آموزش برنامه نویسی جاوا به زبان ساده

```
1.  اولین برنامه //
2.  class Examp1 {
3.          public static void main(String[] args) {
4.          System.out.println(“Examp1!”);
5.      }
6.  }
```

خروجی برنامه

```
Examp1!
```

class Examp1{ … }

در جاوا، هر برنامه با تعریف کلاس شروع می شود. در برنامه بالا، **Examp1** نام کلاس است و تعریف کلاس برابر است با:

```
class Examp1{

 ……..

……..

}
```

به یاد داشته باشید که هر برنامه جاوا دارای تعریف کلاس است و نام کلاس باید با نام فایل در جاوا یکی باشد.

public static void main(String[] args) { … }

خط بالایک تابع اصلی است. هر برنامه در جاوا باید یک تابع اصلی داشته باشد. کامپایلر جاوا اجرای کد را از تابع اصلی شروع می کند.

به یاد داشته باشید که تابع main نقطه ورود برنامه جاوا و اجباری است. تابع main در جاوا برابر است با:

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

{

……

.....

 }
```

```
System.out.println("Examp1“);
```

کد بالا رشته داخل علامت نقل قول را در خروجی استاندارد (صفحه نمایش) چاپ می کند. توجه داشته باشید، این خط از کد در داخل تابع اصلی است که در داخل تعریف کلاس قرار دارد.

متغیرهای جاوا

متغیر مکانی برای ذخیره داده ها در حافظه کامپیوتر است. برای نشان دادن محل ذخیره سازی ، به هر متغیر باید یک اسم (شناسه) یکتا داده شود.

چگونه متغیرها را در جاوا تعریف کنیم؟

مثال

```
1.  int A = 60;
```

درمثال بالا ، A یک متغیر از نوع داده int (عدد صحیح) است و مقدار ۶۰ نیز به آن اختصاص داده شده است.

در این مثال، هنگام تعریف متغیر به آن مقدار اختصاص دادیم اما این کار اجباری نیست. می توانید متغیرها را بدون تعیین مقدار تعریف و بعدا مقدار دهی کنید. مثلا،

```
int A;
speedLimitA = 60;
```

می توان مقدار متغیر را در برنامه تغییر داد ، به همین خاطر به آن "متغیر" می گویند. مثلا،

```
int speedLimitA = 60;
… .. …
A = 120;
```

جاوا یک زبان با نوع استاتیک است. یعنی همه متغیرها قبل از استفاده باید تعریف شوند.

همچنین، نمی توان نوع داده ای متغیر را در برنامه تغییر داد. برای مثال شما قادر به انجام کار زیر نیستید:

```
int A = 60;
… .. …
float A;
```

قوانین نامگذاری متغیرها در جاوا

زبان برنامه نویسی جاوا مانند سایر زبان ها مجموعه ای از قوانین و قراردادهای خاص برای نامگذاری متغیرها دارد:

۱- متغیرها در جاوا حساس به حروف کوچک و بزرگ هستند.
۲- نام متغیر دنباله ای از حروف و ارقام یونیکد است و می تواند با حرف ، $ یا _ شروع شود. با این حال ، آغاز اسم متغیر با حرف معمولی تر است. همچنین ، نام متغیر نمی تواند شامل فضای خالی باشد.
۳- هنگام تعریف متغیرها ، اسمی را انتخاب کنید که معقول باشد. به عنوان مثال ، firstname ، lastname از اسامی f ، l با مفهوم تر هستند.
۴- اگر یک کلمه را برای نام متغیر انتخاب کردید ، همه حروف را کوچک بنویسید. به عنوان مثال ، بهتر است از firstname استفاده کنید نه FIRSTNAME ، یا Firstname .
۵- اگر نام متغیر با بیش از یک کلمه را انتخاب کردید، از تمام حروف کوچک برای کلمه اول استفاده کنید و حروف اول هر کلمه بعدی را بزرگ بنویسید. به عنوان مثال ، firstName .

در زبان برنامه نویسی جاوا ٤ نوع متغیر وجود دارد:

۱- متغیرهای نمونه (متغیر های غیر استاتیک)
۲- متغیرهای کلاس (متغیر های استاتیک)
۳- متغیرهای محلی
۴- مولفه ای

انواع داده های اصلیJava

همانطور که در بالا گفته شد ، جاوا یک زبان با نوع استاتیک است. یعنی همه متغیرها قبل از استفاده باید تعریف شوند.

```
int A;
```

در اینجا A یک متغیر است و نوع داده متغیر int است. نوع داده int تعیین می کند متغیر فقط می تواند شامل اعداد صحیح باشد.

به عبارت ساده ، نوع داده متغیر مقادیری را که یک متغیر می تواند ذخیره کند تعیین می کند. ۸ نوع داده از پیش تعریف شده در زبان برنامه نویسی جاوا وجود دارد که به عنوان انواع داده های اصلی شناخته می شوند.

علاوه بر انواع داده های اصلی ، انواع داده های ارجاعی نیز در جاوا وجود دارد که بعدا با آن ها آشنا خواهید شد.

نوع داده اصلی

۱- **Boolean**

- نوع داده boolean دارای دو مقدار ممکن است ، true یا false است.
- مقدار پیش فرض.false :
- معمولا برای شرایط صحیح / غلط استفاده می شوند.

مثال:

```
1.  class BooleanExample {
2.  public static void main(String[] args) {
3.
4.  boolean flag = true;
5.  System.out.println(flag);
6.  }
7.  }
```

خروجی

```
true
```

۲- **byte**

- نوع داده بایت می تواند مقادیر بین ۱۲۸- تا ۱۲۷ داشته باشد.
- اگر به یقین مقدار متغیر در محدوده [۱۲۸،۱۲۷-] باشد ، به جای نوع داده ای int یا دیگر اعداد صحیح استفاده می شود.
- مقدار پیش فرض: ۰

مثال:

```
1.  class ByteExample {
2.  public static void main(String[] args) {
3.
4.  byte range;
5.
6.  range = 124;
7.  System.out.println(range);
8.
9.  // Error code below. Why?
10. // range = 200
```

```
11. }
12. }
```

خروجی

```
۱۲۴
```

۳- short

- نوع داده کوتاه می تواند مقادیری از ۳۲۷۶۸- تا ۳۲۷۶۷ داشته باشد (عدد صحیح مکمل دو بیتی ۱۶ بیتی امضا شده است).
- اگر به یقین مقدار متغیر در محدوده [۳۲۷۶۸-, ۳۲۷۶۷] باشد ، به جای انواع دیگر داده های صحیح استفاده می شود.
- مقدار پیش فرض: ۰

مثال:

```
1.  class ShortExample {
2.  public static void main(String[] args) {
3.
4.  short temperature;
5.
6.  temperature = -200;
7.  System.out.println(temperature);
8.
9.  }
10. }
```

خروجی

```
-۲۰۰
```

۴- int

- نوع داده int می تواند مقادیری از $۲^{۳۱}$ تا $۲^{۳۱}-۱$ داشته باشد.
- اگر از Java 8 به بعد استفاده می کنید ، می توانید از عدد صحیح ۳۲ بیتی بدون علامت با حداقل مقدار ۰ و حداکثر مقدار $۲^{۳۱}-۱$ استفاده کنید.
- مقدار پیش فرض: ۰

مثال:

```
1.  class IntExample {
2.  public static void main(String[] args) {
3.
4.  int range = -4250000;
5.  System.out.println(range);
6.  }
7.  }
```

خروجی

```
-۴۲۵۰۰۰۰
```

۵- **long**

- نوع داده های long می تواند مقادیری از $۲^{۶۳}$ تا $۲^{۶۳}-۱$ داشته باشد.
- اگر از Java 8 یا بالاتر استفاده می کنید ، می توانید از عدد صحیح ۶۴ بیتی بدون علامت با حداقل مقدار ۰ و حداکثر مقدار $۲^{۶۴}-۱$ استفاده کنید.
- مقدار پیش فرض: ۰

مثال:

```
1. class LongExample {
2. public static void main(String[] args) {
3.
4. long range = -42332200000L;
5. System.out.println(range);
6. }
7. }
```

خروجی

```
-۴۲۳۳۲۲۰۰۰۰۰
```

توجه کنید ، استفاده از L در پایان ۴۲۳۳۲۲۰۰۰۰۰- نشان دهنده کلمه long است.

۶- **double**

- نوع داده double دقت ۶۴ بیت دارد.
- هرگز نباید برای مقادیر دقیق مانند currency استفاده شود.
- مقدار پیش فرض: ۰٫۰(d۰٫۰)

مثال:

```
1. class DoubleExample {
2. public static void main(String[] args) {
3.
4. double number = -42.3;
5. System.out.println(number);
6. }
7. }
```

خروجی

```
-۴۲,۳
```

۷- **float**

- نوع داده float دقت ۳۲ بیت دارد.
- هرگز نباید برای مقادیر دقیق مانند currency استفاده شود.
- مقدار پیش فرض: ۰٫۰(f۰٫۰)
- مثال:

```
1.  class FloatExample {
2.  public static void main(String[] args) {
3.
4.  float number = -42.3f;
5.  System.out.println(number);
6.  }
7.  }
```

خروجی

```
-۴۲,۳
```

توجه داشته باشید که در برنامه بالا از f ۴۲٫۳- به جای ۴۲٫۳- استفاده کردیم. چون ۴۲٫۳- از نوع double است. برای اینکه به کامپایلر بگویید ۴۲٫۳- از نوع float است و نهdouble ، باید از f یا F استفاده کنید.

۸- **char**

- کاراکتر یونیکد ۱۶ بیتی است.
- حداقل مقدار '\u0000' و حداکثر مقدار '\uffff' است.
- مقدار پیش فرض '\u0000' :

مثال:

```
1.  class CharExample {
2.  public static void main(String[] args) {
3.
4.  char letter = '\u0051';
5.  System.out.println(letter);
6.  }
7.  }
```

خروجی

```
Q
```

مقدار یونیکد ’/u0051‘ برابر با Q است.

مثال دیگر:

```
1.  class CharExample {
2.  public static void main(String[] args) {
3.
4.  char letter1 = ‘9’;
5.  System.out.println(letter1);
6.
7.  char letter2 = 65;
8.  System.out.println(letter2);
9.  }
10. }
```

خروجی

```
۹
A
```

هنگامی که letter1 را چاپ می کنید ، خروجی ۹ می شود زیرا به letter1 کاراکتر ۹ اختصاص داده شده است.

هنگامی که letter2 را چاپ می کنید ، خروجی A می شود زیرا مقدار اسکی حرفA ، ۶۵ است.

جاوا همچنین از طریق کلاس java.lang.String پشتیبانی رشته های کاراکتر را فراهم می کند. در اینجا چگونگی ایجاد شی String در جاوا آورده شده است:

```
myString = “Programming is awesome”;
```

لیترال در جاوا

برای درک لیترال ، بیایید مثالی بزنیم که مقداری را به یک متغیر اختصاص دهیم.

```
boolean flag = false;
```

در اینجا،

- boolean – نوع داده است.
- flag – متغیر است

- false – لیترال است.

لیترال نمایانگر کد منبع با یک مقدار ثابت است.

مقادیری مانند۱/۵، ۴، true، ’\u0050‘که مستقیما در برنامه ظاهر می شوند بدون اینکه نیاز به محاسبه داشته باشند لیترال هستند.

در مثال بالا ، flagیک متغیر است. از آنجا که از نوع boolean است ، ممکن است false یا true باشد. کامپایلر برای درک آن نیاز به محاسبه دارد. با این حال ، لیترال مانند ۵- ، ”a“، trueنشان دهنده مقدار ثابت است.

لیترال صحیح

- برای مقدار دهی اولیه متغیرهای انواع داده هایint ، byte، short، intو long از لیترال صحیح استفاده می شود.
- اگر لیترال صحیح با l یا L تمام شود ، از نوع long است. نکته: بهتر است از L به جای l استفاده کنید.

```
// Error! literal 42332200000 of type int is out of range
long myVariable1 = 42332200000;
// ۴۲۳۳۲۲۰۰۰۰۰L is of type long, and it’s not out of range
long myVariable2 = 42332200000L;
```

- لیترال صحیح را می توان در سیستم های دسیمال ، هگزا دسیمال و دودویی بیان کرد.
- اعدادی که با پیشوند 0x شروع می شوند نشانگر هگزا دسیمال هستند. به همین ترتیب ، اعدادی که با پیشوند 0b شروع می شوند ، دودویی را نشان می دهند.

```
// decimal
int decNumber = 34;
// ۰x represents hexadecimal
int hexNumber = 0x2F;
// ۰b represents binary
int binNumber = 0b10010;
```

لیترال ممیز شناور

- برای مقدار دهی اولیه ی متغیرهای از نوع float و double ، از لیترال ممیز شناور استفاده می شود.
- اگر یک لیترال ممیز شناور با f یا F به پایان برسد ، از نوع float است. در غیر این صورت ، از نوع doubleاست. نوع double می تواند اختیاری با D یا d پایان یابد. با این حال ، لازم نیست.
- همچنین می توانند در اعداد نماد علمی با استفاده از E یا e بیان شوند.

```
1.  class DoubleExample {
2.  public static void main(String[] args) {
3.
4.  double myDouble = 3.4;
5.  float myFloat = 3.4F;
6.
7.  // ۳,۴۴۵*۱۰۸۲
8.  double myDoubleScientific = 3.445e2;
9.  System.out.println(myDouble);
10. System.out.println(myFloat);
11. System.out.println(myDoubleScientific);
12. }
13. }
```

خروجی

```
1.  ۳,۴
2.  ۳,۴
3.  ۳۴۴,۵
```

لیترال های کاراکتر و رشته

- حاوی کاراکترهای یونیکد (UTF-16) هستند.
- برای لیترال کاراکتر ، از نقل قول منفرد استفاده می شود. مثلا ’a‘ یا.’\u0111‘
- برای لیترال رشته ای ، از نقل قول مضاعف استفاده می شود. به عنوان مثال ، ”programming“، ”java8 “
- جاوا همچنین از چند دنباله فرار ویژه پشتیبانی می کند. به عنوان مثال ، \bبرایbackspace ، \tبرای انتقال مکان نما به ۸ مکان بعد ، \nبرای انتقال مکان نما به خط بعد ، \fبرای انتقال کنترل به صفحه ی جدی ، \rبرای ابتدای سطر جاری، \” برای چاپ علامت نقل قول ، \’ برای چاپ نقل قول منفرد ، و \\ برای چاپ بک اسلش.

```
1.  class DoubleExample {
2.  public static void main(String[] args) {
3.
4.  char myChar = ‘g’;
5.  char newLine = ‘\n’;
6.  String myString = “Java 8”;
7.
8.  System.out.println(myChar);
9.  System.out.println(newLine);
10. System.out.println(myString);
11. }
12. }
```

خروجی

```
g

Java
```