# آموزش جاوا – قسمت اول

# معرفی زبان جاوا

- جاوا زبانی شی گراست، که می تواند روی پلتفرم های مختلف مانند ویندوز و Mac OS و ورژن های مختلف unix اجرا شود.
- زبان جاوا در سال 1995 به وجود آمد.
- آخرین ورژن آن جاوا، SE 8 نام دارد.

# نوشتن اولین برنامه با جاوا

```
public class HelloWorld{

    public static void main(String []args){
        System.out.println("Hello World");
    }
}
```

# ساختار کدنویسی در جاوا

- حساسیت به حروف کوچک و بزرگ ( Case Sesentive ). مثال: Hello و hello باهم تفاوت دارند.
- نام کلاس ها باید با حرف اول بزرگ شروع شوند. مثال: MainClass
- نام متد ها باید با حرف اول کوچک شروع شوند. مثال: ()doThat
- نام فایلی که ذخیره می کنیم حتما باید هم نام کلاس باشد.

# Java Identifiers

- هر نامی که برای Class ها، Variableها، Method ها به کار می رود، Identifier نام دارد.

# قواعد نام گذاری Identifiers

- حتما باید با یکی از حروف الفبای انگلیسی ( A to Z ) یا علامت $ یا _ شروع شوند.
- بعد از اولین حرف، هر ترکیبی از کاراکتر ها را می توان استفاده کرد.
- یک keyword نمی تواند به عنوان Identifier استفاده شود.
- Identifier ها به حروف کوچک و بزرگ حساس هستند.

# مثال

- نام گذاری صحیح: age, $salary, _value, __1_value
- نام گذاری اشتباه: 123abc, -salary

# مفهوم پکیج ( package ) در جاوا

- در جاوا برای جلوگیری از تداخل در اسامی کلاس ها، اینترفیس ها و جستجوی راحت بین آن ها و همچنین جلوگیری از دسترسی غیر مجاز ، از مفهوم پکیج استفاده می شود.

```
com.example.sshahini.myapplication.school.Student student1;

com.example.sshahini.myapplication.university.Student student2;
```

# Modifiers

- به کلماتی گفته می شود که می توانند وضعیت و نوع کلاس ها، متدها و غیره را عوض کنند.
- Access modifiers: برای تعیین سطح دسترسی مورد استفاده قرار می گیرند.

**public, private, protected, default**

- Non-Access modifiers: برای کارهایی غیر تعیین سطح دسترسی مورد استفاده قرار می گیرند. **final,**

**abstract, static**

- Modifier ها با حرف کوچیک نوشته می شوند.

# Access modifiers

- private: فقط در کلاس خود قابل دیدن هستند.
- public: در همه جا قابل دیدن هستند.
- protected: فقط برای پکیج خود و تمامی Subclass ها قابل دیدن هستند.
- default: برای پکیج فقط قابل دیدن هستند.

# Non Access Modifiers

- static: برای تعریف Class method و Class variable مورد استفاده قرار می گیرد.
- final: زمانی استفاده می شود که دیگر نمی خواهیم به متغیری اجازه تغییر بدهیم.
- abstract: برای تعریف کلاس یا متد Abstract به کار می رود.

# تعریف کلاس در جاوا

Modifier

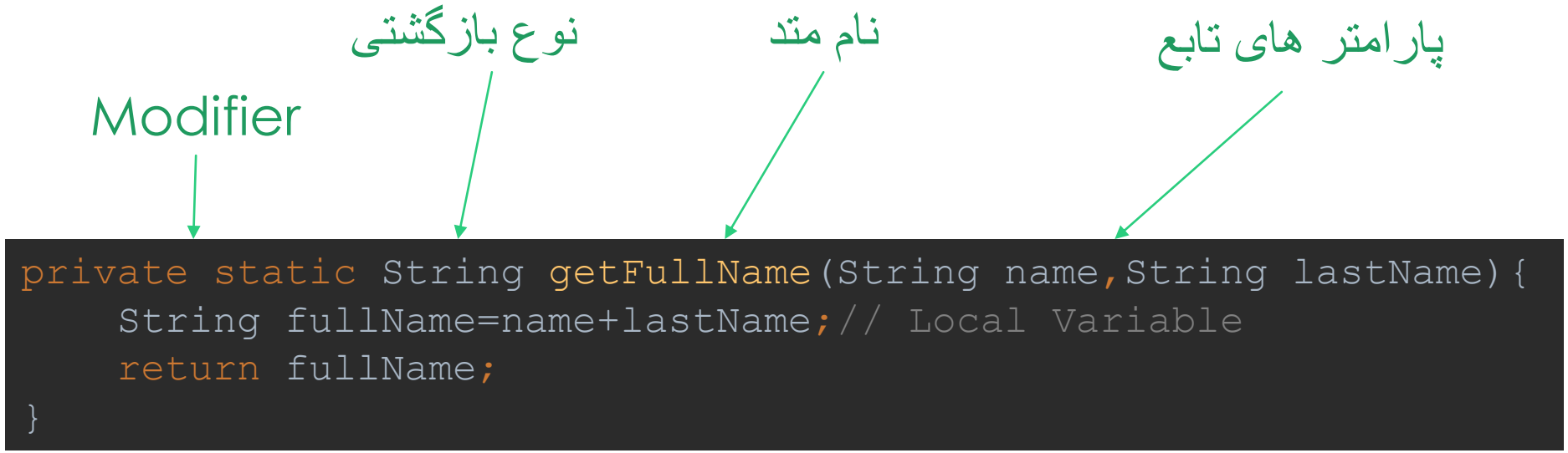
Class keyword

نام کلاس

```
public class Student {
    public static int MAXIMUM_SCORE=20;//Class variable
    private String name;
    private String lastName;// Instance Variable

    private String getFullName(){
        String fullName=name+lastName;// Local Variable
        return fullName;
    }
}
```

# تعریف متد در جاوا




```
private static String getFullName(String name,String lastName){
    String fullName=name+lastName;// Local Variable
    return fullName;
}
```

# انواع داده در جاوا

- Primitive Data Types: در جاوا 8 نوع داده ی اصلی وجود دارند.
- Reference/Object Data Types: به داده هایی گفته می شوند که توسط Constructor کلاس خود ساخته می شوند.

# Primitive Data Types

- byte
  - کمترین مقدار: -128
  - بیشترین مقدار: 127
  - مقدار پیش فرض: 0
- short
  - کمترین مقدار: -32,768
  - بیشترین مقدار: 32,767
  - مقدار پیش فرض: 0

# Primitive Data Types

- int
  - کمترین مقدار: -2,147,483,648
  - بیشترین مقدار: 2,147,483,647
  - مقدار پیش فرض: 0
- long
  - کمترین مقدار: -9,223,372,036,854,775,808
  - بیشترین مقدار: 9,223,372,036,854,775,807
  - مقدار پیش فرض: 0L

# Primitive Data Types

- float
  - برای داده های اعشاری از آن استفاده می شود.
  - هیچوقت از آن برای مقادیر دقیق مثل

    واحد پولی استفاده نکنید.
  - مقدار پیش فرض: 0.0f
- double
  - مقدار پیش فرض: 0.0d
  - 64 بیت

# Primitive Data Types

- boolean
  - فقط دو مقدار می گیرد: true و false
  - زمانی استفاده می شود که بخواهیم صحیح بودن یا نبودن یک وضعیت را بسنجیم.
- char
  - 16 bit Unicode character
  - کمترین مقدار: \u0000 یا 0
  - بیشترین مقدار: \uffff یا 65,535

# Reference Data Types

- توسط Constructor کلاس خود ساخته می شوند.
- مقدار پیش فرض آن ها null است.
- مثال:

```
Student student=new Student();
```

# ایجاد متغیر

نوع داده

مقدار

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

نام متغیر

# انواع متغیرها در جاوا

- Local Variables
- Class Variables (Static Variables)
- Instance Variables (Non-static variables)

# عملگرهای پایه ( Basic operators )

- عملگرهای حسابی ( Arithmetic Operators )
- عملگرهای رابطه ای (Relational Operators )
- عملگرهای بیتی ( Bitwise Operators )
- عملگرهای منطقی ( Logical Operators )
- عملگرهای انتسابی ( Assignment Operators )
- عملگرهای متفرقه ( Misc Operators )

# عملگرهای حسابی

| عمگلر ها | رفتار و مثال |
| --- | --- |
| + | (جمع) Adds values on either side of the operator<br>**Example:** A + B will give 30 |
| - | (تفریق) Subtracts right hand operand from left hand operand<br>**Example:** A - B will give -10 |
| * | (ضرب) Multiplies values on either side of the operator<br>**Example:** A * B will give 200 |
| / | (تقسیم) Divides left hand operand by right hand operand<br>**Example:** B / A will give 2 |
| % | (باقی مانده) Divides left hand operand by right hand operand and returns remainder<br>**Example:** B % A will give 0 |
| ++ | (افزایش) Increases the value of operand by 1<br>**Example:** B++ gives 21 |
| -- | (کاهش) Decreases the value of operand by 1<br>**Example:** B-- gives 19 |

# عملگرهای رابطه ای

| عملگر ها | رفتار و مثال |
| --- | --- |
| == | (مساوی) Checks if the values of two operands are equal or not, if yes then condition becomes true.<br>**Example:** (A == B) is not true. |
| != | (نا مساوی) Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.<br>**Example:** (A != B) is true. |
| > | (بزرگتر از)Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.<br>**Example:** (A > B) is not true. |
| < | (کوچک تر از)Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.<br>**Example:** (A < B) is true. |
| >= | >= (بزرگتر مساوی)Checks if the value of left operand is greater than or equal to the value of right operand.<br>**Example** (A >= B) is not true. |
| <= | <= (کوچکتر مساوی)Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.<br>**Example** (A <= B) is true. |

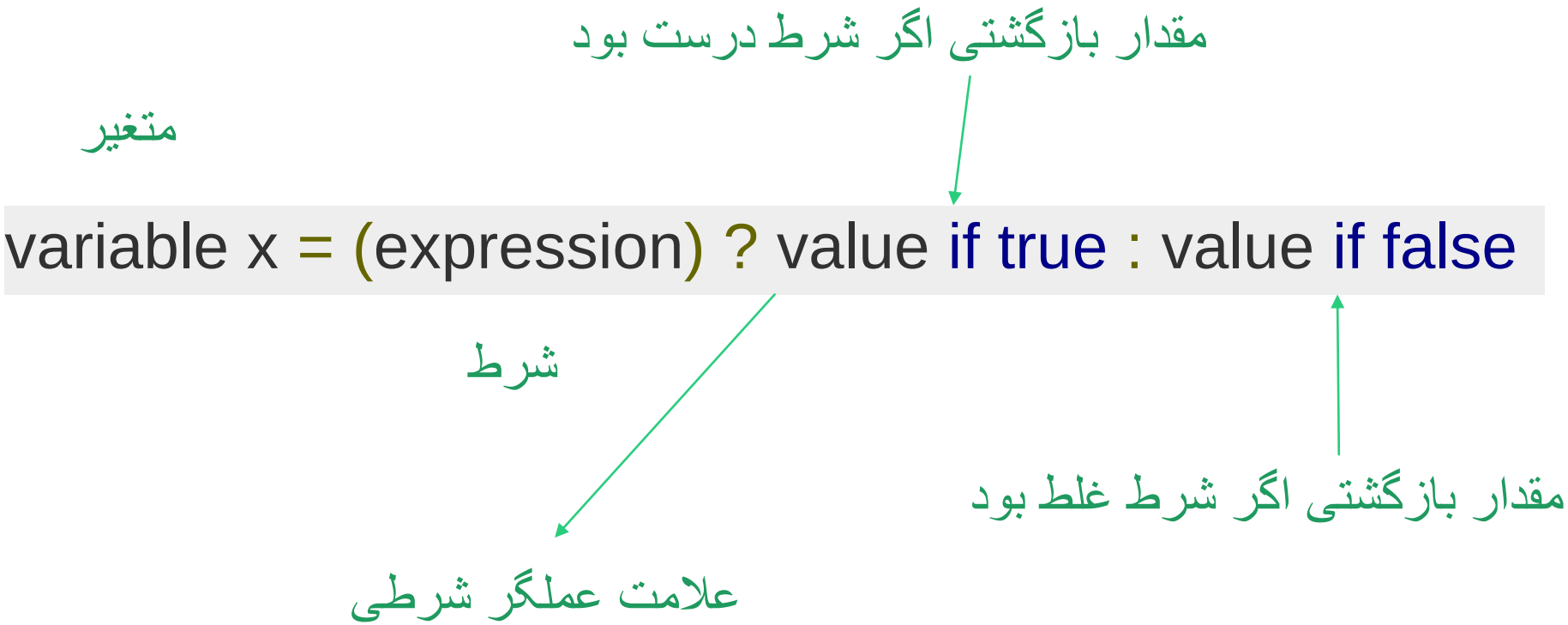
# عملگرهای منطقی

| عملگر ها | رفتار و مثال |
| --- | --- |
| && | **(logical and)** Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.<br>**Example** (A && B) is false. |
| \|\| | **(logical or)** Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.<br>**Example** (A \|\| B) is true. |
| ! | **(logical not)** Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.<br>**Example** !(A && B) is true. |

# عملگرهای انتسابی

| عملگر ها | رفتار و مثال |
| --- | --- |
| = | Simple assignment operator, Assigns values from right side operands to left side operand.<br>**Example:** C = A + B will assign value of A + B into C |
| += | Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand.<br>**Example:** C += A is equivalent to C = C + A |
| -= | Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand.<br>**Example:** C -= A is equivalent to C = C - A |
| *= | Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand.<br>**Example:** C *= A is equivalent to C = C * A |
| /= | Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand<br>**Example** C /= A is equivalent to C = C / A |
| %= | Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand.<br>**Example:** C %= A is equivalent to C = C % A |

# عملگر شرطی


مقدار بازگشتی اگر شرط درست بود
متغیر
variable x = (expression) ? value if true : value if false
شرط
مقدار بازگشتی اگر شرط غلط بود
علامت عملگر شرطی

# مثال

```
public class Test {

    public static void main(String args[]){
        int a, b;
        a = 10;
        b = (a == 1) ? 20: 30;
        System.out.println( "Value of b is : " +  b );  //عدد 30 چاپ خواهد شد

        b = (a == 10) ? 20: 30;
        System.out.println( "Value of b is : " + b ); //عدد 20 چاپ خواهد شد
    }
}
```

# عملگر instanceof

متغیر

کلاس مورد نظر

```
( Object reference variable ) instanceof (class/interface type)
```

علامت عملگر instanceof

# مثال

```
public class Test {

    public static void main(String args[]){
        String name = "James";
        // following will return true since name is type of String
        boolean result = name instanceof String;
        System.out.println( result ); // مقدار ترو برخواهد گشت
    }
}
```

# حلقه یا Loop

- تعریف: زمانی که بخواهیم یک سلسله مراتب کار را به صورت تکراری انجام دهیم از حلقه ها استفاده می کنیم.
- مثال: چاپ اعداد 1 تا 20

با استفاده از حلقه

```
for (int i = 1; i <= 20; i++) {
    System.out.println("i = "+i);
}
```

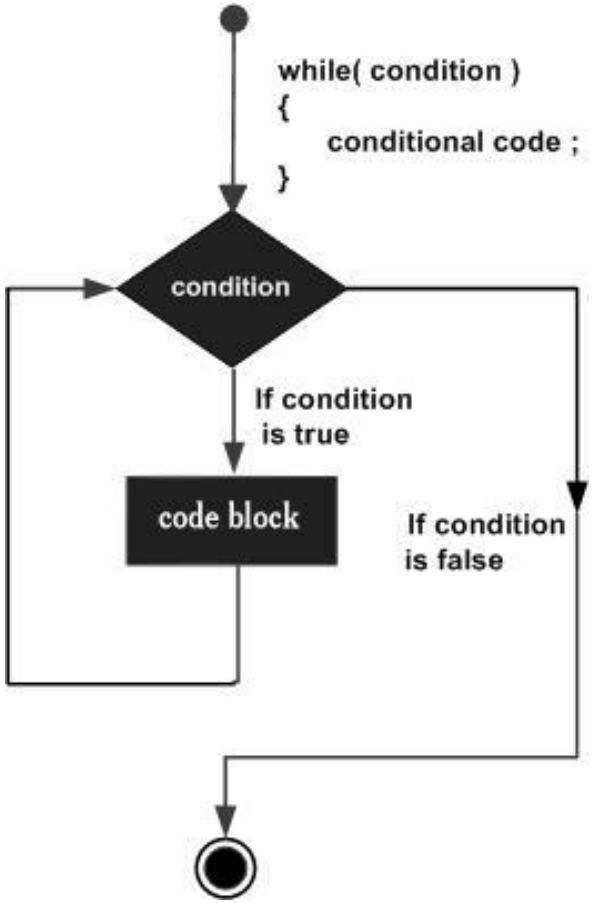
بدون استفاده از حلقه

```
int i=1;
System.out.println("i = 1");
i=2;
System.out.println("i = 2");
i=3;
System.out.println("i = 3");
.
.
.
```

# انواع حلقه ها در جاوا

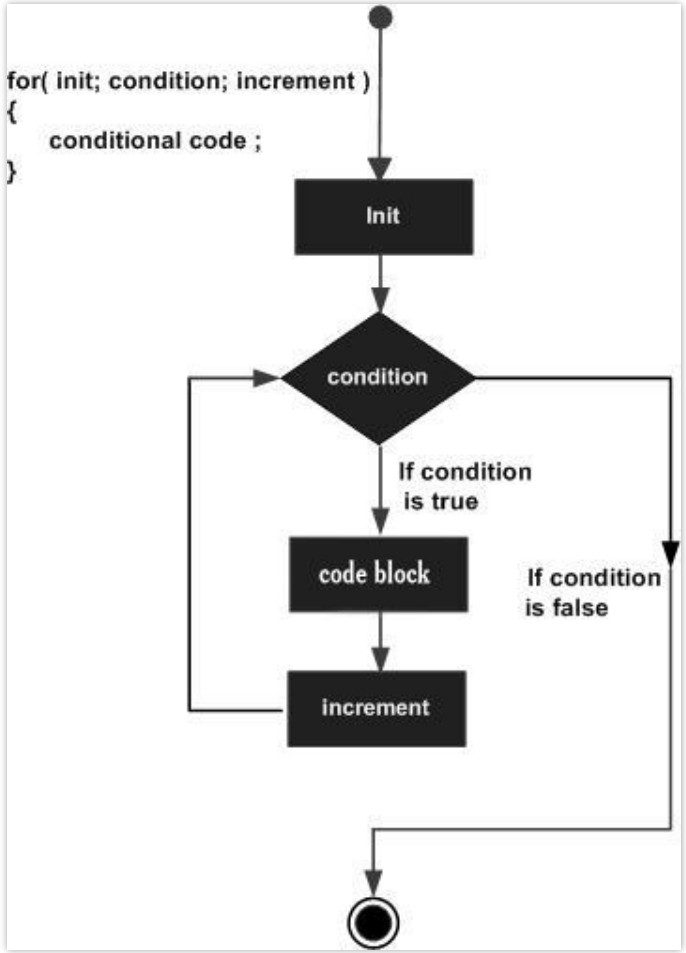
- حلقه ی while
- حلقه ی for
- حلقه ی do... while

# حلقه ی while



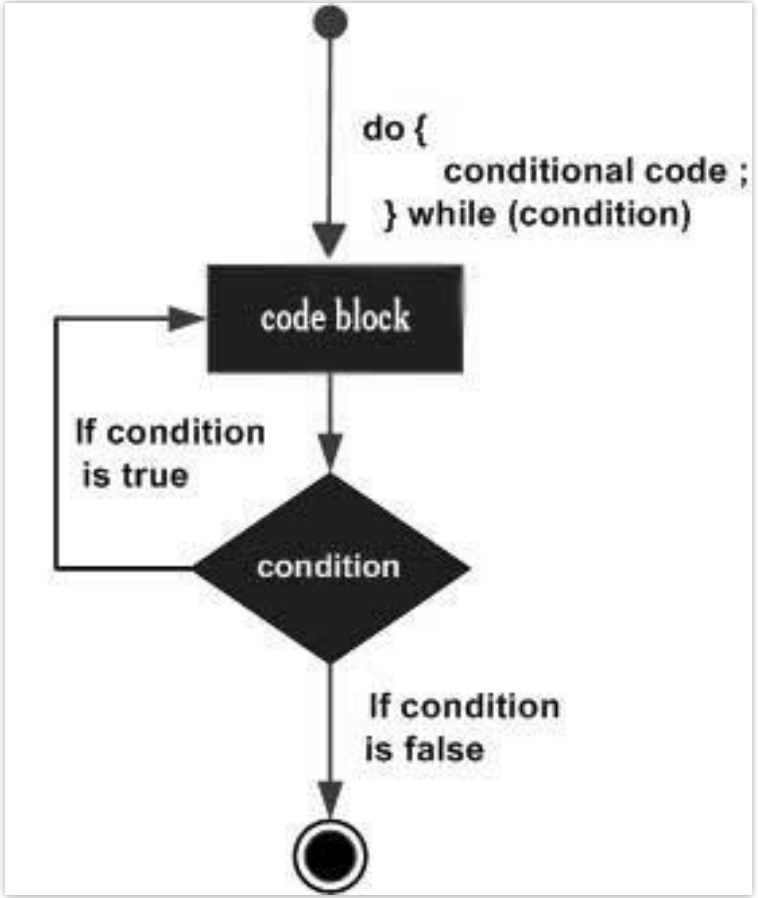

```
int i=1;
while (i<=20){
    System.out.println("i = "+ i);
    i++;
}
```

# حلقه ی for




```
for (int i = 1; i <= 20; i++) {
    System.out.println("i = "+i);
}
```

# حلقه ی do...while

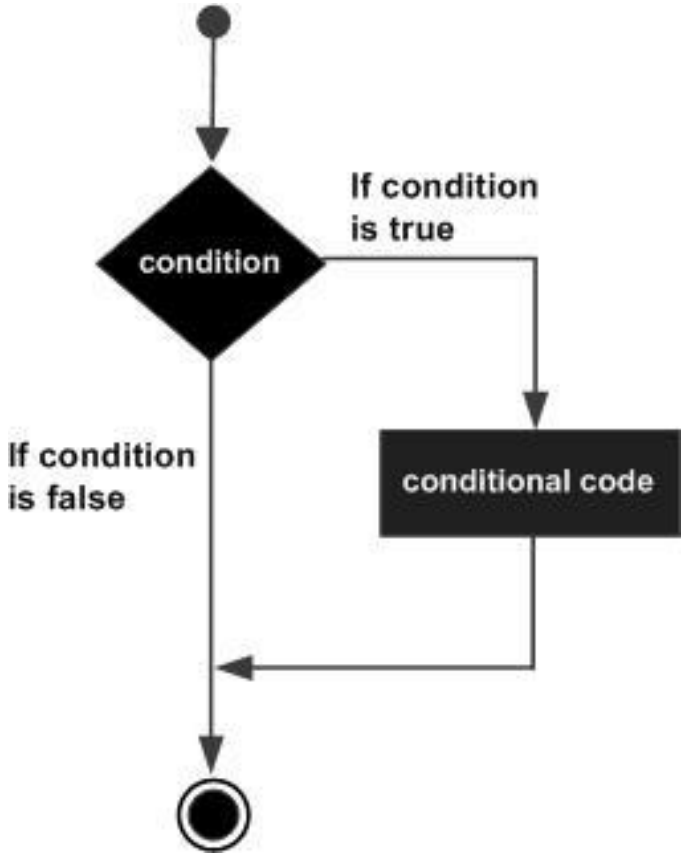



```
int i=1;
do{
    System.out.println("i = "+i);
    i++;
}while (i<=20);
```

# روش های تصمیم گیری در جاوا

- در جاوا برای اینکه بخواهیم بر اساس شرایط خاصی، یک تکه کد اجرا شود، باید از روش های تصمیم گیری استفاده کنیم.
- روش های تصمیم گیری:
- استفاده از عملگر ?
- استفاده از دستور if
- استفاده از ساختار if...else
- استفاده از ساختارهای تو در توی if ( nested if )
- استفاده از ساختار switch

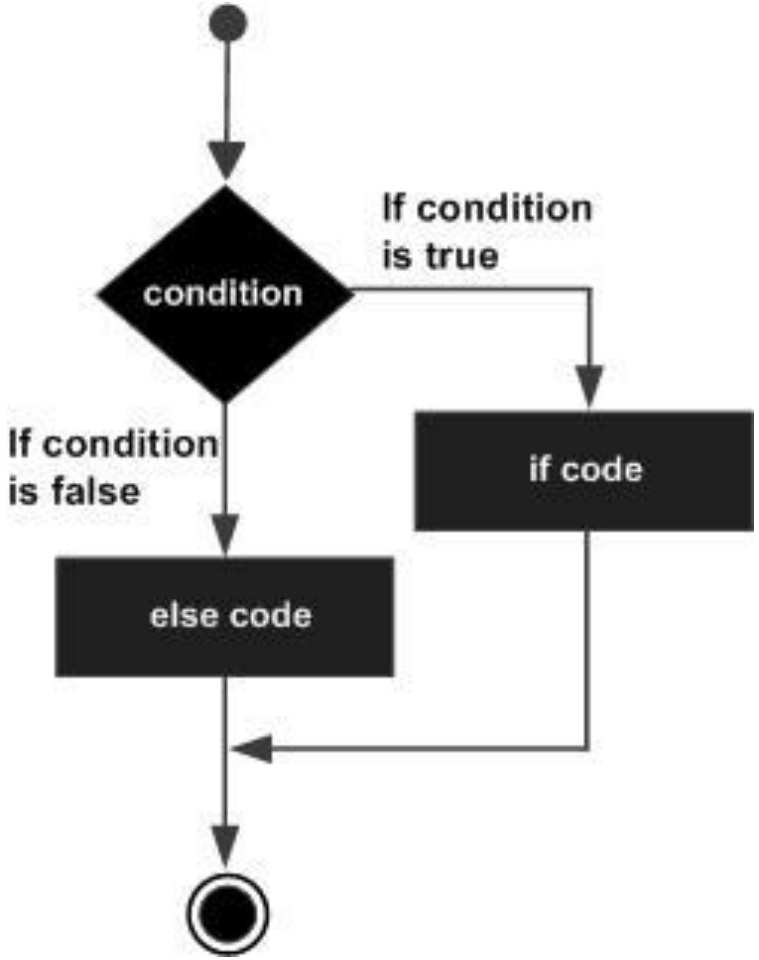
# دستور if




مثال:

```
Student student=new Student();
if (student.getAge()==18){
    System.out.println("دانش آموز 18 سال دارد.");
}
```

# ساختار if else




مثال:

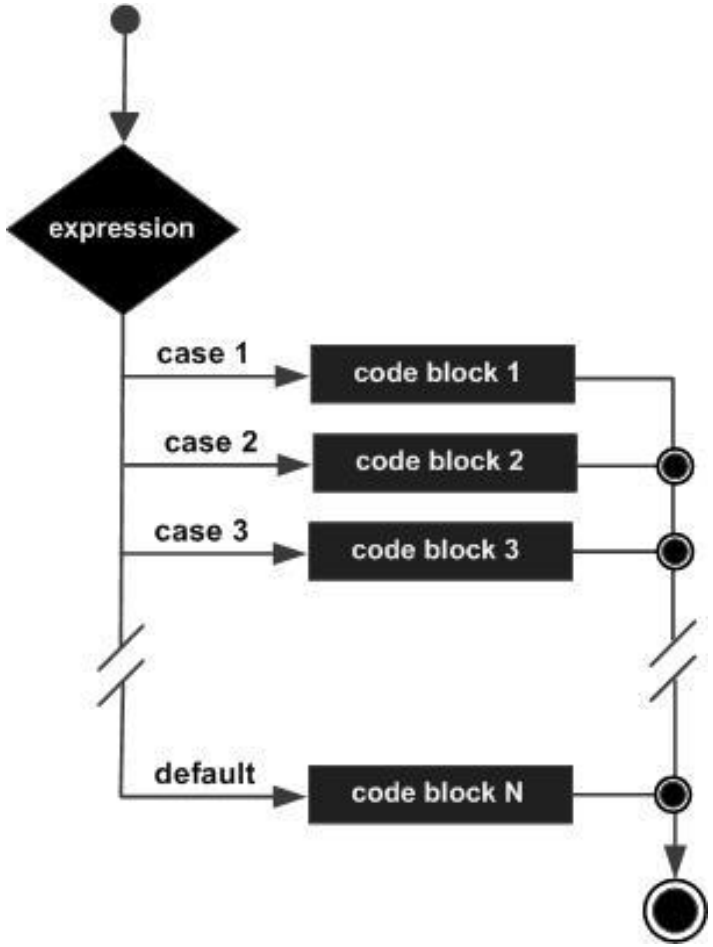
```
int a=10;
if (a==10){
    System.out.println("شرط درست است");
}else {
    System.out.println("شرط درست نیست");
}
```

# استفاده از ساختارهای تو در توی if ( nested if )

مثال:

```
Student student=new Student();
if (student.getAge()==18){
    System.out.println("دانش آموز 18 سال دارد.");
    if (student.getScore()==20){
        System.out.println("او نمره ی 20 را کسب کرده است.");
    }
}
```

# استفاده از ساختار switch




مثال:

```
Student student=new Student();
switch (student.getGrade()){
    case 'A':
        System.out.println("خیلی عالی");
        break;
    case 'B':
        System.out.println("خوب");
        break;
    case 'C':
        System.out.println("باید تلاشتو بیشتر کنی");
        break;
    default:
        System.out.println("نمره نا معتبر است!");
        break;
}
```