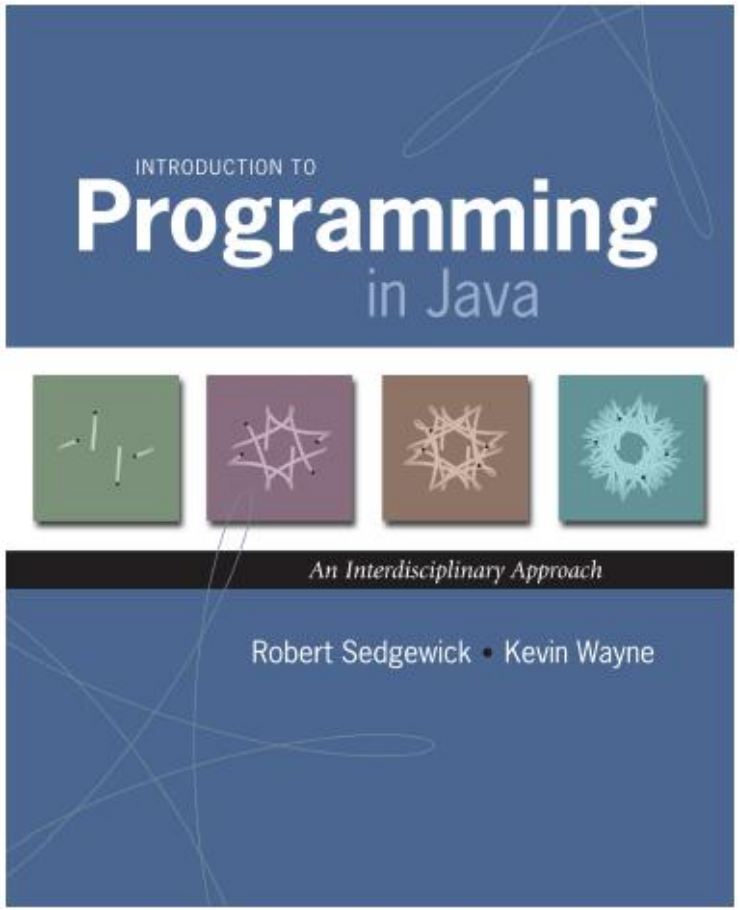
عناصر برنامه‌نویسی:
جریان کنترل

سید ناصر رضوی **www.snrazavi.ir**
۱۳۹۵

۱-۳ دستورات شرطی و حلقه‌ها



اجزای برنامه‌نویسی

any program you might want to write

objects

functions and modules

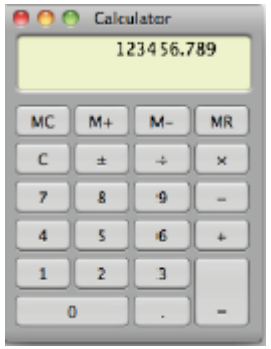
graphics, sound, and image I/O

arrays

conditionals and loops

Math | Text I/O

primitive data types | assignment statements



تا اینجا:

کامپیوتر به عنوان یک ماشین حساب

اجزای برنامه‌نویسی

any program you might want to write

objects

functions and modules

graphics, sound, and image I/O

arrays

conditionals and loops

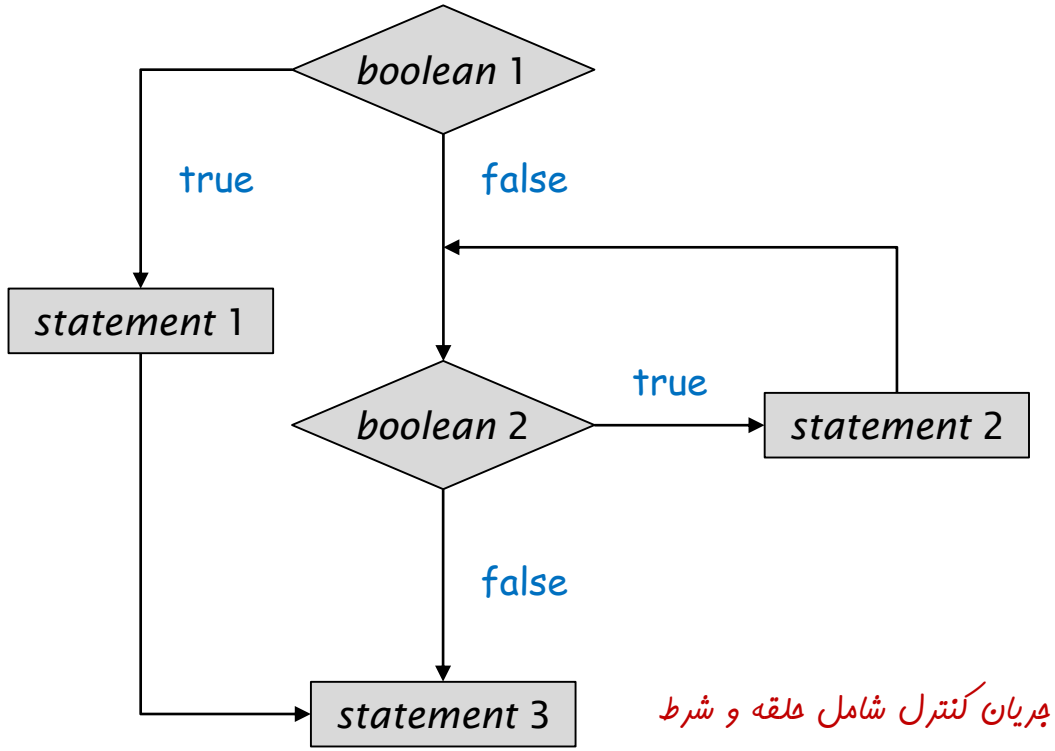
Math | Text I/O

primitive data types | assignment statements

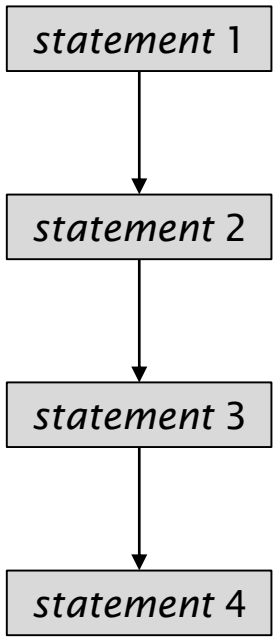
به سوی بینهایت!

جریان کنترل

- جریان کنترل.
  - دنباله‌ای از دستورات در برنامه که واقعاً اجرا می‌شوند.
  - دستورات شرطی و حلقه‌ها: تغییر جریان کنترل.



جریان کنترل شامل حلقه و شرط

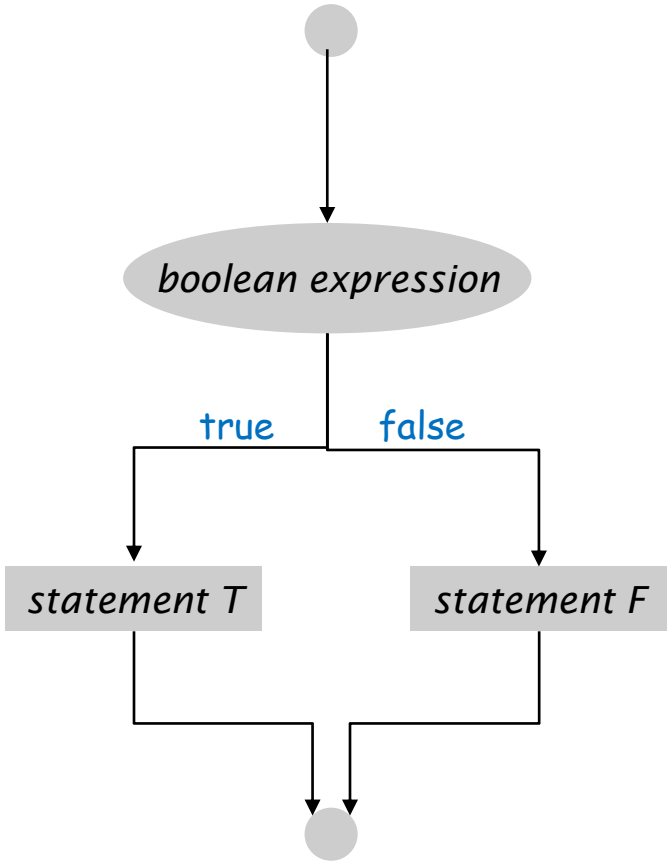


جریان کنترل خط مستقیم

دستورات شرطی

دستور if

- دستور `if`. یک ساختار انشعاب متداول.
  - یک عبارت بولی ارزیابی می‌شود.
  - اگر حاصل `true` بود، آنگاه یک مجموعه از دستورات اجرا می‌شوند.
  - اگر حاصل `false` بود، آنگاه یک مجموعه‌ی دیگر از دستورات اجرا می‌شوند.

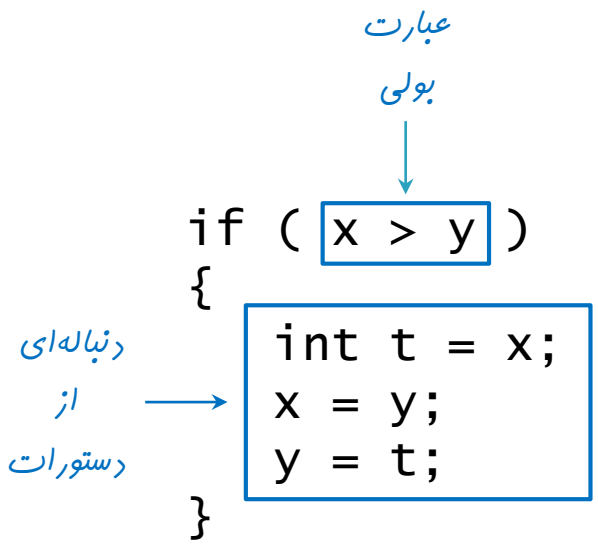


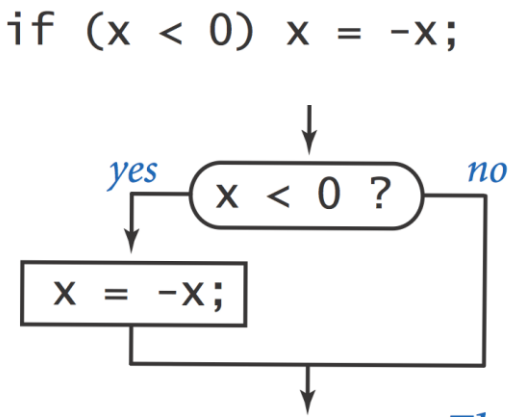

```
if (boolean expression) {
    statement T;
}
else {
    statement F;
}
```

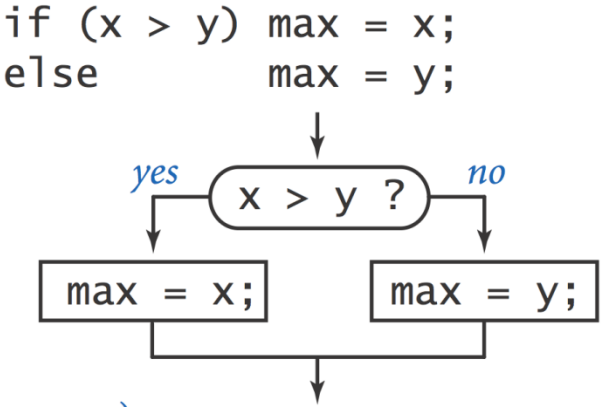
در اینجا هر مجموعه از دستورات می‌تواند قرار بگیرد.

دستور if

- دستور `if`. یک ساختار انشعاب متداول.
  - یک عبارت بولی ارزیابی می‌شود.
  - اگر حاصل `true` بود، آنگاه یک مجموعه از دستورات اجرا می‌شوند.
  - اگر حاصل `false` بود، آنگاه یک مجموعه‌ی دیگر از دستورات اجرا می‌شوند.







دستور if

- مثال. انجام اعمال مختلف بر مبنای مقدار یک متغیر.

```
public class Flip {
   public static void main(String[] args) {
      if   (Math.random() < 0.5) System.out.println("Heads");
      else Math.random() > 0.5) System.out.println("Tails");
   }
}
```

```
% java Flip
Heads

% java Flip
Heads

% java Flip
Tails

% java Flip
Heads
```

چند مثال از کاربرد دستور if

absolute value	`if (x < 0) x = -x;`

put x and y into sorted order	`if (x > y)`
	`{`
	`  int t = x;`
	`  x = y;`
	`  y = t;`
	`}`

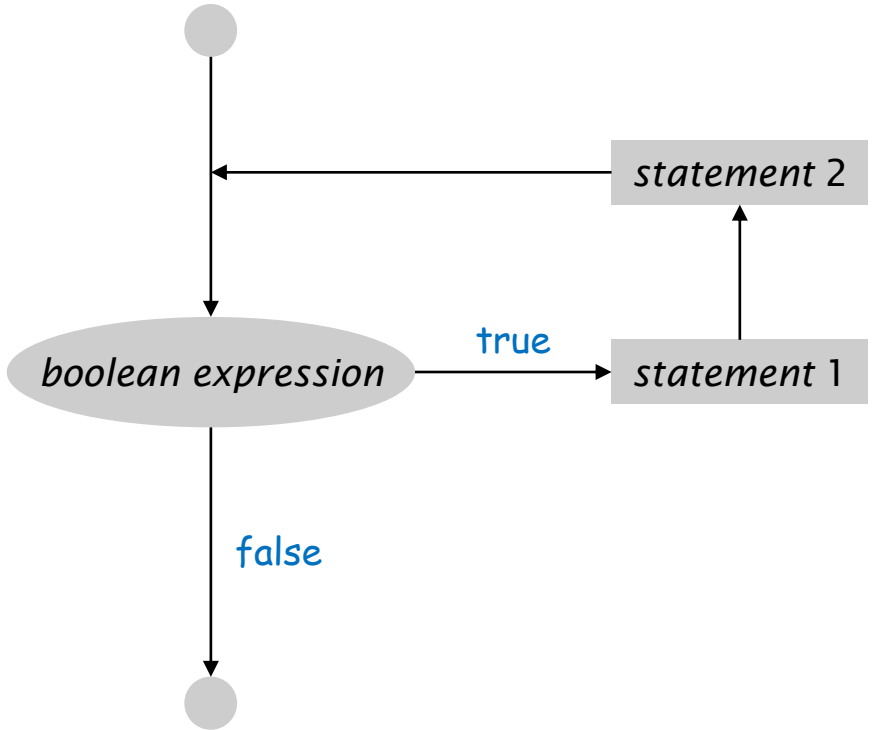
maximum of x and y	`if (x > y) max = x;` `else  max = y;`
error check for division operation	`if (den == 0) System.out.println("Division by zero");` `else  System.out.println("Quotient = " + num / den);`

error check for quadratic formula	`double discriminant = b*b - 4.0*c;`
	`if (discriminant < 0) {`
	`  System.out.println("No real roots");`
	`}`
	`else {`
	`  System.out.println((-b + Math.sqrt(discriminant))/2.0);`
	`  System.out.println((-b - Math.sqrt(discriminant))/2.0);`
	`}`

حلقه‌ی while

حلقه‌ی while

- حلقه‌ی `while` یک ساختار تکرار متداول.
  - یک عبارت بولی ارزیابی می‌شود.
  - اگر حاصل `true` بود، تعدادی دستور اجرا می‌شود.
  - تکرار.




```
while (boolean expression) {
    statement 1 ;
    statement 2 ;
}
```
← بدنه حلقه

توان‌های عدد ۲

- مثال. توان‌های عدد ۲ را که کوچک‌تر یا مساوی 2^N هستند، چاپ کنید.
  - مقدار i را از 0 به N افزایش دهید. [در هر تکرار یک واحد]
  - در هر تکرار مقدار v را دو برابر کنید.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

```
0 1
1 2
2 4
3 8
4 16
5 32
6 64
```

N = 6

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true
7	128	false

توان‌های عدد ۲


```java
public class PowersOfTwo {

   public static void main(String[] args) {

      // last power of two to print
      int N = Integer.parseInt(args[0]);

      int i = 0; // loop control counter
      int v = 1; // current power of two
      while (i <= N) {
         System.out.println(i + " " + v);
         i = i + 1;
         v = 2 * v;
      }
   }
}
```

```
% java PowersOfTwo 3
0 1
1 2
2 4
3 8

% java PowersOfTwo 6
0 1
1 2
2 4
3 8
4 16
5 32
6 64
```

چالش‌های حلقه‌ی while

- پرسش. آیا در تکه کد زیر برای چاپ توان‌های عدد ۲ اشتباهی وجود دارد؟

```
int i = 0;
int v = 1;
while (i <= N)
   System.out.println(i + " " + v);
   i = i + 1;
   v = 2 * v;
```

محاسبه‌ی جذر اعداد

- هدف. پیاده‌سازی تابع `Math.sqrt()`
- روش نیوتن-رافسون به منظور محاسبه‌ی ریشه‌ی دوم عدد c
  - مقدار t_0 را برابر با c قرار بده.
  - تا زمانی که شرط $t_i = c/t_i$ برقرار نشده، عمل زیر را تکرار کن:
    مقدار t_{i+1} را برابر با میانگین مقادیر t_i و c/t_i قرار بده.

```
% java Sqrt 2.0
1.414213562373095
```

$$
\begin{aligned}
t_0 &= 2.0 \\
t_1 &= \frac{1}{2}\left(t_0+\frac{2.0}{t_0}\right) = 1.5 \\
t_2 &= \frac{1}{2}\left(t_1+\frac{2.0}{t_1}\right) = 1.4166666666666665 \\
t_3 &= \frac{1}{2}\left(t_2+\frac{2.0}{t_2}\right) = 1.4142156862745097 \\
t_4 &= \frac{1}{2}\left(t_3+\frac{2.0}{t_3}\right) = 1.4142135623746899 \\
t_5 &= \frac{1}{2}\left(t_4+\frac{2.0}{t_4}\right) = 1.414213562373095
\end{aligned}
$$

محاسبه‌ی جذر عدد ۲

محاسبه‌ی جذر اعداد

- هدف. پیاده‌سازی تابع `Math.sqrt()`
- روش نیوتن-رافسون به منظور محاسبه‌ی ریشه‌ی دوم عدد c
  - مقدار t_0 را برابر با c قرار بده.
  - تا زمانی که شرط $t_i = c/t_i$ برقرار نشده، عمل زیر را تکرار کن:

    مقدار t_{i+1} را برابر با میانگین مقادیر t_i و c/t_i قرار بده.

```
% java Sqrt 2.0
1.414213562373095
```

```
public class Sqrt {

   public static void main(String[] args) {
      double epsilon = 1e-15;
      double c = Double.parseDouble(args[0]);
      double t = c;
      while (Math.abs(t - c/t) > t*epsilon) {
         t = (c/t + t) / 2.0;
      }
      System.out.println(t);
   }
}
```

محاسبه‌ی جذر اعداد: ردیابی اجرا


```
public class Sqrt {
   public static void main(String[] args) {
      double epsilon = 1e-15;
      double c = Double.parseDouble(args[0]);
      double t = c;
      while (Math.abs(t - c/t) > t*epsilon) {
         t = (c/t + t) / 2.0;
      }
      System.out.println(t);
   }
}
```

iteration	t	c/t
	2.0000000000000000	1.0
1	1.5000000000000000	1.3333333333333333
2	1.4166666666666665	1.4117647058823530
3	1.4142156862745097	1.4142114384748700
4	1.4142135623746899	1.4142135623715002
5	1.4142135623730950	1.4142135623730951

روش نیوتن-رافسون

روش محاسبه‌ی ریشه‌ی دوم.

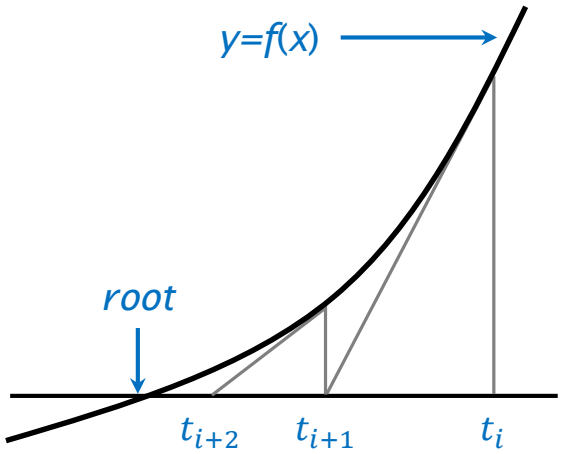
- هدف: یافتن ریشه‌ی هر تابع مانند $f(x)$
- با تخمین t_0 شروع کن.
- خط مماس بر منحنی را در $x = t_i$ ترسیم کن.
- مقدار t_{i+1} را برابر با مختصات x نقطه‌ی برخورد خط مماس و محور x قرار بده.
- مراحل فوق را تا رسیدن به دقت دلخواه تکرار کن.

برای محاسبه‌ی ریشه‌ی دوم c باید داشته باشیم $f(x) = x^2 - c$

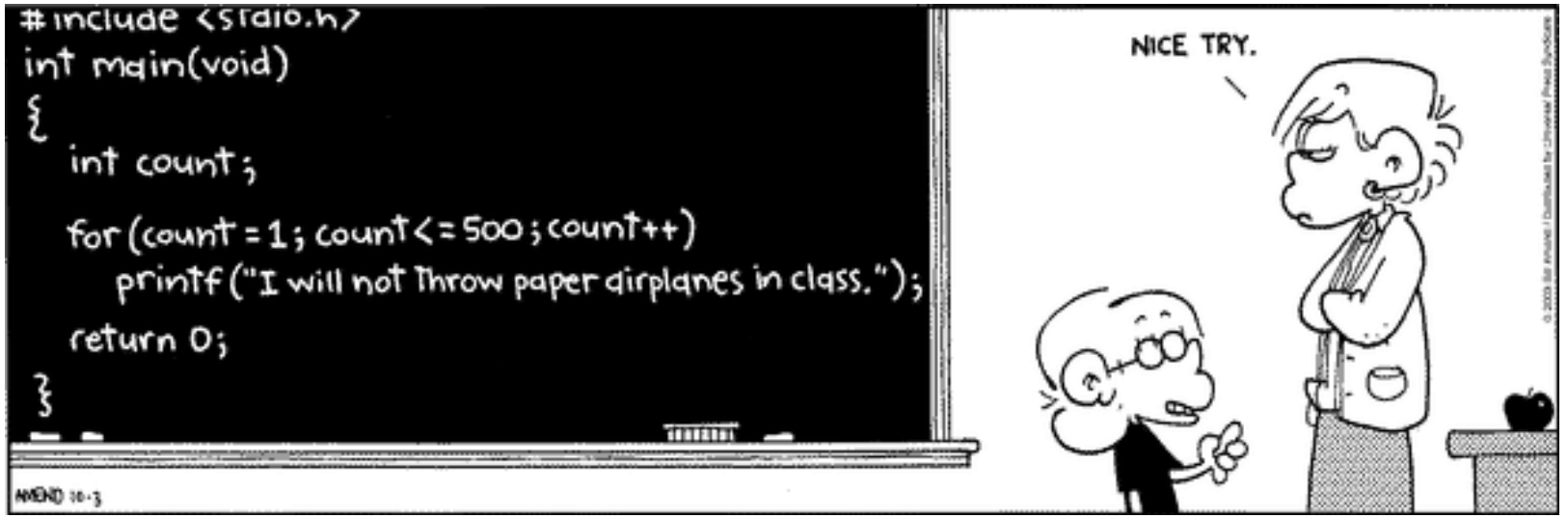
شرایط تکنیکی.

- تابع $f(x)$ باید هموار باشد.
- تخمین اولیه باید مناسب باشد.

$$t_{i+1} = t_i - \frac{f(t_i)}{f'(t_i)}$$

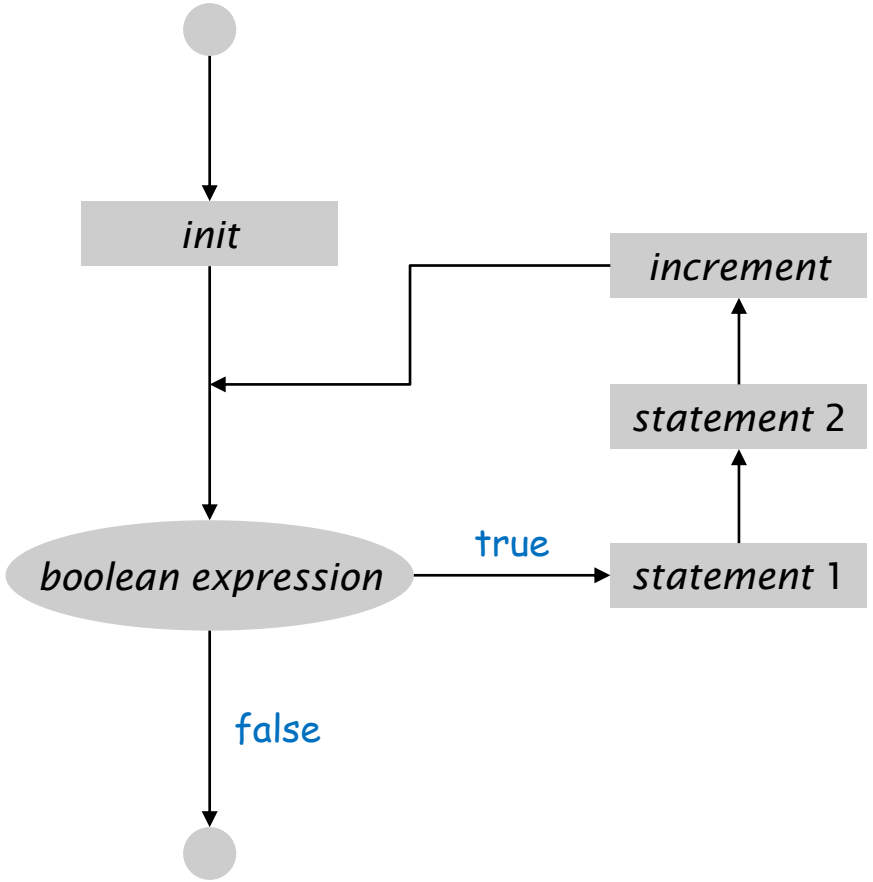


حلقه‌ی for


#include <stdio.h>
int main(void)
{
int count;
for (count = 1; count<= 500; count++)
printf ("I will not Throw paper airplanes in class.");
return 0;
}
NICE TRY.

حلقه‌ی for

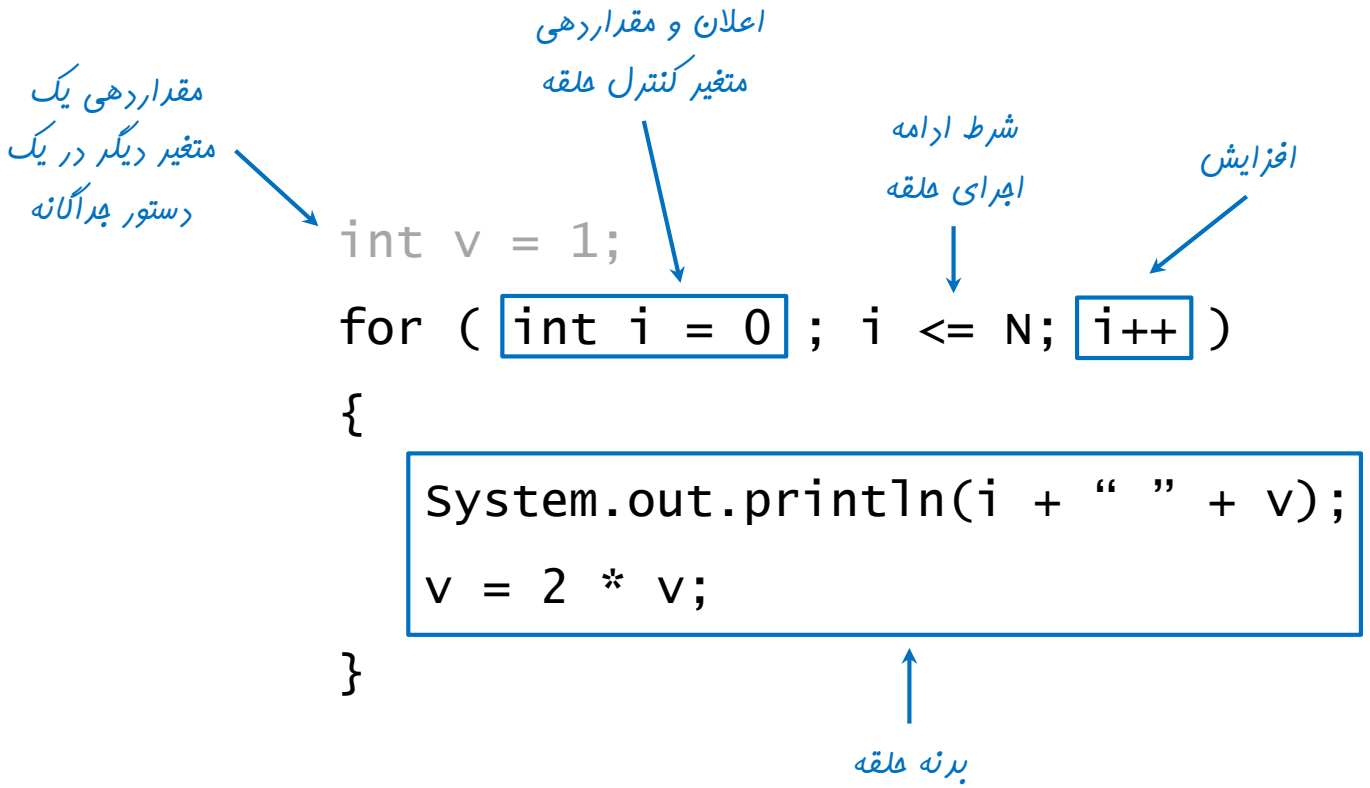
- حلقه‌ی for. یک ساختار تکرار متداول دیگر.
  - دستور مربوط به مقداردهی اولیه را اجرا می‌کند.
  - یک عبارت بولی ارزیابی می‌شود.
  - اگر true، تعدادی دستور اجرا می‌شود.
  - سپس دستور افزایش اجرا می‌شود.
  - تکرار.




```
for (init; boolean expression; increment) {
    statement 1;
    statement 2;
}
```

← برنه حلقه

ساختار یک حلقه‌ی for



- پرسش. این حلقه چه چیزی را چاپ می‌کند؟

ترسیم خط کش

- ایجاد بخش‌بندی‌های مربوط به یک خط کش.
  - مقدار رشته‌ی `ruler` را برابر با " " قرار بده.
  - به ازای هر مقدار `i` از `1` تا `N`:

دو نسخه از مقدار رشته‌ی `ruler` را در دو طرف مقدار `i` قرار بده.

```
public class RulerN {

   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      String ruler = " ";
      for (int i = 1; i <= N; i++) {
         ruler = ruler + i + ruler;
      }
      System.out.println(ruler);
   }
}
```

i	ruler
	" "
1	" 1 "
2	" 1 2 1 "
3	" 1 2 1 3 1 2 1 "

ترسیم خط کش

- مشاهده. حلقه‌ها می‌توانند حجم عظیمی از خروجی را تولید کنند.

```
% java RulerN 1
 1

% java RulerN 2
 1 2 1

% java RulerN 3
 1 2 1 3 1 2 1

% java RulerN 4
 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

% java RulerN 5
 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

% java RulerN 100
 Exception in thread "main"
 java.lang.OutOfMemoryError
```

چند مثال از حلقه‌ها

print largest power of two less than or equal to N	`int v = 1;`
	`while (v <= N/2)`
	`  v = 2 * v;`
	`System.out.println(v);`

compute a finite sum (1 + 2 + … + N)	`int sum = 0;`
	`for (int i = 1; i <= N; i++)`
	`  sum += i;`
	`System.out.println(sum);`

compute a finite product (N! = 1 * 2 * … * N)	`int product = 1;`
	`for (int i = 1; i <= N; i++)`
	`  product *= i;`
	`System.out.println(product);`

print a table of function values	`for (int i = 0; i <= N; i++)` `  System.out.println(i + " " + 2 * Math.PI*i/N);`

ساختارهای تو در تو

دستورات if تو در تو

- مثال. پرداخت یک نرخ مالیات خاص بر مبنای سطح درآمد.

Income	Rate
0 - 47,450	22%
47,450 - 114,650	25%
114,650 - 174,700	28%
174,700 - 311,950	33%
311,950 -	35%

```
double rate;
if      (income <  47450) rate = 0.22;
else if (income < 114650) rate = 0.25;
else if (income < 174700) rate = 0.28;
else if (income < 311950) rate = 0.33;
else if (income > 311950) rate = 0.35;
```

دستورات if تو در تو

- کاربرد. از دستورات if تو در تو برای انتخاب از میان چندین گزینه‌ی مختلف استفاده کنید.

```
if (income < 47450) rate = 0.22;
else {
   if (income < 114650) rate = 0.25;
   else {
      if (income < 174700) rate = 0.28;
      else {
         if (income < 311950) rate = 0.33;
         else rate = 0.35;
      }
   }
}
```

دستورات if تو در تو

- آیا همه‌ی آکولادها لازم هستند؟ نه همیشه.

شکل خلاصه شده

```
if      (income <  47450) rate = 0.22;
else if (income < 114650) rate = 0.25;
else if (income < 174700) rate = 0.28;
else if (income < 311950) rate = 0.33;
else                      rate = 0.35;
```

```
if (income < 47450) rate = 0.22;
else {
   if (income < 114650) rate = 0.25;
   else {
      if (income < 174700) rate = 0.28;
      else {
         if (income < 311950) rate = 0.33;
         else rate = 0.35;
      }
   }
}
```

دستورات if تو در تو: چالش

- پرسش. در محاسبات زیر برای محاسبه‌ی نرخ مالیات چه چیزی نادرست است؟

Income	Rate
0 - 47,450	22%
47,450 - 114,650	25%
114,650 - 174,700	28%
174,700 - 311,950	33%
311,950 -	35%

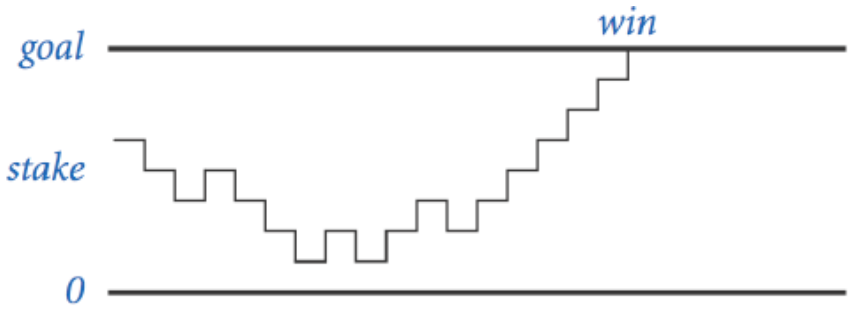
```
double rate = 0.35;
if (income <  47450) rate = 0.22;
if (income < 114650) rate = 0.25;
if (income < 174700) rate = 0.28;
if (income < 311950) rate = 0.33;
```

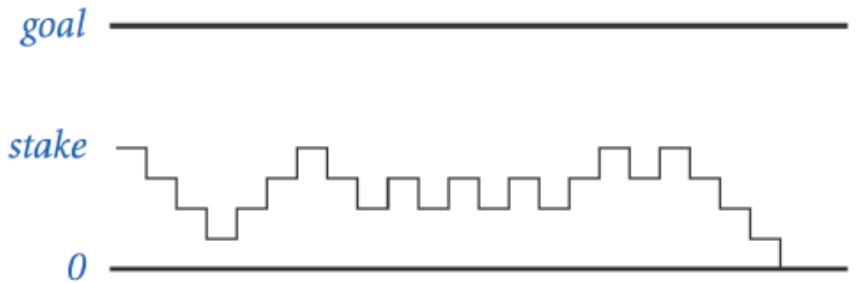
شبیه‌سازی مونت-کارلو

ورشکستگی قمارباز

- مسئله. یک قمارباز با مبلغ **$stake** شروع می‌کند و هر بار بر روی **$1** شرط می‌بندد تا این که کل پول خود را ببازد یا موجودی وی به مقدار **$goal** برسد.
  - شانس برد چقدر است؟
  - برای بردن به چند شرط‌بندی نیاز است؟

- یک رویکرد. شبیه‌سازی مونت کارلو
  - شیر یا خط کن و نتیجه را بررسی کن.
  - عمل فوق را تکرار و آمار مورد نیاز را محاسبه کن.





ورشکستگی قمارباز


```
public class Gambler {
   public static void main(String[] args) {
      int stake = Integer.parseInt(args[0]);
      int goal  = Integer.parseInt(args[1]);
      int T     = Integer.parseInt(args[2]);
      int wins = 0;
      // repeat experiment T times
      for (int t = 0; t < T; t++) {
         // do one gambler's ruin experiment
         int cash = stake;
         while (cash > 0 && cash < goal) {
            // flip coin and update
            if (Math.random() < 0.5) cash++;
            else                     cash--;
         }
         if (cash == goal) wins++;
      }
      System.out.println(wins + " wins of " + T);
   }
}
```

شبیه‌سازی و تحلیل نتایج

stake goal T

```
% java Gambler 5 25 1000
191 wins of 1000
% java Gambler 5 25 1000
203 wins of 1000
% java Gambler 500 2500 1000
197 wins of 1000
```

- حقیقت. احتمال برنده شدن = $stake \div goal$

تفاوت میان موجودی اولیه و هدف

- حقیقت. تعداد مورد انتظار شرط‌بندی‌ها = $stake \times desired\ gain$
- مثال. شانس تبدیل ۵۰۰ دلار به ۲۵۰۰ دلار برابر است با ۲۰ درصد. همچنین تعداد متوسط شرط‌بندی‌ها برابر است با ۱ میلیون!
- ملاحظه. هر دو حقیقت بالا به صورت ریاضی قابل اثبات هستند؛ اما برای سناریوهای پیچیده‌تر، شبیه‌سازی کامپیوتری اغلب بهترین (تنها) روش است.

جریان کنترل: خلاصه

- جریان کنترل.
  - دنباله‌ای از دستورات در برنامه که واقعاً اجرا می‌شوند.
  - دستورات شرطی و حلقه‌ها: تغییر جریان کنترل.

جریان کنترل	توضیحات	مثال‌ها
خط مستقیم	تمام دستورات برنامه به ترتیب داده شده اجرا می‌شوند	
دستورات شرطی	برخی از دستورات برنامه بسته به مقدار بعضی از متغیرهای خاص اجرا می‌شوند	`if` `if-else`

حلقه‌های تکرار	تا زمانی که شرایط خاصی برقرار باشد، برخی از دستورات به طور مکرر اجرا می‌شوند	`while`
		`for`
		`do-while`