a + " b = " + b);
```

- توجه داشته باشید دستور از اپراتور "+" به عنوان " الحاق رشته " و در صورت لزوم "تبدیل رشته" استفاده می کند

تساوی

- تساوی از طریق اپراتور = نمایش داده می شود و به معنی این است که مقدار سمت راست اپراتور (که rvalue نامیده می شود) را بگیر و در قسمت سمت چپ اپراتور (lvalue) کپی کن.

```
a = 4;
```

- به این مفهوم در اصطلاح aliasing (استفاده از نامی مستعار) گفته می شود.

اپراتورهای ریاضی

- اپراتورهای اساسی ریاضی همانند اپراتورهای موجود در سایر زبان های برنامه نویسی عبارتند از:
  - ✓ جمع(+)
  - ✓ تفریق(-)
  - ✓ تقسیم(/)
  - ✓ ضرب(*)
  - ✓ مدول(% باقیمانده را از تقسیم اعداد صحیح برمی گرداند)

اپراتورهای جمع و تفریق یگانی

- اپراتورهای جمع و تفریق یگانی همان اپراتورهای جمع و تفریق باینری هستند.کامپایلر از طریق شیوه ای که دستورات را می نویسید،نحوه استفاده از آن را بدست می آورد.برای مثال عبارت

```
x = -a;
```

دارای یک معنای مشخص است. کامپایلر قادر به محاسبه عبارت زیر می باشد:

```
x = a * -b;
```

اما خواننده ممکن است دچار سردرگمی شود برای همین بهتر است بنویسیم:

```
x = a * (-b);
```

افزایش و کاهش خودکار
Auto increment and decrement

- دو تا از میانبرهای خوب،اپراتورهای افزایش و کاهش هستند. (اغلب با نام اپراتورهای افزایش خودکار (++) و کاهش خودکار (--) معرفی می شوند.)

```
a = a +1;   ->   a ++;
```

- هر اپراتور به دو شیوه نمایش داده می شود:پیشوندی(prefix) و پسوندی(postfix)
  - *Pre- (increment | decrement) (++a) :*

    عملیات انجام می شود و مقدار در a قرار می گیرد
  - *Post - (increment | decrement) (a++) :*

    مقدار در a قرار می گیرد و عملیات انجام می شود.

عملگرهای رابطه ای

- عملگرهای رابطه ای عبارتند از:
  - ✓ کوچکتر (<)
  - ✓ بزرگتر (>)
  - ✓ کوچکتر مساوی(=<)
  - ✓ بزرگتر مساوی(=>)
  - ✓ برابری(==)
  - ✓ نابرابری(=!)
- بررسی برابری objectها عملگرهای رابطه ای == و =! با همه ی objectها کار می کنند اما مفهوم آنها

اغلب برای برنامه نویسان تازه کار جاوا گیج کننده است.
برای محاسبه محتویات واقعی یک object ، باید از متد equals() که برای همه objectها موجود است استفاده کنید.

عملگرهای منطقی

- هر کدام از عملگرهای منطقی (||)OR، (&&)AND و (!)NOT ،براساس رابطه منطقی بین آرگومان هایش یک مقدار true یا false را تولید می کند.
- توجه کنید که مقادیر boolean در مواردی که خروجی string مد نظر است،به صورت خودکار به فرم متنی مناسب تبدیل می شوند.

اپراتور سه گانه if-else

- اپراتور سه گانه که اپراتور شرطی نیز نامیده می شود،گرچه به خاطر داشتن سه عملوند کمی غیرعادی به نظر می رسند اما چون مقداری را تولید می کنند اپراتور هستند.(برخلاف عبارات if-else معمولی که در درس های آینده خواهیم دید).
- فرم نوشتن این عبارات به صورت زیر است:

```
boolean-exp ? value0 : value1 ;
```

اپراتورهای رشته ای + و +=

- یکی از موارد خاص استفاده از اپراتورها در جاوا:
همان گونه که در قبل گفته شد،اپراتورهای + و =+ می توانند برای الحاق رشته ها نیز استفاده شوند.

اپراتور Cast

- اگر یک مقدار صحیح را به یک متغیر اعشاری تخصیص دهید،کامپایلر به صورت خودکار مقدار int را به float تبدیل می کند.
- جاوا اجازه می دهد که هر نوع داده اولیه را به نوع داده اولیه دیگر (به جز نوع boolean) تخصیص دهید.
- کوتاه سازی و گرد کردن

هنگامی که در حال انجام تبدیل های نزدیک کننده (narrowing conversion) هستید به مسئله کوتاه سازی و گرد کردن توجه داشته باشید.