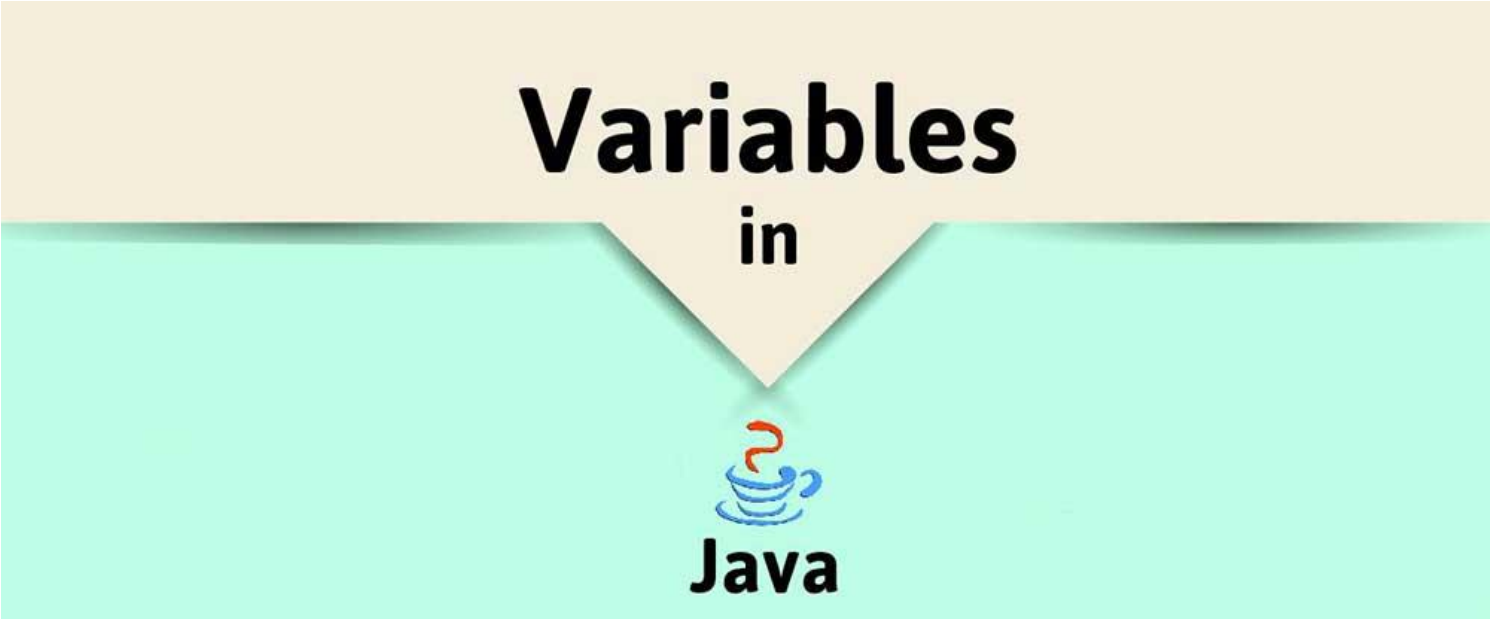
جلسه هشتم - آموزش انواع متغیر ها در جاوا



متغیر در اصطلاح به ظرفی میگویند که میتواند مقادیر مختلفی را درون خود ذخیره نماید. متغیر ها در جاوا هم دقیقا همینطور هستند. یعنی در زبان های برنامه نویسی مفهومی به نام متغیر داریم که انواع مختلفی دارد و با توجه به نوعی که برای آنها تعریف میکنیم میتوانند مقادیر خاصی را درون خود داشته باشند. برای آشنایی با انواع متغیر ها در جاوا با **برنامه چی** همراه باشید.

در این جلسه چه چیزی یاد خواهیم گرفت؟

مفهوم متغیر ها در جاوا

تعریف متغیر ها در جاوا

متغیر های محلی (Local Variables) در جاوا

متغیر های نمونه (Instance Variables) در جاوا

متغیر های استاتیک یا کلاس(Static/Class Variables)

در جلسه بعد چه چیزی یاد میگیریم؟

مفهوم متغیر ها در جاوا

متغیر قسمتی از حافظه است که نامگذاری شده و برنامه ما میتواند آن را دستکاری نماید. همه متغیر ها در جاوا نوع خاصی دارند، که همین نوع خاص باعث میشود سایز و ظاهر موری آن متغیر ها، بازه مقادیری که میتواند در آن موری ذخیره شود و همه انواع اعمالی که میتوانید روی آن متغیر ها اعمال کنید را مشخص میکند.

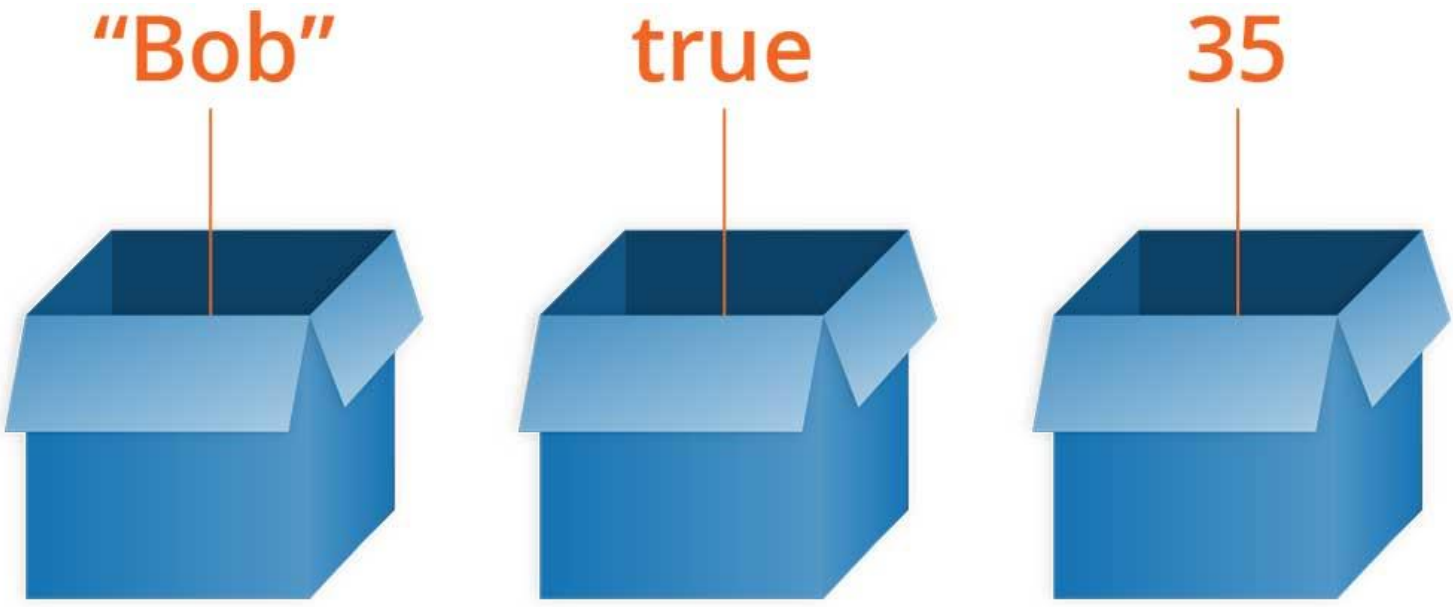
دیدن مقاله در سایت

تعریف متغیر ها در جاوا

شما باید قبل از اینکه بتوانید از انها استفاده کنید، با نحوه تعریف انواع متغیر ها در جاوا آشنا باشید. در ادامه یک فرم ساده از تعریف متغییر ها در جاوا را میبینید:

```
data type variable [ = value][, variable [ = value] ...] ;
```

data type در اینجا همان دیتا تایپ هایی هستند که در جلسه قبل درباره آنها صحبت کردیم. variable هم اینجا جایگزین نامی میشود که برای متغیر خودمان انتخاب میکنیم.



در ادامه نمونه های واقعی از تعریف کردن متغیر ها در جاوا و همچنین مقدار دهی اولیه آنها را میبینیم.

مثال:

```
int a, b, c;         // 3 Adad int Taarif mikonim be nam haye a, b, and c.
int a = 10, b = 10;  // Mesali az Initialize
byte B = 22;         // Meghdar dehi avaliye yek motaghayere byte be name B.
double pi = 3.14159; // moarefi va meghdar dehi yek motaghayer be name PI.
char a = 'a';        // Motaghayere char be name a ba meghdare 'a' Initialize mishavad.
```

در این درس انواع مختلف متغیر ها در جاوا را یاد میگیریم. در زبان برنامه نویسی جاوا سه نوع متغیر مختلف داریم:

- **متغیر های محلی (Local Variables)**
- **متغیر های نمونه (Instance Variables)**
- **متغیر های کلاس یا استاتیک (Class/Static Variables)**

متغیر های محلی (Local Variables) در جاوا

- متغیر های محلی در متد ها، Constructor ها و بلاک ها تعریف میشوند.
- متغیر های محلی هنگامی ساخته میشود که متدها، Constructor ها و یا بلاک ها اجرا شده باشند و تنها تا زمانی که این متدها، Constructor ها و یا بلاک ها در حال اجرا باشند، وجود خواهند داشت. یعنی مثلا با تمام شدن کارهای متد، این متغیر ها هم از بین میروند.
- تعیین کننده های دسترسی (Access Modifiers) نمیتوانند برای متغیر های محلی به کار بروند.
- Local Variable ها فقط برای اجزایی که داخل متدها، متدهای سازنده و یا بلاک هایی که درون آنها تعریف شده اند، قابل رویت هستند.
- Local Variable ها درون حافظه داخلی Stack قرار میگیرند. (با این نوع حافظه بعدا آشنا خواهید شد).
- برای متغیر های محلی هیچگونه مقدار پیشفرضی وجود ندارد. در نتیجه برای اینکه در حین اجرای برنامه به خطای Null Pointer Exception برنخورید، حتما باید قبل از اولین استفاده آنها را مقدار دهی اولیه (initialize) کنید.

مثال

در مثال زیر، age یک متغیر محلی است که درون متد ()pupAge تعریف شده است. بنابراین حوزه دید (scope) این متغیر هم محدود به همین متد میباشد. (توضیح کوتاه: مثلا متغیر های محلی اگر درون یک متد تعریف بشوند، فقط برای کدهایی که درون همان متد قرار دارند قابل رویت خواهند بود. به این قانون، حوزه دید یا scope متغیر ها در جاوا گفته میشود).

```
public class Test {
  public void pupAge() {
    int age = 0;
    age = age + 7;
    System.out.println(&amp;amp;quot;Puppy age is : &amp;amp;quot; + age);
  }

  public static void main(String args[]) {
    Test test = new Test();
    test.pupAge();
  }
}
```

کدهای بالا باید نتیجه خروجی زیر را تولید کنند:

```
Puppy age is: 7
```

مثال

در مثال زیر از متغیر محلی age استفاده کرده ایم ولی آن را مقداردهی اولیه (Initialize) نکرده ایم. به همین دلیل برنامه در هنگام کامپایل شدن یک خطا به ما نشان میدهد.

```
public class Test {
  public void pupAge() {
    int age;
    age = age + 7;
    System.out.println(&amp;amp;quot;Puppy age is : &amp;amp;quot; + age);
  }

  public static void main(String args[]) {
    Test test = new Test();
    test.pupAge();
  }
}
```

مثال بالا باید خروجی زیر را تولید کند:

```
Test.java:4:variable number might not have been initialized
age = age + 7;
      ^
\ error
```

متغیر های نمونه (Instance Variables) در جاوا

- متغیر های نمونه درون کلاس تعریف میشوند اما باید خارج از متد، Constructor یا هر بلاکی باشند.
- وقتی یک مقدار فضا در حافظه heap به یک آبجکت اختصاص داده میشود، برای مقدار هر متغیر نمونه، یک slot از حافظه کنار گذاشته میشود.
- متغیر های نمونه زمانی که یک آبجکت با استفاده از کلمه کلیدی new به وجود می آید، ساخته میشوند. زمانی هم از بین میروند که آن object از بین رفته باشد.
- متغیر های نمونه مقادیری را نگهداری میکنند که قرار است در چند متد، متد سازنده و یا بلاک مختلف از آنها استفاده بشود. یا اینکه آبجکت های ضروری که قرار است در کل کلاس در دسترس باشند را درون خود نگهداری میکنند.
- متغیر های نمونه میتوانند در سطح کلاس، قبل و بعد از جایی که استفاده شده اند تعریف بشوند.
- Access Modifier ها میتوانند برای متغیر های نمونه تعریف بشوند.
- Instance Variable ها برای تمام متد ها، Constructor ها و بلاک های درون کلاس قابل رویت هستند. به صورت معمول به شما پیشنهاد میکنیم که سطح دسترسی این متغیر ها را private (خصوصی) تعریف کنید. (درباره سطوح دسترسی در جلسه های آینده توضیح خواهیم داد). به هرحال با استفاده از access modifier ها میتوانید این نوع از متغیر ها را برای sub-class ها هم قابل رویت کنید.
- متغیر های نمونه دارای مقدار پیشفرض هستند. برای عدد ها مقدار پیشفرض 0 است. برای Boolean ها مقدار false میباشد. برای آبجکت ها نیز این مقدار برابر null میباشد. شما میتوانید این متغیر ها را در زمان تعریف کردنشان یا در متد Constructor مقدار دهی کنید.

- متغیر های نمونه میتوانند مستقیما با صدا زدن نام آنها در کلاس، فراخوانی بشوند. به هر حال در ون متد های static (زمانی که به متغیر های نمونه دسترسی داده شده باشد)، آنها باید با استفاده از نام کاملشان صدا زده بشوند. به این صورت: ObjectReference.VariableName.

مثال

```
import java.io.*;
public class Employee {

  // this instance variable is visible for any child class.
  public String name;

  // salary  variable is visible in Employee class only.
  private double salary;

  // The name variable is assigned in the constructor.
  public Employee (String empName) {
    name = empName;
  }

  // The salary variable is assigned a value.
  public void setSalary(double empSal) {
    salary = empSal;
  }

  // This method prints the employee details.
  public void printEmp() {
    System.out.println(&amp;amp;quot;name  : &amp;amp;quot; + name );
    System.out.println(&amp;amp;quot;salary :&amp;amp;quot; + salary);
  }

  public static void main(String args[]) {
    Employee empOne = new Employee(&amp;amp;quot;Ransika&amp;amp;quot;);
    empOne.setSalary(1000);
    empOne.printEmp();
  }
}
```

این کدها خروجی زیر را تولید میکنند.

```
name  : Ransika
salary :1000.0
```

متغیر های استاتیک یا کلاس (Static/Class Variables)

- متغیر های کلاس که به عنوان متغیر های static هم شناخته میشوند، با کلیدواژه static در سطح کلاس تعریف میشوند. اما این تعریف ها خارج از متد، Constructor و بلاک ها تعریف میشوند.
- در هر کلاسی فقط یک کپی از هر متغیر استاتیک وجود خواهد داشت، و به این ربط ندارد که چند آبجکت مختلف از روی آن ساخته خواهد شد.
- متغیر های نمونه به ندرت استفاده میشوند مگر اینکه بخواهیم آنها را به عنوان ثابت تعریف کنیم. متغیر های constant یا ثابت متغیر هایی هستند که به عنوان public، private، final و static تعریف میشوند. مقدار متغیرهای ثابت هیچوقت از آن چیزی که در زمان ایجاد کردن برای آنها تعریف کرده ایم تغییر نخواهد کرد.
- متغیر های static درون حافظه استاتیک قرار میگیرند. معمولا نوع استاتیک را همراه با کلمه کلیدی final تعریف میکنند.
- متغیر های استاتیک هنگامی که برنامه شروع میشود ایجاد میشوند و با توقف برنامه از بین خواهند رفت.
- اسکوپ یا حوزه دید این متغیر ها در جاوا همانند متغیرهای نمونه میباشد. اما به هرحال بیشتر متغیرهای استاتیک بصورت public تعریف میشوند زیرا همه کاربران کلاس باید به آنها دسترسی داشته باشند.
- مقادیر پیشفرض هم برای این متغیر ها در جاوا مانند Instance Variable ها میباشد. برای عددها 0، برای Boolean ها false، و برای ارجاع های آجکت این مقدار null میباشد. مقدار دهی برای آنها را هم میتوانید در زمان تعریف کردن یا در متد های Constructor انجام بدهید. علاوه بر اینها، مقادیر میتوانند در بلاک های استاتیک مخصوص مقدار دهی اولیه (Initializer) به این متغیر ها در جاوا نسبت داده بشوند.
- برای دسترسی به متغیر های استاتیک، باید نام آنها را به همراه کلاسی که در آن تعریف شده اند صدا بزنیم. به این صورت: ClassName.VariableName
- وقتی متغیر های کلاس را به عنوان public static final تعریف کنیم، پس نام متغیرها (یا همان ثابت ها)، باید تماما با حروف بزرگ نوشته بشوند. اگر متغیر های استاتیک، public و final نباشند، قوانین نامگذاری آنها مانند متغیر های local و نمونه میباشد.

مثال

```
import java.io.*;
public class Employee {

  // salary  variable is a private static variable
  private static double salary;

  // DEPARTMENT is a constant
  public static final String DEPARTMENT = &amp;amp;quot;Development &amp;amp;quot;;

  public static void main(String args[]) {
    salary = 1000;
    System.out.println(DEPARTMENT + &amp;amp;quot;average salary:&amp;amp;quot; + salary);
  }
}
```

این کدها باید خروجی زیر را تولید کنند:

```
Development average salary:1000
```

در جلسه بعد چه چیزی یاد میگیریم؟

شما در این درس از access modifier ها استفاده کردید. در جلسه بعد درباره تعیین کننده های سطوح دسترسی (Access Modifier) ها و Non-Access Modifier ها با جزئیات بحث خواهیم کرد.

جلسه قبل: دیتا تایپ چیست؟

جلسه بعد: تعیین سطح دسترسی جاوا