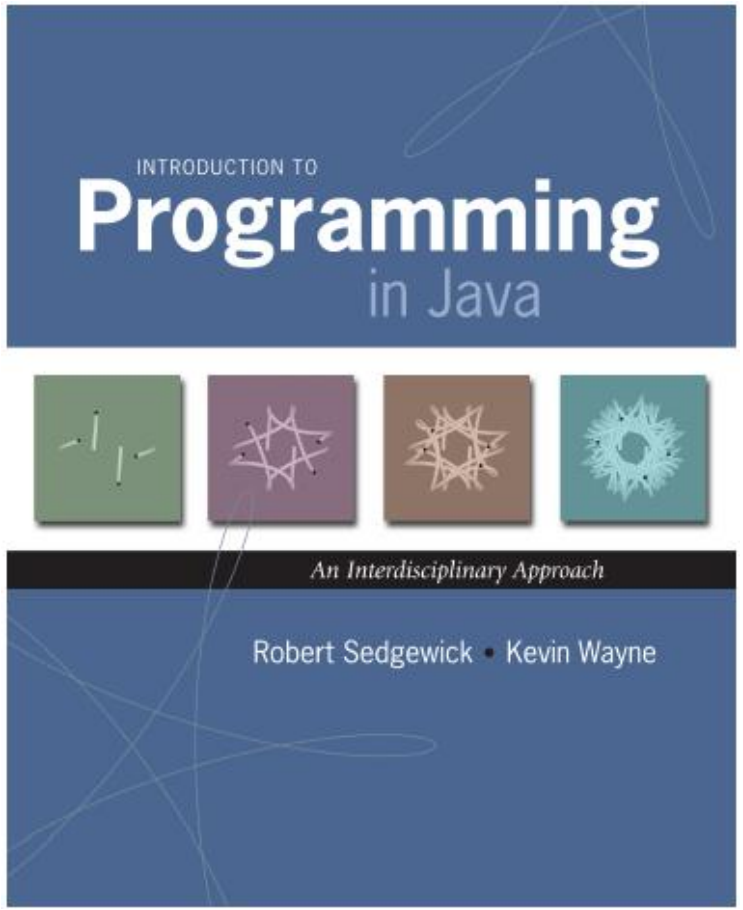
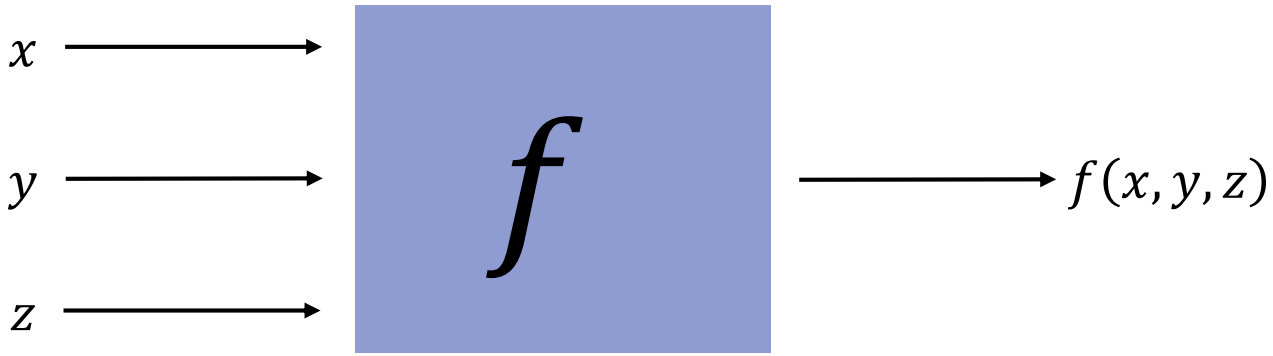
توابع و کتابخانه‌ها: توابع

سید ناصر رضوی **www.snrazavi.ir**
۱۳۹۶

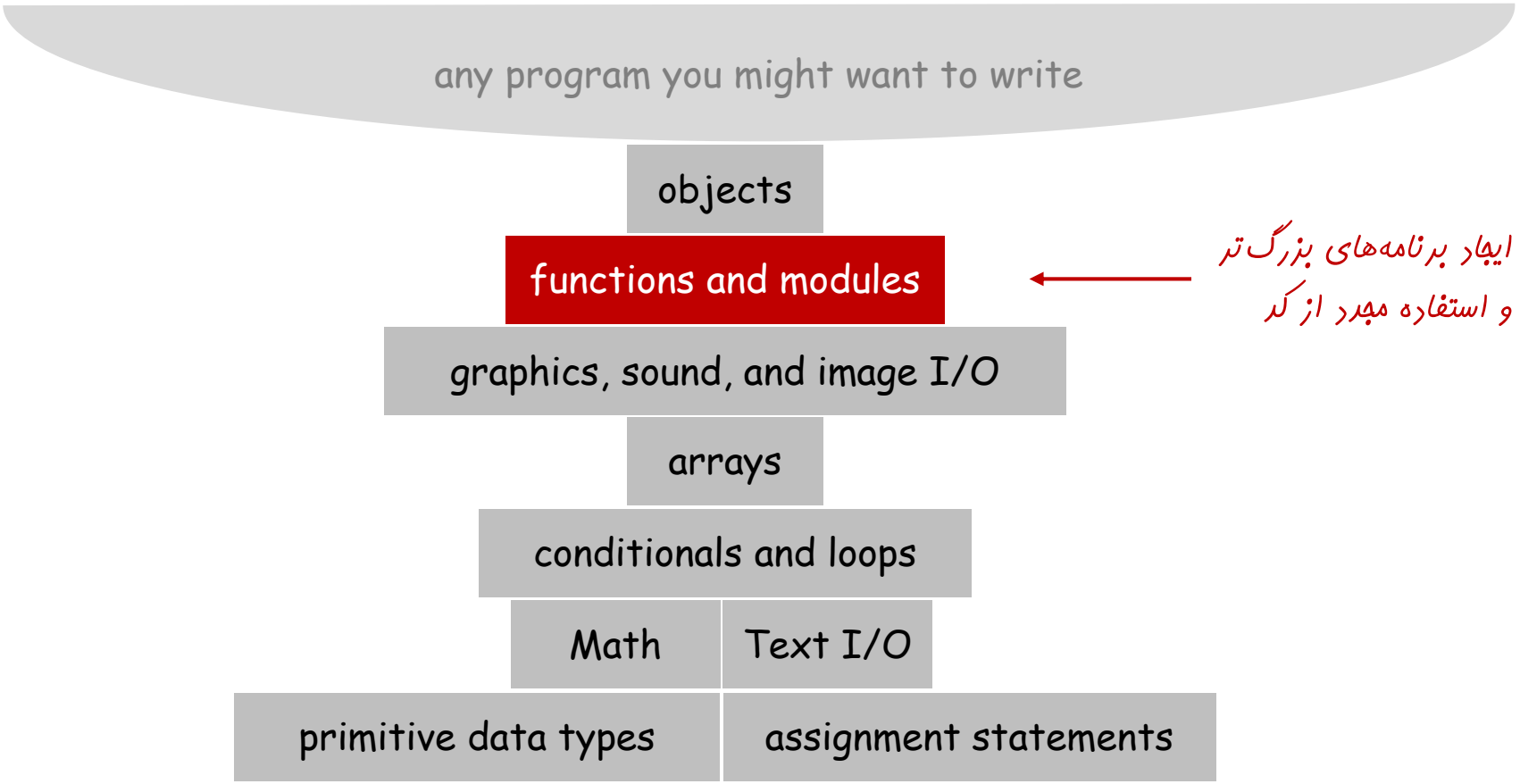
۲-۱ توابع


INTRODUCTION TO
Programming
in Java
An Interdisciplinary Approach
Robert Sedgewick • Kevin Wayne

۲-۱ توابع



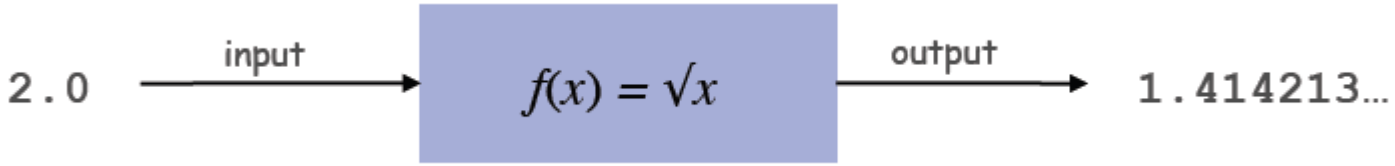
اجزای برنامه‌نویسی

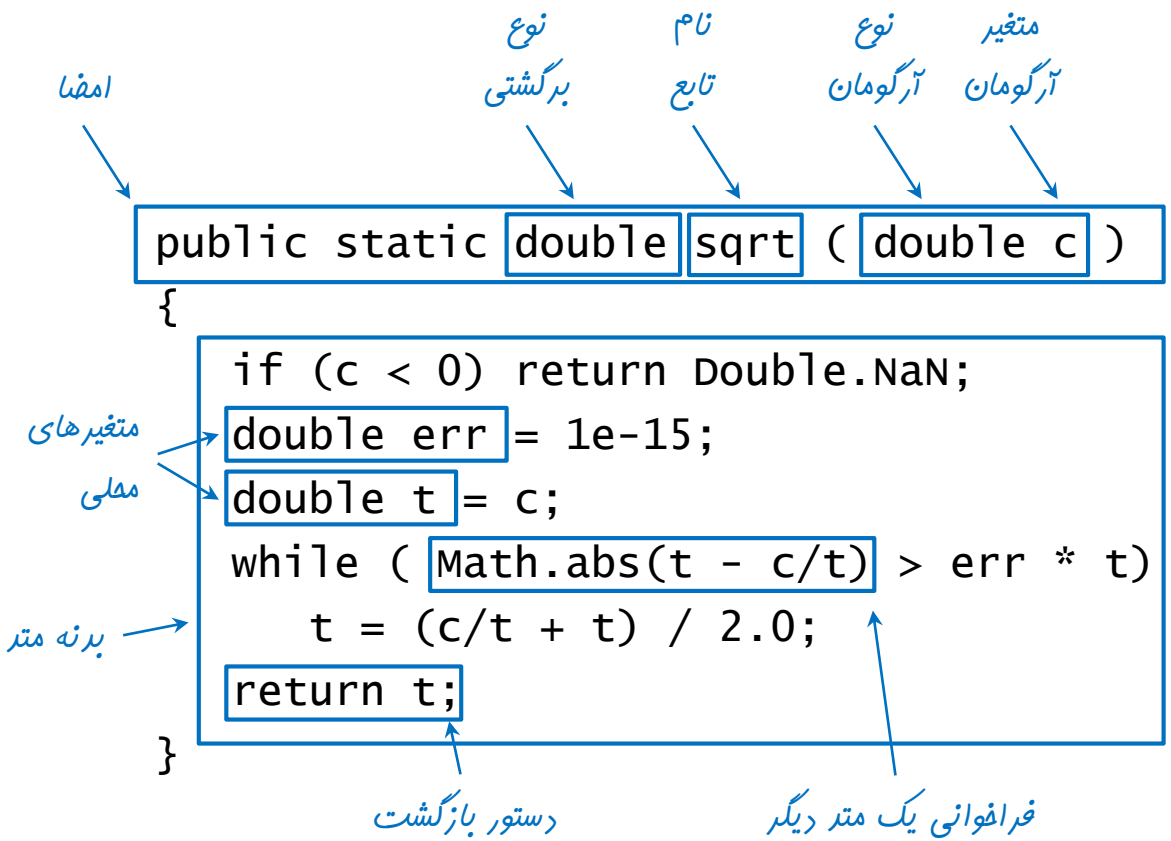


توابع (متدهای استاتیک)

- تابع جاوا.
  - دریافت صفر یا چندکمیت به عنوان ورودی.
  - برگرداندن یک کمیت به عنوان خروجی.
  - اثرات جانبی (مانند نوشتن در خروجی یا ترسیم نمودار).
- کاربردها.
  - دانشمندان از توابع ریاضی برای محاسبه فرمول‌ها استفاده می‌کنند.
  - برنامه‌نویس‌ها از توابع برای ایجاد برنامه‌های پیمانه‌ای استفاده می‌کنند.
  - شما از توابع برای هر دو منظور استفاده می‌کنید.
- مثال.
  - توابع پیش ساخته: `Math.random()`، `Math.abs()` و `Integer.parseInt()`
  - کتابخانه‌های ورودی/خروجی: `StdIn.readInt()`، `StdDraw.line()` و `StdAudio.play()`
  - توابع تعریف شده به وسیله کاربر: `main()`

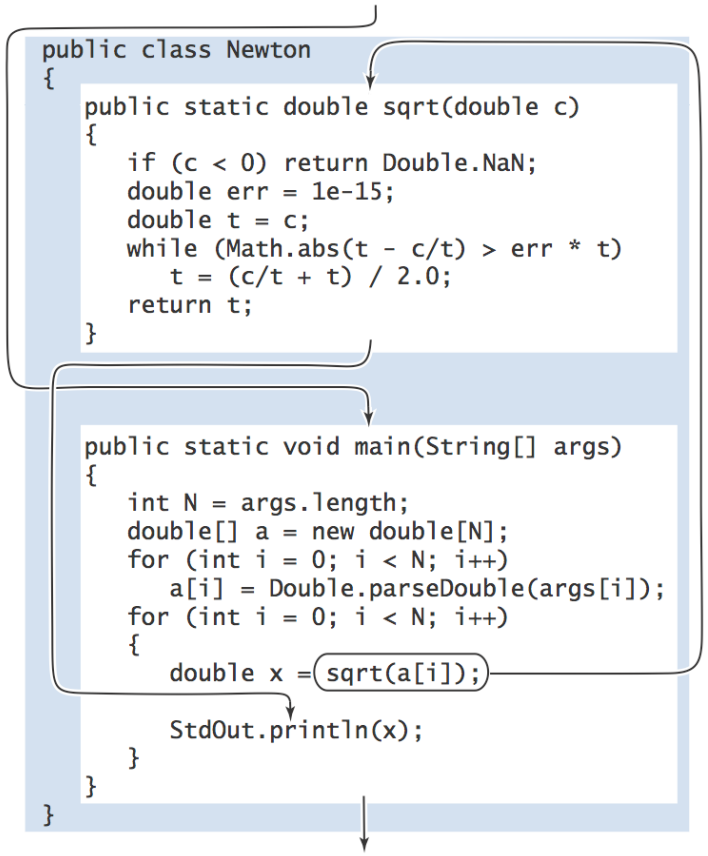
اجزای یک تابع جاوا





جریان کنترل

- نکته کلیدی. توابع یک روش جدید به منظور کنترل جریان اجرا فراهم می‌کنند.



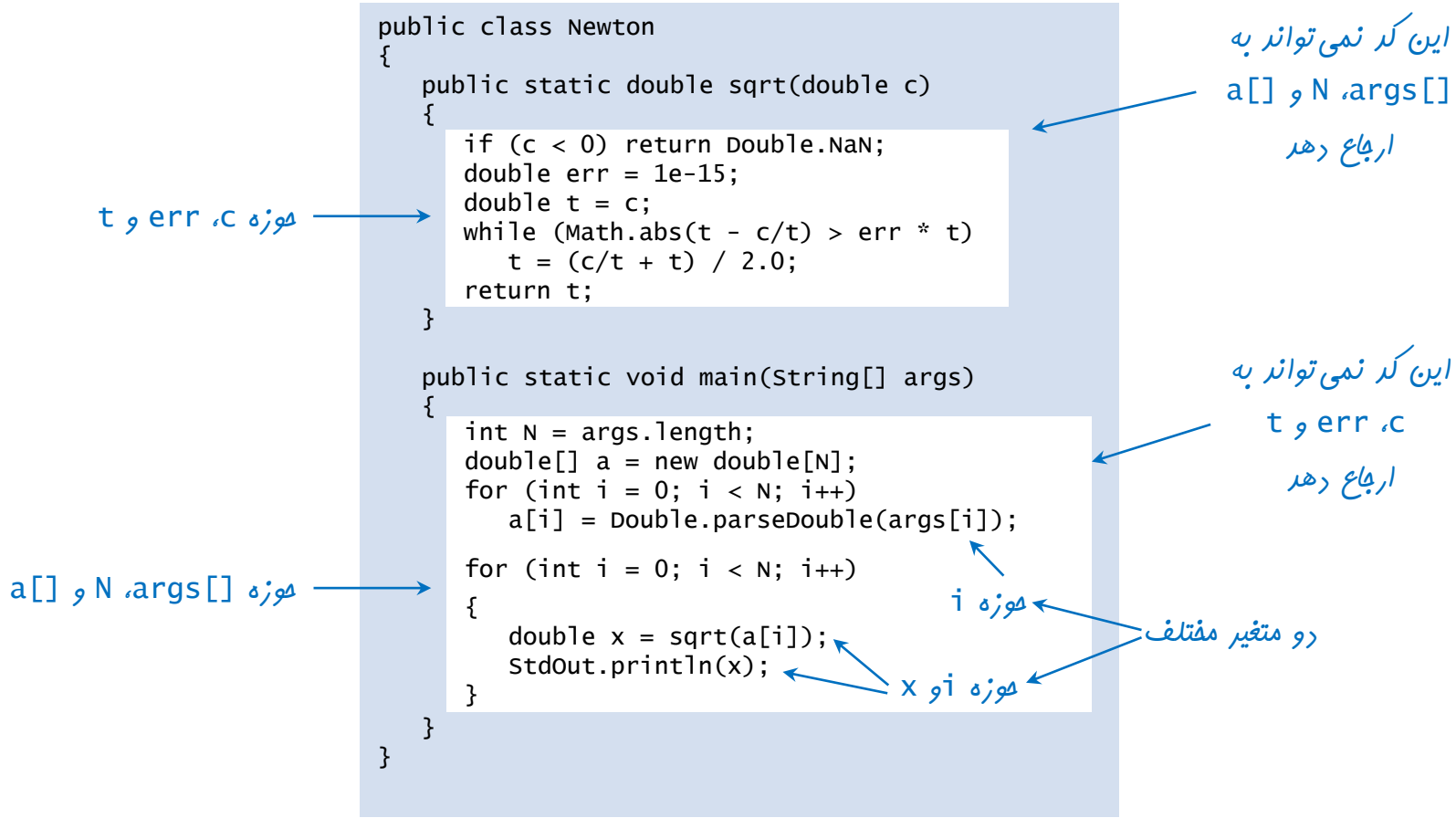
```
public class Newton
{
   public static double sqrt(double c)
   {
      if (c < 0) return Double.NaN;
      double err = 1e-15;
      double t = c;
      while (Math.abs(t - c/t) > err * t)
         t = (c/t + t) / 2.0;
      return t;
   }

   public static void main(String[] args)
   {
      int N = args.length;
      double[] a = new double[N];
      for (int i = 0; i < N; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < N; i++)
      {
         double x = sqrt(a[i]);

         StdOut.println(x);
      }
   }
}
```

- مراحل فراخوانی یک تابع.
  - کنترل به ابتدای کد تابع فراخوانی شده منتقل می‌شود.
  - آرگومان‌ها با استفاده از مقادیر موجود در فراخوانی مقداردهی می‌شوند.
  - کد تابع اجرا می‌شود.
  - به جای نام تابع در کد فراخوان مقدار برگشتی تابع فراخوانی شده قرار داده می‌شود.
  - کنترل دوباره به کد فراخوان منتقل می‌شود.

- توجه. این شیوه فراخوانی، «ارسال با مقدار» نام دارد.

حوزه

- حوزه (برای یک نام). قسمتی از کد که می‌تواند به آن نام ارجاع دهد.
- مثال. حوزه یک متغیر برابر است با کد پس از اعلان آن متغیر در درون بلوک کد.




```
public class Newton
{
   public static double sqrt(double c)
   {
      if (c < 0) return Double.NaN;
      double err = 1e-15;
      double t = c;
      while (Math.abs(t - c/t) > err * t)
         t = (c/t + t) / 2.0;
      return t;
   }

   public static void main(String[] args)
   {
      int N = args.length;
      double[] a = new double[N];
      for (int i = 0; i < N; i++)
         a[i] = Double.parseDouble(args[i]);

      for (int i = 0; i < N; i++)
      {
         double x = sqrt(a[i]);
         StdOut.println(x);
      }
   }
}
```

توابع: چالش ۱-الف

- پرسش. پس از کامپایل و اجرای کد زیر چه اتفاقی می‌افتد؟

```
public class Cubes1 {
   public static int cube(int i) {
      int j = i * i * i;
      return j;
   }

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

```
% javac Cubes1.java
% java Cubes1 6
1 1
2 8
3 27
4 64
5 125
6 216
```

توابع: چالش ۱-ب

- پرسش. پس از کامپایل و اجرای کد زیر چه اتفاقی می‌افتد؟

```
public class Cubes2 {
   public static int cube(int i) {
      int i = i * i * i;
      return i;
   }

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

توابع: چالش ۱-پ

- پرسش. پس از کامپایل و اجرای کد زیر چه اتفاقی می‌افتد؟

```
public class Cubes3 {
   public static int cube(int i) {
      i = i * i * i;
   }


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

توابع: چالش ۱-ت

- پرسش. پس از کامپایل و اجرای کد زیر چه اتفاقی می‌افتد؟

```
public class Cubes4 {
   public static int cube(int i) {
      i = i * i * i;
      return i;
   }

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

توابع: چالش ۱-ث

- پرسش. پس از کامپایل و اجرای کد زیر چه اتفاقی می‌افتد؟

```
public class Cubes5 {
   public static int cube(int i) {
       return i * i * i;
   }


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

چند مثال از توابع

absolute value of a `double` *value*

```
public static double abs(double x) {
    if (x < 0) return -x;
    else       return  x;
}
```

primality test

```
public static boolean isPrime(int N) {
    if (N < 2) return false;
    for (int i = 2; i <= N/i; i++)
        if (N % i == 0) return false;
    return true;
}
```

hypotenuse of a right triangle

```
public static double hypotenuse(double a, double b) {
    return Math.sqrt(a*a + b*b);
}
```

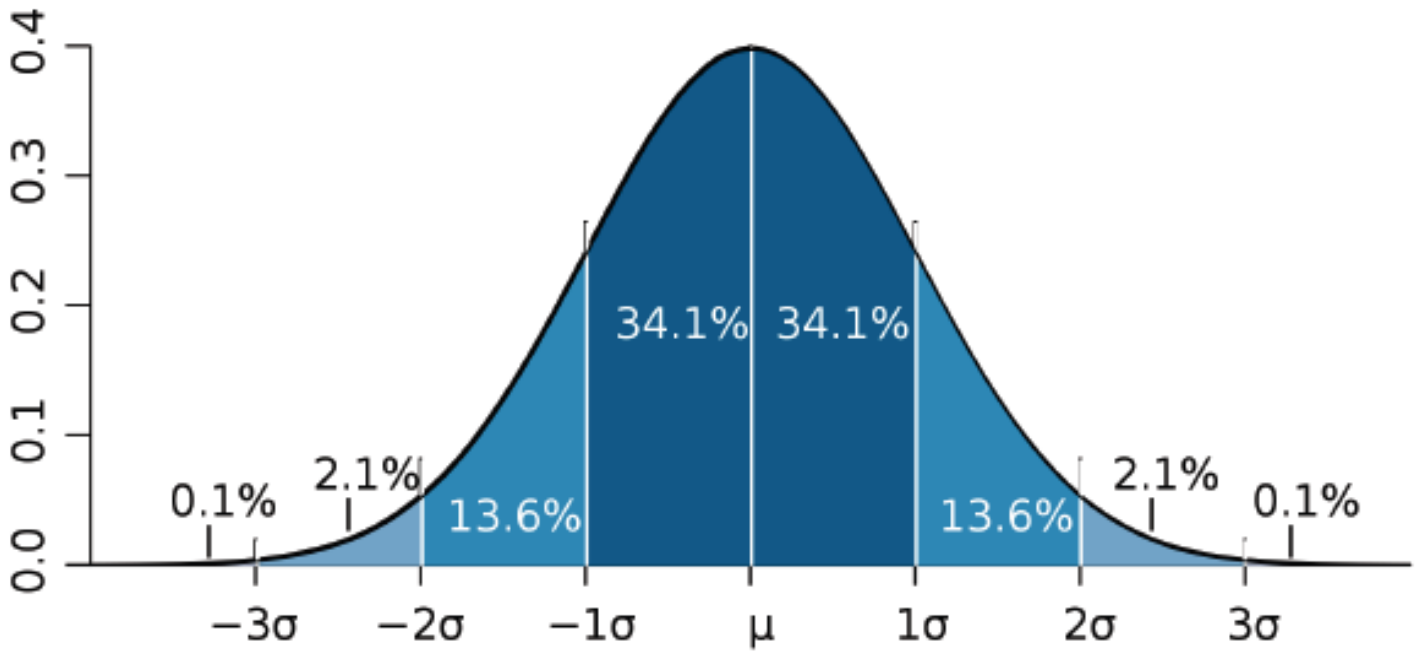
Harmonic number

```
public static double H(int N) {
    double sum = 0.0;
    for (int i = 1; i <= N; i++)
        sum += 1.0 / i;
    return sum;
}
```

uniform random integer in [0, N)

```
public static int uniform(int N) {
    return (int) (Math.random() * N);
}
```

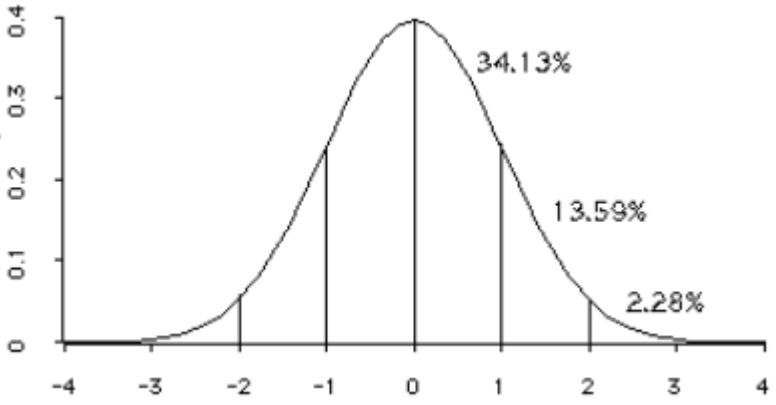
توزیع گاوسی



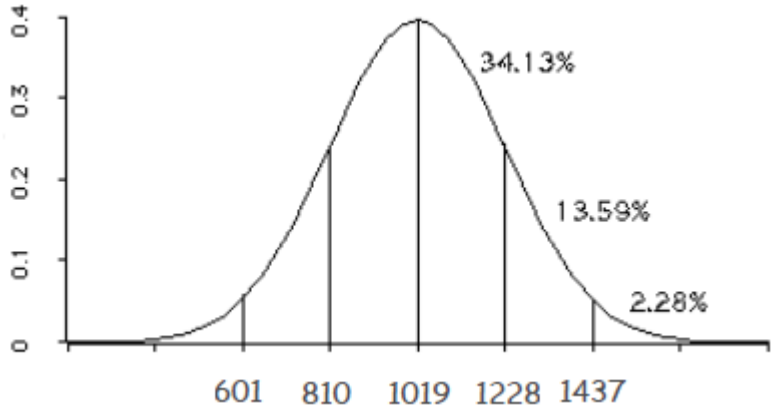
توزیع گاوسی

- توزیع گاوسی استاندارد.
  - «توزیع زنگوله شکل»
  - پایه اغلب تحلیل‌های آماری در علوم اجتماعی و فیزیکی.

- مثال. نمرات آزمون SAT در سال ۲۰۰۰ از یک توزیع گاوسی با میانگین $\mu = 1019$ و انحراف معیار $\sigma = 209$ پیروی می‌کند.



$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$



$$\phi(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

پیاده‌سازی در جاوا

- توابع ریاضی. هر زمان ممکن است از توابع پیش ساخته استفاده کنید. در غیر این صورت خودتان توابع مورد نیاز را پیاده‌سازی کنید.

```
public class Gaussian {
   public static double phi(double x) {
      return Math.exp(-x*x / 2) / Math.sqrt(2 * Math.PI);
   }

   public static double phi(double x, double mu, double sigma) {
      return phi((x - mu) / sigma) / sigma;
   }
}
```

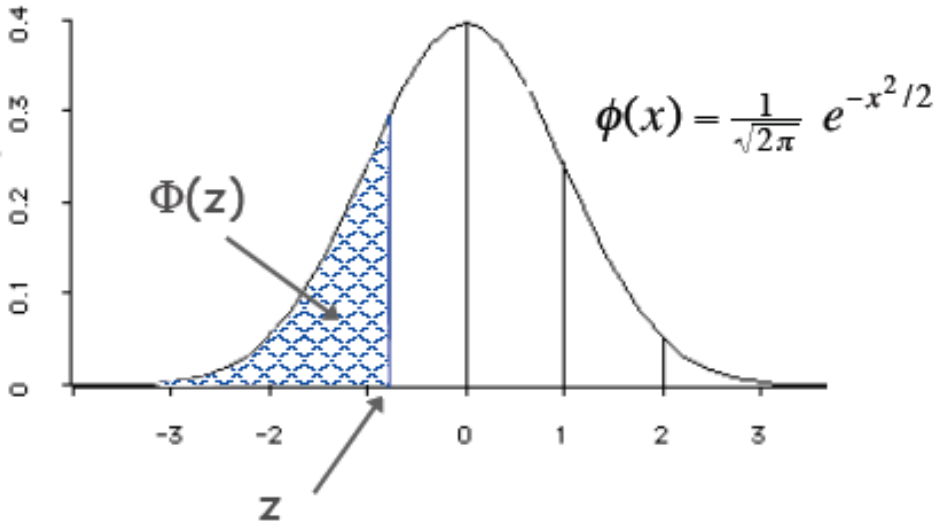
$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

$$\phi(x, \mu, \sigma) = \phi\left(\frac{x-\mu}{\sigma}\right)/\sigma$$

- بارگذاری اضافه. توابع همنام با امضاهای مختلف با یکدیگر فرق دارند.
- چندین آرگومان. یک تابع می‌تواند هر تعداد آرگومان ورودی داشته باشد.
- فراخوانی توابع دیگر. یک تابع می‌تواند توابع دیگر را فراخوانی کند.

تابع توزیع تجمعی گاوسی

- هدف. محاسبه تابع توزیع تجمعی گاوسی $\Phi(z)$.
- چالش. هیچ عبارت «شکل بسته» وجود ندارد و در کتابخانه جاوا نیز موجود نیست.



بسط تیلور

$$\Phi(x) = \int_{-\infty}^{z} \phi(x)dx$$
$$= \frac{1}{2} + \phi(z)\left(z + \frac{z^3}{3} + \frac{z^5}{3 \cdot 5} + \frac{z^7}{3 \cdot 5 \cdot 7} + \cdots\right)$$

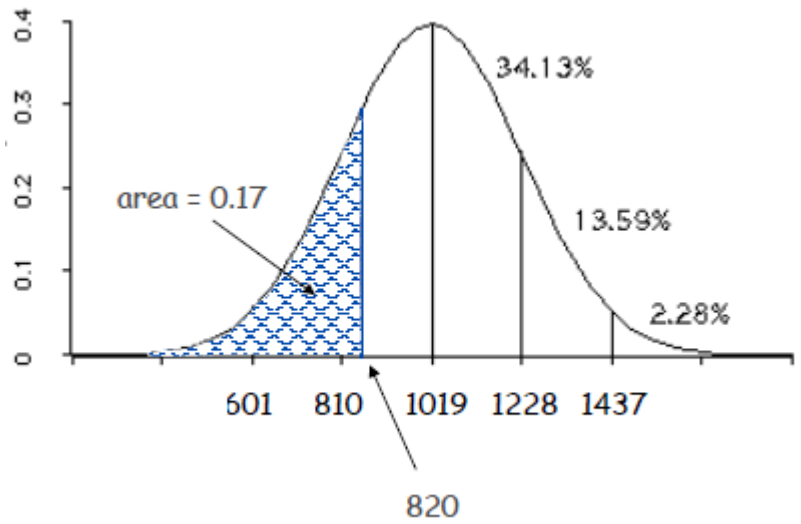
پیاده‌سازی جاوا


```
public class Gaussian {
   public static double phi(double x)
   // as before

   public static double Phi(double z) {
      if (z < -8.0) return 0.0;
      if (z >  8.0) return 1.0;
      double sum = 0.0, term = z;
      for (int i = 3; sum + term != sum; i += 2) {
         sum = sum + term;
         term = term * z * z / i;
      }
      return 0.5 + sum * phi(z);
   }

   public static double Phi(double z, double mu, double sigma) {
      return Phi((z - mu) / sigma);
   }
}
```

نمرات آزمون SAT

- پرسش. چند درصد از شرکت کنندگان در آزمون SAT واجد شرایط نیستند. برای واجد شرایط بودن، نمره شرکت کننده باید حداقل برابر با ۸۲۰ باشد.

- پاسخ.



$$\Phi(820,1019,209) \approx 0.17051$$

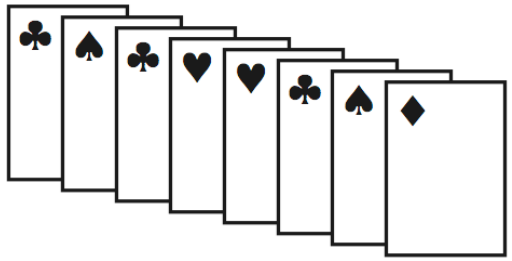
```
double fraction = Gaussian.Phi(820, 1019, 209);
```

ایجاد توابع

- توابع به شما امکان می‌دهند یک سطح جدید از انتزاع ایجاد کنید:
  - فراتر از آن چیزی که کتابخانه‌های از پیش بسته‌بندی شده در اختیار شما قرار می‌دهند.
  - هر زمان بخواهید ابزار مورد نیازتان را ایجاد می‌کنید: `Gaussian.Phi()` و ...
- فرآیند.
  - گام ۱: شناسایی یک ویژگی مفید
  - گام ۲: پیاده‌سازی
  - گام ۳: استفاده
  - گام ۳+: استفاده مجدد در هر برنامه‌ای که بخواهید.

مسئله جمع کننده کالابرگ‌ها

- مسئله جمع کننده کالابرگ‌ها. با داشتن N نوع کارت مختلف، برای این که از هر کارت (حداقل) یکی داشته باشیم چند کارت باید انتخاب شود؟



فرض می‌کنیم در هر انتخاب، احتمال انتخاب کارت‌ها با یکدیگر برابر است.

- الگوریتم شبیه‌سازی. به صورت تکراری هر بار یک عدد صحیح بین 0 و N - 1 انتخاب کن، تا زمانی که از هر کارت حداقل یکی داشته باشیم.

- پرسش. چگونه می‌توانیم بررسی کنیم که از هر نوع کارت حداقل یکی داریم؟
- پاسخ. با استفاده از یک آرایه بولی به گونه‌ای که `found[i]` تنها وقتی درست است که از کارت نوع `i` حداقل یکی داشته باشیم.

مسئله جمع کننده کالابرگ‌ها


```
public class Coupon {
   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      int cardcnt = 0;     // number of cards collected
      int valcnt = 0;      // number of distinct cards

      // do simulation
      boolean[] found = new boolean[N];
      while (valcnt < N) {
         int val = (int) (Math.random() * N);
         cardcnt++;
         if (!found[val]) {
            valcnt++;
            found[val] = true;
         }
      }

      // all N distinct cards found
      System.out.println(cardcnt);
   }
}
```

نوع کارت بعدی
(بین ۰ و N - 1)

مسئله جمع کننده کالابرگ‌ها: شکل تابعی


```
public class Coupon {
   public static int uniform(int N) {
      return (int) (Math.random() * N);
   }

   public static int collect(int N) {
      int cardcnt, valcnt = 0;
      boolean[] found = new boolean[N];
      while (valcnt < N) {
         int val = uniform(N);
         cardcnt++;
         if (!found[val]) valcnt++;
         found[val] = true;
      }
      return cardcnt;
   }

   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      StdOut.println(collect(N));
   }
}
```

`found[]`	*cards collected*
`cardcnt`	*number collected*
`valcnt`	*number that differ*
`val`	*current value*

```
% java Coupon 1000
6552

% java Coupon 1000
6481

% java Coupon 1000000
12783771
```

آرایه‌ها به عنوان آرگومان

find the maximum of the arrays value

```
public static double max(double[] a) {
    double max = Double.NEGATIVE_INFINITY;
    for (int i = 0; i < a.length; i++)
        if (a[i] > max) max = a[i];
    return  max;
}
```

dot product

```
public static double dot(double[] a double[] b) {
    double sum = 0.0;
    for (int i = 0; i < a.length; i++)
        sum += a[i] * b[i];
    return  sum;
}
```

exchange two elements in the array

```
public static void exch(String[] a, int i, int j) {
    String temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
```

shuffle the array

```
public static void shuffle(String[] a) {
    int N = a.length;
    for (int i = 0; i < N; i++)
        exch(a, i, i + uniform(N - i);
}
```

تمرین‌ها

- تمرین‌های بخش ۱-۲

```
4, 5, 8, 12, 15, 19, 21, 27, 28, 29, 32, 34, 39
```