# تعیین کننده سطح دسترسی جاوا یا modifier در جاوا

تعیین کننده سطح دسترسی جاوا یا modifier در جاوا، کلمه های کلیدی ای هستند که شما به تعریف متغیر ها اضافه میکنید تا معنی این تعریف ها را عوض کنید. در جاوا چندین نوع مدیفایر داریم که در این جلسه از دوره آموزشی، میخواهیم آنها را بررسی کنیم. با **برنامه چی** همراه باشید.

**در این جلسه چه چیزهایی یاد میگیرید؟**

انواع Modifier در جاوا

چه سطح هایی در جاوا وجود دارند؟

Access Modifier در جاوا یا تعیین کننده سطح دسترسی جاوا

Non-Access Modifier در جاوا

سوالات خودتان را بپرسید

در جلسه بعد چه چیزهایی یاد میگیریم؟

**دیدن همین مقاله در سایت**

## انواع Modifier در جاوا

زبان برنامه نویسی جاوا دارای انواع مختلفی از مدیفایر ها میباشد که شامل دو مورد زیر میشوند:

- مدیفایر های دسترسی جاوا (Java Access Modifiers)
- مدیفایر های غیر دسترسی (Non Access Modifiers)

برای اینکه از تعیین کننده سطح دسترسی جاوا استفاده کنید، باید کلمه کلیدی آن را در تعریف یک کلاس، متد یا متغیر بیاورید. مدیفایر در زمان تعریف کردن، در اول جمله ظاهر میشود. مانند چیزی که در مثال پایین میبینید.

**مثال**

```
public class className {
  // ...
}
private boolean myFlag;
static final double weeks = 9.5;
protected static final int BOXWIDTH = 42;

public static void main(String[] arguments) {
  // body of method
}
```

## چه سطح هایی در جاوا وجود دارند؟

در این درس میخواهیم درباره این بحث کنیم که modifier ها در جاوا میتوانند سطوح دسترسی را برای المان هایی که تعریف میکنیم مشخص کنند. اما این تعیین کننده سطح دسترسی جاوا چه چیزی هستند؟

بعضی چیز ها در برنامه نویسی وجود دارند که تا قبل از اینکه در عمل از آنها استفاده نکنید، معنی و مفهوم و کاربرد درست آنها را یاد نمیگیرید. این بحث هم از همین موارد محسوب میشود. پس به شما پیشنهاد میکنم تعریف های این بخش را به خوبی یاد بگیرید تا بعدا در درست کردن برنامه ها، انها را فراموش نکرده باشید. (وقتی هم که وارد برنامه نویسی بشید بعد از دو سه بار استفاده از modifier ها در جاوا، کلا یاد میگیرید که چطوری باهاشون کار بکنید، پس اصلا نگران نباشید).

**پکیج (Package):** هنگامی در حال درست کردن یک برنامه به زبان جاوا هستید، یعنی همان برنامه های اندروید، پروژه شما به چندین قسمت تقسیم میشود. بزرگترین تکه اپلیکیشن شما پکیج نام دارد. پکیج تعریف خاص و پیچیده ای ندارد و در حقیقت یک فولدر است که کلاس های خاصی از برنامه درون آن قرار میگیرند.

معمولا برنامه ها را بر اساس ویژگی هایی که دارند به چندین پکیج تقسیم میکنیم. برنامه های ماژولار به این شکل تقسیم بندی میشوند. یعنی مثلا اگر اپلیکیشن شما دارای قابلیتی به نام سبد خرید است، کلاس های مربوط به این قابلیت (Feature) را درون یک پکیج جداگانه قرار میدهیم. این روش مزایای خاص خودش را دارد که برای آشنای بیشتر با آن میتوانید مقاله زیر را مطالعه کنید.

**برنامه نویسی ماژولار چیست؟**

**کلاس (Class):** درون یک پکیج کلاس های مختلفی وجود دارند که بسته به حجم اپلیکیشن شما و نیازی که به کار های مختلف دارید، تعداد این کلاس ها هم میتواند کم و زیاد بشود. در اصل هر کلاس میتواند یک عملکرد خاص را برای شما انجام بدهد.

مثلا فرض کنید که در قسمتی از نرم افزار نیاز دارید تا اطلاعاتی را از سرور دریافت کنید و درون پایگاه داده ذخیره کنید. معمولا برای اینکار یک کلاس جداگانه میسازند که این عمل را انجام میدهد. بعد هرجایی که نیاز بود، این کلاس را صدا میزنیم تا اطلاعاتی که میخواهیم را برای ما دانلود کند.

**متد (Method):** درباره متد ها در درس های قبلی و کلا مقاله های سایت زیاد بحث کرده ایم. اگر هیچ آشنایی با متد ها ندارید پیشنهاد میکنم دروس قبلی دوره آموزش جاوا را حتما مطالعه کنید. اما بصورت خلاصه باید بگویم یک کلاس از چندین متد تشکیل شده است که این متد ها میتوانند یکدیگر را صدا بزنند و با هم ارتباط داشته باشند.

حتی بعضی از متد های کلاس های دیگر را هم میتوانید از درون کلاسی که داخل آن قرار دارید صدا بزنید و از آن استفاده کنید. (یعنی مثلا اگر الان داخل کلاس A باشید و به یک متد درون کلاس B نیاز پیدا کنید، میتوانید بعد از نام کلاس B، نام متدی که میخواهید را هم صدا بزنید).

## Access Modifier در جاوا یا تعیین کننده سطح دسترسی جاوا

یک سری modifier در جاوا در اختیار شما قرار دارند که با استفاده از آنها میتوانید دسترسی را برای کلاس، متغیر، متد و constructor ها کنترل کنید. کنترل کردن تعیین کننده سطح دسترسی جاوا، به این معنی است که گاهی شما نیاز دارید که بعضی از متدها و متغیر ها فقط درون کلاس کار خودشان را انجام بدهند (یا درون پکیج یا شاید در کل برنامه).

**بعضی چیز ها در برنامه نویسی وجود دارند که تا قبل از اینکه در عمل از آنها استفاده نکنید، معنی و مفهوم و کاربرد درست آنها را یاد نمیگیرید. این بحث هم از همین موارد محسوب میشود. پس به شما پیشنهاد میکنم تعریف های این بخش را به خوبی یاد بگیرید تا بعدا در درست کردن برنامه ها، انها را فراموش نکرده باشید.**

با استفاده از modifier در جاوا میتوانید مشخص کنید که کدام قسمت های برنامه بتوانند متغیر، کلاس یا متدی که تعریف میکنید را ببینند و به آن دسترسی داشته باشند. چهار تعیین کننده سطح دسترسی جاوا وجود دارد که عبارتند از:

**پیشفرض (بدون modifier).**

قابل رویت و دسترسی برای همه اعضای پکیج، که پیشفرض هم همین است. هیچ مدیفایری نیاز ندارد.

**Private: قابل رویت فقط برای کلاسی که داخل آن هستید.**

وقتی که از مدیفایر private برای تعریف کردن یک متغیر، متد یا کلاس استفاده میکنید، در حقیقت فقط به خود کلاسی که درون آن قرار دارید اجاز میدهید که آن متغیر را ببیند. یعنی مثلا اگر در کلاس دیگری از برنامه خودتان باشید و بخواهید یک متد private که در یک کلاس دیگر قرار دارد را صدا بزنید، اجازه این کار به شما داده نخواهد شد.

**Public: قابل رویت برای همه دنیا.**

وقتی از این modifier در جاوا استفاده کنید، آن المانی که تعریف میکنید از همه جای برنامه قابل رویت و دسترسی میباشد. حتی برای پکیج های دیگر و کلاس هایی که درون آنها قرار دارند. شاید فکر کنید که بهتر باشد همیشه از این تعیین کننده سطح دسترسی جاوا استفاده کنید. اما وقتی برنامه شما کمی بزرگ بشود، برای جلوگیری از اختلال در اجرای برنامه مجبور میشوید از مدیفایر ها در جای درست خود استفاده کنید. درباره استفاده از modifier در جاوا در درس های بعدی مفصل توضیح خواهیم داد.

**Protected: قابل رویت برای پکیج و همه کلاس های زیرین (Sub-Class).**

اگر از مدیفایر protected استفاده کنید، المانی که ساخته میشود درون کلاس ها و ساب کلاس های پکیج خودش، و همچنین درون ساب کلاس های پکیج های دیگر (نه کلاس ها) قابل رویت میباشد.

| حوزه دید (SCOPE) | DEFAULT | PRIVATE | PROTECTED | PUBLIC |
|---|---|---|---|---|
| همان کلاس | YES | YES | YES | YES |
| ساب کلاس های همان پکیج | YES | NO | YES | YES |
| کلاس های معمولی همان پکیج | YES | NO | YES | YES |
| ساب کلاس های پکیج دیگر | NO | NO | YES | YES |
| کلاس های معمولی پکیج دیگر | NO | NO | NO | YES |

## Non-Access Modifier در جاوا

یک سری modifier در جاوا در اختیار شما قرار دارد که با استفاده از آنها میتوانید به کاربرد های بیشتری دسترسی داشته باشید. یعنی این نوع تعیین کننده سطح دسترسی جاوا برای کنترل دسترسی متد، کلاس یا متغیر به کار نمیروند. بلکه با استفاده از آنها میتوانید انواع خاصی از این موارد را تولید کنید. یعنی این مدیفایر ها در کنار Access Modifier ها (که بالاتر درباره آنها بحث کردیم)، به کار میروند و به قابل رویت بودن کلاس، متد یا متغیر کاری ندارند.

**Static: برای ساختن متد های کلاس و متغیر های آن.**

اگر یک متد یا یک متغیر را بصورت استاتیک تعریف کنید، آنها فقط درون کلاس قابل دسترسی هستند و اگر از روی آن کلاس یک نمونه (Object) بسازید، دیگر متغیر های static را نمیتوانید درون آنها داشته باشید.

**Final: برای نهایی کردن implement های کلاس ها، متغیر ها و متد ها.**

وقتی یک متغیر final باشد دیگر در کل برنامه مقدار آن عوض نمی شود. یعنی هنگام تعریف مقدار آن را مشخص میکنید و دیگر نمیتوانید مقدار آن را عوض کنید. اگر یک کلاس final تعریف کنید دیگر نمیتوانید از آنها ارث بری داشته باشید (یعنی کلاس عقیم میشود). اگر هم یک متد final باشد دیگر نمیتوانید آن را Override بکنید.

**Abstract: برای درست کردن کلاس ها و متدهای abstract.**

از کلاس های abstract نمیتوانیم مستقیما نمونه بگیریم. Abstract به معنی انتزاعی میباشد و اگر کلاس را Abstract تعریف کنیم، باید کلاس های داخلی برای آن تعریف کنیم و سپس میتوانیم از این کلاس های داخلی نمونه بگیریم.

**synchronized و volatile: برای thread ها به کار میروند.**

## سوالات خودتان را بپرسید

اگر درمورد دوره و تعیین کننده سطح دسترسی در جاوا هر سوالی داشتید در قسمت نظرات بپرسید تا به سرعت به آنها پاسخ بدهیم. امیدواریم این جلسه در مورد modifier در جاوا توضیحات کافی و ساده را ارائه داده باشیم.

## در جلسه بعد چه چیزهایی یاد میگیریم؟

در جلسه بعدی درباره عملگر های پایه ای در زبان جاوا صحبت خواهیم کرد. درس بعدی به شما یک دید کلی میدهد که چگونه این عملگر ها میتوانند در برنامه های جاوا استفاده بشوند.

جلسه قبل: آموزش انواع متغیرهای جاوا

جلسه بعد: عملگرهای جاوا