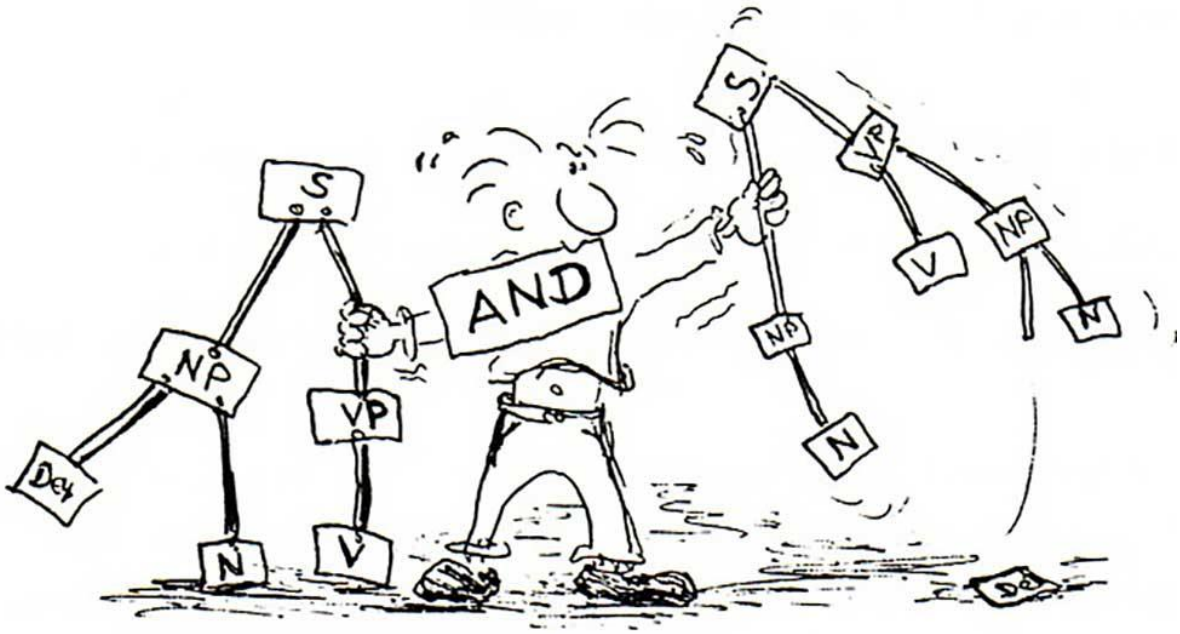
# سینتکس جاوا پایه ای و به زبان ساده


وقتی درباره زبان جاوا صحبت میکنیم، باید بدانیم که این زبان از یک سری اشیاء (Object) درست شده ا ست که میتوانند با صدا زدن متدهای شئ های دیگر، با یکدیگر ارتباط برقرار میکنند. در ابتدای این مقاله نگاه مختصری به معانی کلاس، شئ، متد و متغیر های نمونه (Instance Variable) می اندازیم. برای آشنایی با سینتکس جاوا، با **برنامه چی** همراه باشید.

**دیدن همین مقاله روی سایت**

## سینتکس جاوا چیست؟

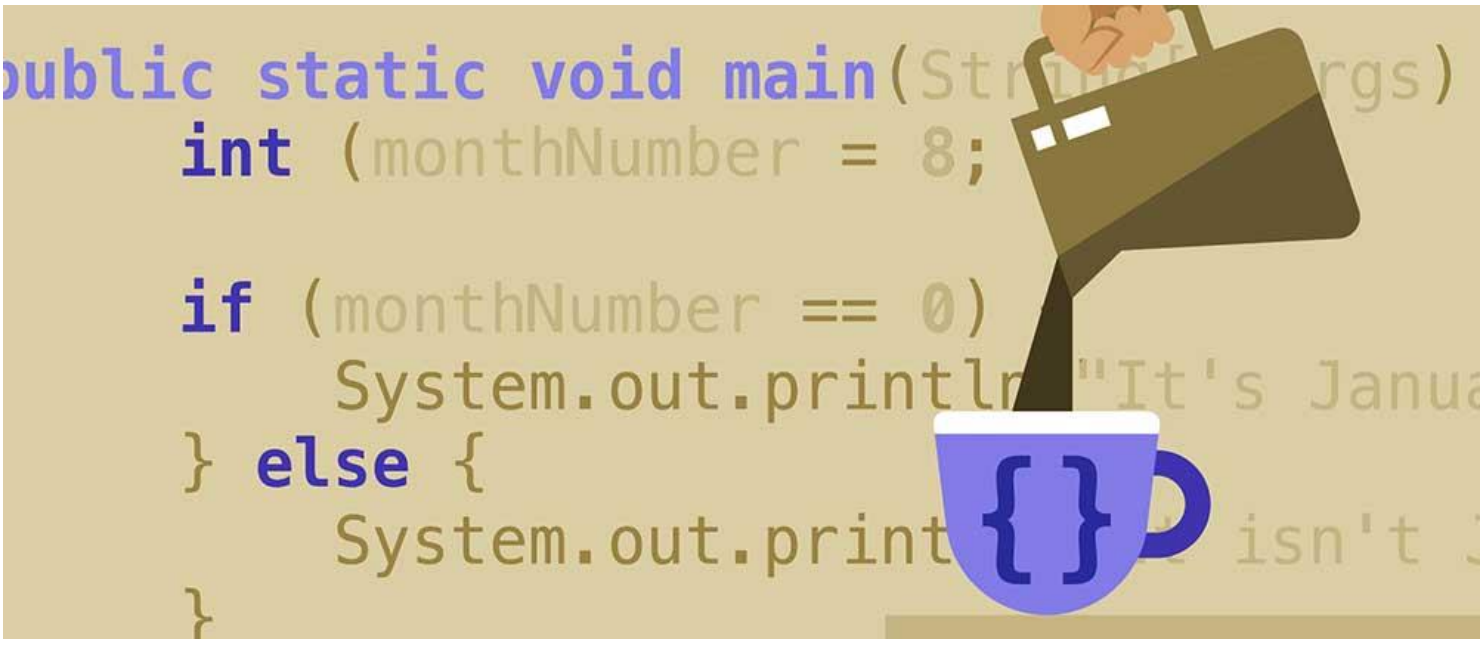


Syntax در دنیای برنامه نویسی، دقیقا مانند املای کلمات در دستور زبان است. زبانی که به آن صحبت میکنیم، دارای قواعدی برای نوشتن است. مثلا در زبان فارسی، کلمه ی خانواده را نمیتوانیم بصورت خوانواده بنویسیم. این یک اشتباه املایی در زبان است. سینتکس هم دقیقا همین است. سینتکس برای این به وجود آمده که از نوشتن کدهای اشتباه در برنامه جلوگیری کند. نباید املای کدهایی که مینویسیم اشتباه باشد. در اینصورت برنامه به درستی اجرا نخواهد شد. علاوه بر غلط های املایی، قواعد نگارشی نیز توسط سینتکس آزموده میشوند.

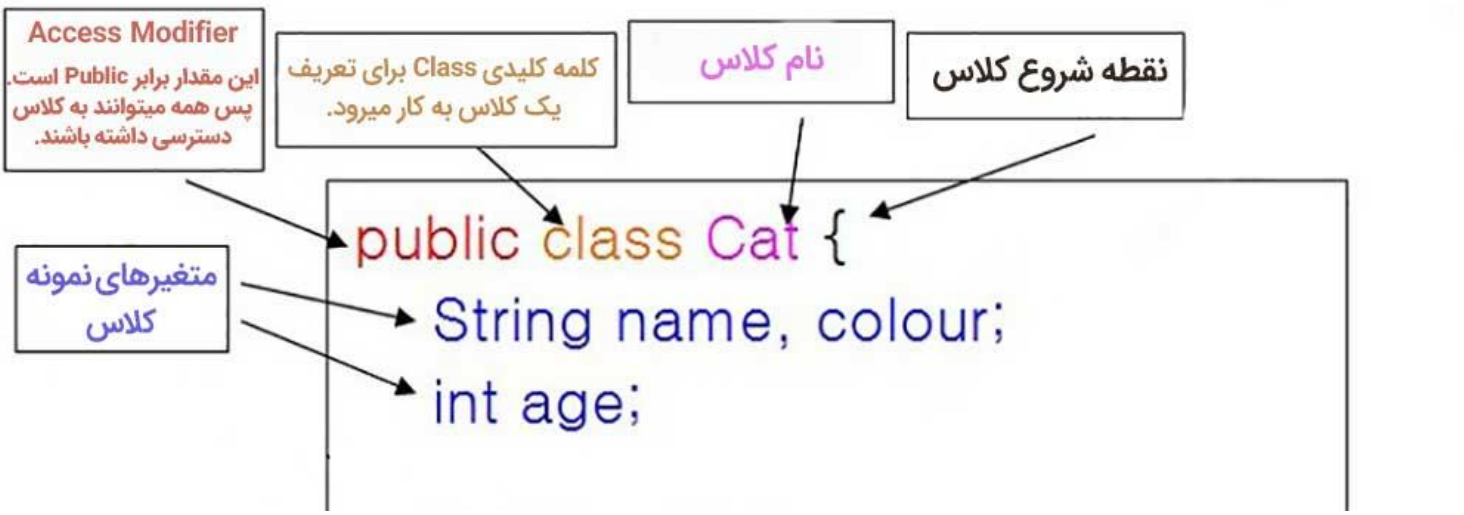
سینتکس در اصل به معنی دستور زبان است. به عنوان مثال در برنامه نویسی به زبان جاوا، باید آخر هر دستور، نقطه ویرگول (;) قرار بدهیم. اگر اینکار را نکنیم، مشکل سینتکس به وجود می آید. همانند دستور زبان که مشخص میکند جای فعل باید در آخر جمله باشد، سینتکس هم ساختار نوشتاری کد ها را مشخص میکند. تفاوت زبان های برنامه نویسی با یکدیگر، در تفاوت سینتکس های آنهاست. قواعد نوشتاری آنها با یکدیگر فرق میکند.

همچنین وقتی شما به یادگیری زبان تازه روی می آورید، در اصل در حال یادگیری سینتکس آن زبان هستید. برای درک بهتر قواعد نوشتاری یک مثال از زبان فارسی میزنم. جمله ی "علی در خانه است"، یک جمله ی درست است. زیرا اجزای جمله در جای مناسب خود قرار دارند و همچنین غلط املایی در آن وجود ندارد. اما جمله "خانه علی در است"، کاملا اشتباه میباشد. زیرا اجزای جمله در جای نادرستی قرار گرفته اند. (دقت کنید در سینتکس، با معنی جمله کاری نداریم. فقط قواعد نوشتار درست در آن بررسی میشوند).

## آشنایی با بعضی از اجزای سینتکس جاوا



- **شئ (Object):** شی ها دارای دو خصوصیت به نام "صفت" و "رفتار" هستند. با یک مثال این دو خ صو صیت را شرح میدهم: یک حیوان مانند گربه را در نظر بگیرید. این گربه دارای یک سری خ صو صیات مانند رنگ مو، رنگ چ شم، اندازه، وزن، قد و غیره میبا شد. این خ صو صیات با نام صـفت شـناخته میشـوند. این گربه همچنین یک سـری کارها را میتواند انجام بدهد، مانند بالا رفتن از درخت، غذا خوردن، شـکار کردن، صـدا در آوردن و غیره. به این کار ها هم در اصـطلاح رفتار گفته میشـود. یک شـی نمونه ای اسـت که از روی کلاس سـاخته میشـود. اشـیاء هم در سینتکس جاوا و هم در بقیه قسمت های برنامه نویسی بسیار مهم هستند. در درس های بعدی بصورت مفصل درباره آنها صحبت خواهیم کرد.
- **کلاس (class):** کلاس را میتوان به عنوان یک شـابلون یا الگو در نظر گرفت که رفتار ها و ویژگی اشیائی که قرار است از روی آن ساخته شوند را مشخص میکند.
- **متد (Method):** متد به زبان ساده، یک رفتار ا ست. یک کلاس میتواند شامل تعداد زیادی متد باشد. منطق نرم افزار و کارهایی که میتواند انجام بدهد، در متدها نوشته میشوند. همچنین همه داده ها و اتفاقاتی که قرار است رخ بدهد در متد ها وجود دارند.
- **متغیر نمونه (Instance Variable):** هر شـی، یک مجموعه منحصـر به فرد از متغیر های نمونه برای خودش دارد. صفت های یک شی با ا ستفاده از مقادیری که در متغیر های نمونه آن ریخته میشود، ساخته میشوند.



# اولین برنامه جاوا

به عنوان اولین پروژه در زبان جاوا میخواهیم یک کد ساده از سینتکس جاوا را بررسی کنیم که عبارت Hello World را نشان میدهد.

**نمونه:**

```
public class MyFirstJavaProgram {

   /* This is my first java program.

    * This will print 'Hello World' as the output

    */

   public static void main(String []args) {

      System.out.println("Hello World"); // prints Hello World

   }

}
```

اکنون میخواهیم مرحله به مرحله از نو شتن تا اجرای این کد را ت شریح کنیم تا شما بتوانید اولین پروژه جاوا خود را ساخته باشید. مراحل زیر را دنبال کنید:

1. نرم افزار Eclipse را باز کنید و کدهایی که در بالا نوشته شد را در آن کپی کنید. (اگر اولین بار ا ست که با زبان جاوا کار میکنید، نگران چیزهایی که در کد نو شته شده ا ست نبا شید. در ادامه دوره با همه آنها آشنا خواهید شد).
2. نام فایل را MyFirstJavaProgram (یا هرچیزی که خودتان دوست دارید) قرار بدهید.
3. حال به محلی که کد خود را در آنجا ذخیره کرده اید بروید. کلید Shift را روی کیبورد نگه دارید و راست کلیک کنید. سپس گزینه Open Command Prompt Here را انتخاب نمایید.
4. در پنجره ای که باز میشود، عبارت javac MyFirstJavaProgram را تایپ کنید. (دقت کنید که حرف اول javac با حروف کوچک نو شته شده با شد. همچنین اگر نام فایل را به دلخواه خودتان انتخاب کرده اید، باید بعد از عبارت javac، دقیقا عین همان نام را با رعایت بزرگی و کوچکی حروف تایپ نمایید).

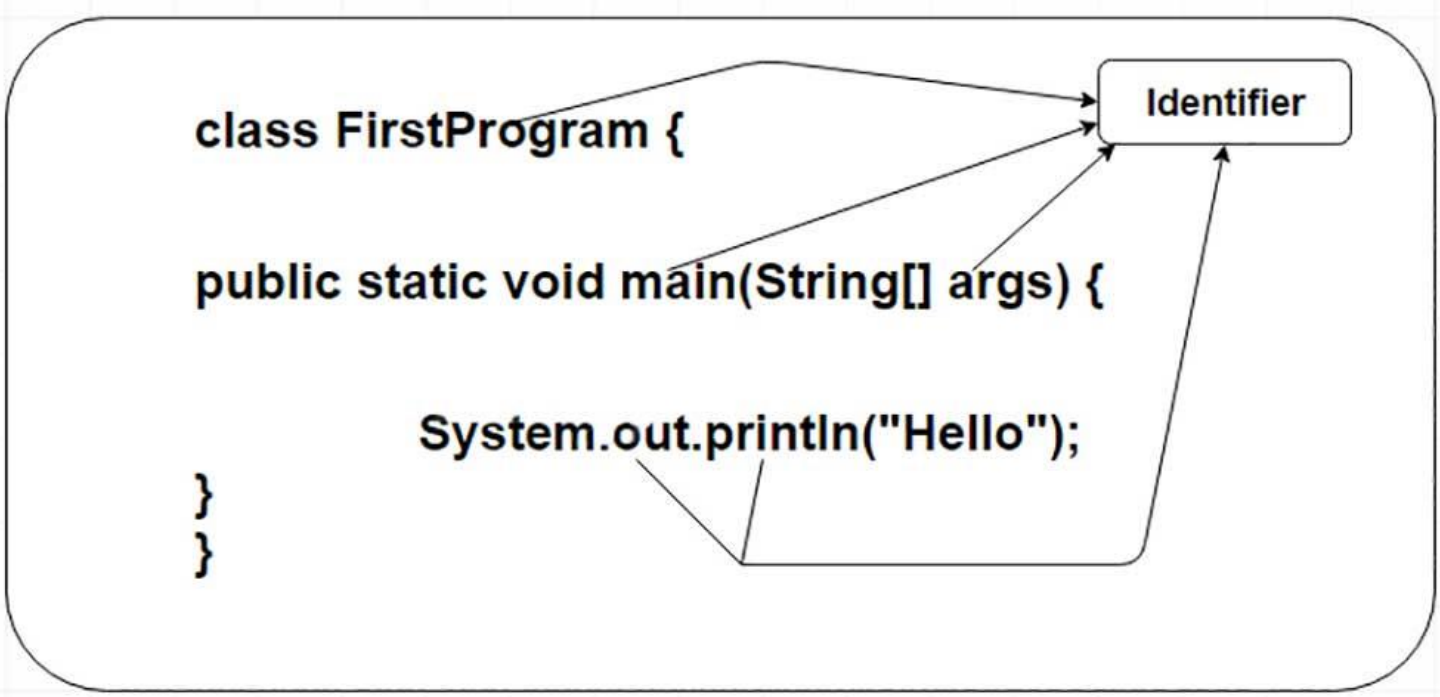
5. در این مرحله کد هایی که شما نو شته اید کامپایل (ترجمه) می شوند. البته به شرط اینکه در کد های شما (یعنی همان سینتکی جاوا)، ا شکالی وجود ندا شته با شد. بعد از کامپایل شدن، در پنجره ی Command Prompt (همان پنجره سیاهی که در آن د ستور کامپایل شدن را دادید)، باید عبارت java MyFirstJavaProgram را تایپ کنید.
6. کلمه Hello World در این صفحه برای شما نشان داده خواهد شد.

## سینتکس جاوا برای افراد مبتدی

برای برنامه نویسی به زبان جاوا، باید چند نکته پایه ای که در زیر برای شما مینویسم را همیشه به خاطر داشته باشید.

- **حساس بودن به حروف بزرگ و کوچک:** زبان جاوا به حروف بزرگ و کوچک حساس است. یعنی مثلا کلمات Hello و hello در سینتکس زبان جاوا با یکدیگر تفاوت دارند.
- **نام کلاس ها:** نام همه کلاس ها در سینتکس زبان جاوا باید با حروف بزرگ شروع بشوند. اگر نام کلاس از چندین کلمه تشکیل شده باشد، حرف ابتدایی هر کلمه باید با حروف بزرگ نوشته بشود. در نام گذاری ها شما مجاز به استفاده از فاصله خالی (Space) نمیباشید. مثال: class MyFirstJavaProject
- **نام متد ها:** نام همه متد ها باید با حروف کوچک شروع شود. اگر نام متد از چندین کلمه پشت سرهم تشکیل شده باشد، حرف ابتدایی کلمه های بعدی (بجز کلمه اول) باید با حروف بزرگ نو شته بشوند. در ا صطلاح به این روش تایپ کردن، Camel Case گفته میشود. مثال: public void myMethodName()
- **نام فایل های برنامه:** نام فایل های برنامه باید دقیقا با کلاسی که در آن فایل قرار دارد یکسان باشد. در هنگام ذخیره کردن فایل، باید با استفاده از نام کلاس آنرا ذخیره کنید. (یادتان باشد که جاوا به حروف بزرگ و کوچک حساس ا ست). پ سوند این فایل های هم java. میبا شد. اگر نام کلاس و فایل شما یکسان نباشد، برنامه شما کامپایل نخواهد شد.
- **مثال:** اگر نام کلاس شما MyFirstJavaProgram باشد، نام فایلی که این کلاس در آن وجود دارد هم باید MyFirstJavaProgram.java باشد.
- **متد Main:** نقطه شروع همه برنامه های جاوا یک متد به نام Main میباشد. این متد در همه برنامه های جاوا، فقط یکبار وجود دارد. یعنی حتی اگر برنامه ما فقط یک متد داشته باشد، آن متد Main است.

# شناسه ها (Identifier) های زبان جاوا




- تمامی کامپوننت ها (اجزا) جاوا به نام نیاز دارند. نام هایی که برای متدها، متغیرها و کلاس ها استفاده میشود را Identifier یا شناسه می نامند.
- برای  شنا سه ها در زبان جاوا نکاتی وجود دارد که باید آنها را به خاطر دا شته با شید. این موارد عبارتند از:
- همه  شنا سه های باید با حروف (از A تا Z یا a تا z)، علامت دلار ($) و یا یک خط فا صله بزرگ (_) شروع بشوند.
- بعد از اولین کاراکتر، میتوانید هر ترکیبی از کاراکتر ها را که دوست داشتی استفاده کنید.
- کلمه های کلیدی که در زبان جاوا استفاده میشوند (و بعدا با آنها آشنا خواهید شد)، نمیتوانند به عنوان شناسه استفاده بشوند.
- **یک نکته مهم این است که شناسه ها به حروف بزرگ و کوچک حساس هستند.**
- نمونه ای از نام های مجاز برای شناسه ها: age, @salary, _value, ___1_value
- نمونه هایی از نام های غیرمجاز برای شناسه ها: 123abc, -salary

## Modifier ها در زبان جاوا

| سطوح دسترسی | | | | |
|---|---|---|---|---|
| Modifier | کلاس | پکیج | کلاس داخلی | بقیه دنیا |
| public | Y | Y | Y | Y |
| protected | Y | Y | Y | N |
| no modifier | Y | Y | N | N |
| private | Y | N | N | N |

Modifier در فارسی به نام تغییر دهنده شناخته میشود. مانند همه زبان های برنامه نویسی دیگر، میتوانید د ستر سی به متد، کلاس و موارد دیگر را در زبان جاوا تغییر بدهید. دو گروه از Modifier ها را داریم:

- **Access Modifier (اصلاح کننده های دسترسی):** default, public, protected, private
- **Non-access Modifier (اصلاح کننده های بدون دسترسی):** final, abstract, strictfp

در مورد این Modifier ها در جلسه های آینده جزئیات بیشتری را آموزش خواهیم داد.

## متغیر های جاوا

متغیر های جاوا به دسته های زیر تقسیم میشوند:

1. **متغیر های محلی**
2. **متغیر های کلاس (متغیر های استاتیک)**
3. **متغیر های نمونه (متغیرهای غیر استاتیک یا Instance Variables)**

## آرایه های جاوا

آرایه ها در ا صل ا شیاء (Object) هایی ه ستند که میتواند تعدادی متغیر از یک جنس را درون خودش نگه دارد. اما به هر حال، آرایه خودش یک شی میباشد که در حافظه Heap قرار دارد. در فصل های بعدی تعریف کردن، ساختن و مقدار دهی این آرایه ها را یاد خواهید گرفت.

# Enum ها در جاوا

Enum ها در ورژن 5 جاوا معرفی شـــدند و متغیر ها را محدود میکنند تا فقط مقادیر از پیش تعیین شـده ای را داشـته باشـد. به مقادیری که در این لیسـت قرار دارند، اصـطلاحا Enum گفته میشـود. با استفاده از Enum ها میتوانید تعداد باگ ها را در کدهای خودتان کاهش بدهید.

به عنوان مثال اگر یک برنامه برای فروش آبمیوه دا شته با شیم، میتوانید  سایز لیوان ها را به  سه مقدار کوچک، متوسط و بزرگ محدود کنید. با این کار میتوانید مطمئن با شید که هیچکس نمیتواند لیوانی با سایز های دیگر را سفارش بدهد.

## مثال

```
class FreshJuice {
   enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
   FreshJuiceSize size;
}

public class FreshJuiceTest {

   public static void main(String args[]) {
      FreshJuice juice = new FreshJuice();
      juice.size = FreshJuice.FreshJuiceSize.MEDIUM ;
      System.out.println("Size: " + juice.size);
   }
}
```

مثال بالا نتیجه زیر را تولید خواهد کرد:

## خروجی

```
Size: MEDIUM
```

**نکته: Enum ها میتوانند داخل یا بیرون از کلاس تعریف بشوند. متدها، متغیرها و Constructor ها میتوانند داخل Enum ها تعریف شوند.**

## کلمه های کلیدی (Keyword) های جاوا

لیست زیر کلمه های کلیدی که از قبل در جاوا رزرو شده اند را نشان میدهد. این کلمه ها نمیتوانند به عنوان متغیر، ثابت یا هیچ شناسه ای به کار برده شوند.

| | | | |
|---|---|---|---|
| abstract | assert | boolean | break |
| byte | case | catch | char |
| class | const | continue | default |
| do | double | else | enum |
| extends | final | finally | float |
| for | goto | if | implements |
| import | instanceof | int | interface |
| long | native | new | package |
| private | protected | public | return |
| short | static | strictfp | super |
| switch | synchronized | this | throw |
| throws | transient | try | void |
| volatile | while | | |

# توضیحات (Comment) ها در جاوا

جاوا از توضیحات تک خطی یا چند خطی، مانند C پشتیبانی میکند. همه کاراکتر هایی که در توضیحات یا Comment قرار میگیرند، توسط کامپایلر زبان جاوا نادیده گرفته میشوند.

### مثال

```
public class MyFirstJavaProgram {

   /* This is my first java program.
    * This will print 'Hello World' as the output
    * This is an example of multi-line comments.
    */

   public static void main(String []args) {
      // This is an example of single line comment
      /* This is also an example of single line comment. */
      System.out.println("Hello World");
   }
}
```

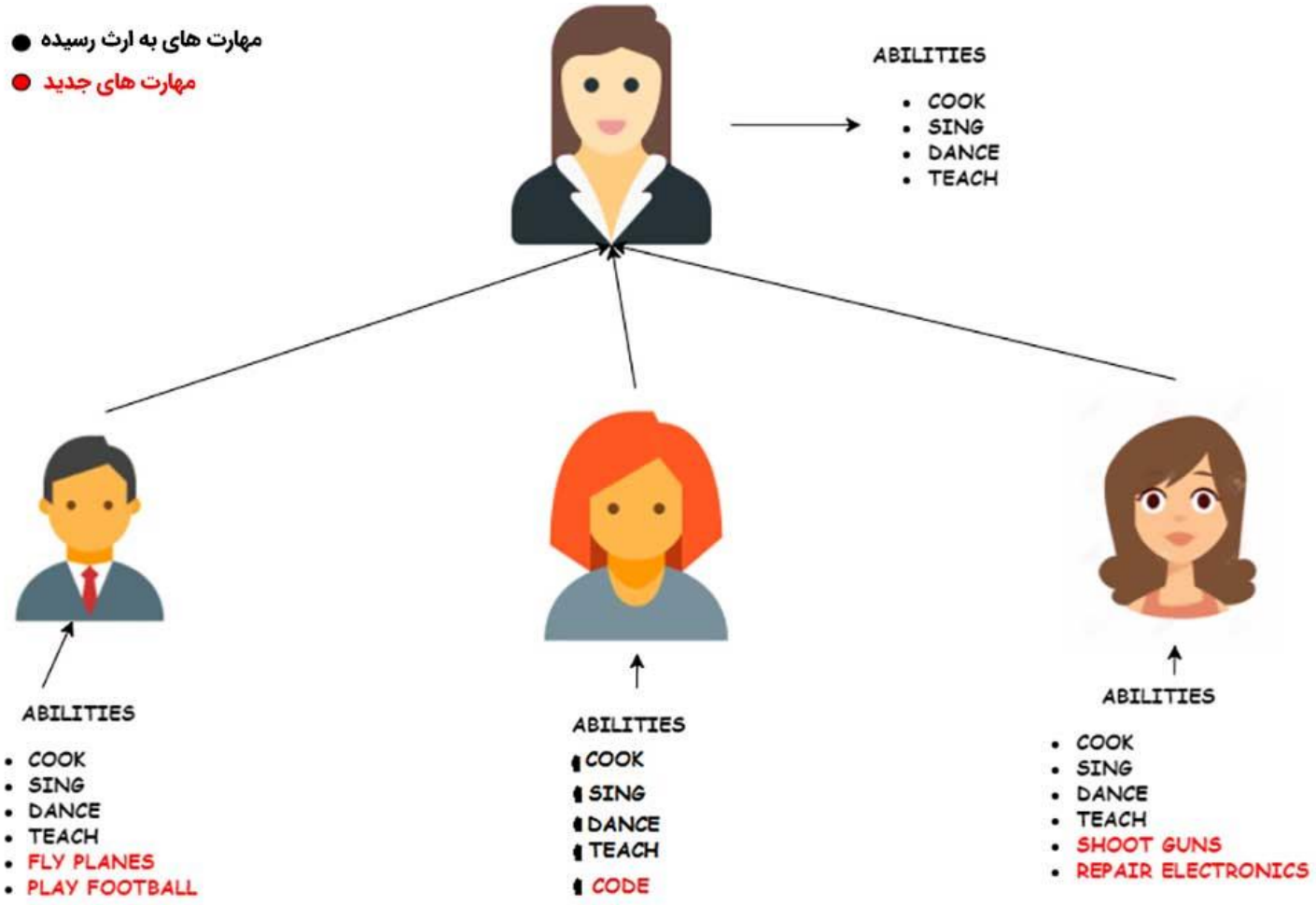
### خروجی

```
Hello World
```

# استفاده از خط های خالی

اگر در میان کدهای شـما خط هایی وجود دارد که کاملا خالی وجود داشـته باشـند، (مثلا Space درون آنها باشد)، توسط زبان جاوا نادیده گرفته میشوند.

# ارث بری




در زبان جاوا کلاس ها میتوانند از کلاس های دیگر سرچشمه بگیرند. به زبان ساده تر فرض کنید که میخواهید یک کلاس بنویسید که مقداری از کدهایی که به آنها نیاز دارید، قبلا در یک کلاس دیگر نوشته شده است. شما میتوانید به جای اینکه دوباره همان کد ها را بنویسید، از کدهای موجود در همان کلاسی که قبلا وجود داشت استفاده کنید. این مفهوم ارث بری به زبان ساده است.

این مفهوم که در زبان انگلیسی به آن inheritance گفته میشود، باعث میشود که نیاز نداشته باشید متد ها و کدهایی که قبلا در کلاس های دیگری موجود هستند را در کلاس جدیدتان دوباره نویسی کنید. در این مدل، کلاسی که از قبل وجود داشته و از کدهای درون آن میخواهیم استفاده کنیم را SuperClass و کلاس جدید که قرار است از کدهای قدیمی استفاده کند، SubClass می نامند.

## اینترفیس ها (Interface)

در زبان جاوا اینترفیس ها را میتوان به عنوان قراردادی در میان اشیاء دید که نحوه ارتباط آنها با یکدیگر را مشخص میکند. اینترفیس ها در رث بری نقش حیاتی را بازی میکنند.

در واقع اینترفیس یک قرارداد ا ست که متد ها و کلاس هایی که باید ا ستفاده کنید را م شخص میکند. اما تصمیم گیری درباره اینکه از کدام متدهای اینترفیس استفاده شود، بر عهده SubClass است. یعنی همان کلاسی که نوشته ایم و از اینترفیس در آن استفاده میکنیم.

اگر چیز زیادی از ارث بری و مفهوم اینترفیس ها متوجه نشـدید اصـلا نگران نباشـید. در درس های آینده به صورت مفصل درباره آن توضیح خواهیم داد.

## در جلسه بعدی چه چیزی یاد میگیریم؟

جلسه بعدی بصورت تخصصی تر درباره اشیاء و کلاس ها در زبان جاوا صحبت میکنیم. در پایان جلسه بعدی شما کاملا با مفاهیم شئ و کلاس ها در جاوا آشنا خواهید شد.

جلسه قبل: آموزش نصب اندروید استودیو و JDK

جلسه بعد: کلاس جاوا و شی