Tölvunarfræði 1

Fyrirlestur 14: Föll í Java II

Hjálmtýr Hafsteinsson

Haust 2015

HÁSKÓLI ÍSLANDS
IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

Í síðasta fyrirlestri

- Föll (*static methods*) í Java
  - Tilgangur falla
  - Skilgreining og notkun
  - Dæmi
- Helstu eiginleikar falla

Kafli 2.1

Í þessum fyrirlestri

- Skipulag forrita með föllum
- Stærri forritadæmi
- Notkun fylkja í föllum
  - Fylki sem viðföng
  - Fylki sem skilagildi

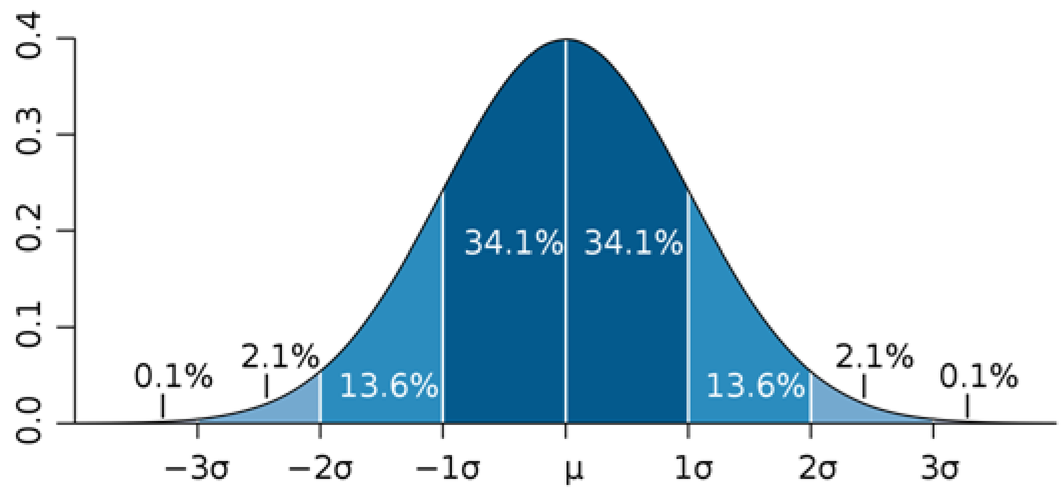
Kafli 2.1

Gauss-dreifing (*Gaussian distribution*)

- Mikið af atburðum sem koma fyrir í lífinu virðast fylgja Gauss líkindadreifingunni (*normaldreifing*)
  - Dreifingin myndar bjölluferil (*bell curve*)
  - Ferlinum sjálfum er lýst með fallinu:

Fyrir μ=0 og σ=1

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



  - Almennt:

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

μ er meðaltalið
σ er staðalfrávikið

HÁSKÓLI ÍSLANDS
IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI- OG TÖLVUNARFRÆÐIDEILD

Java föll fyrir Gauss dreifingu

- **`Math`** forritasafnið hefur ekki föll fyrir Gauss dreifingu
  - Þurfum því að skrifa þau sjálf:

```
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;
   }
}
```

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

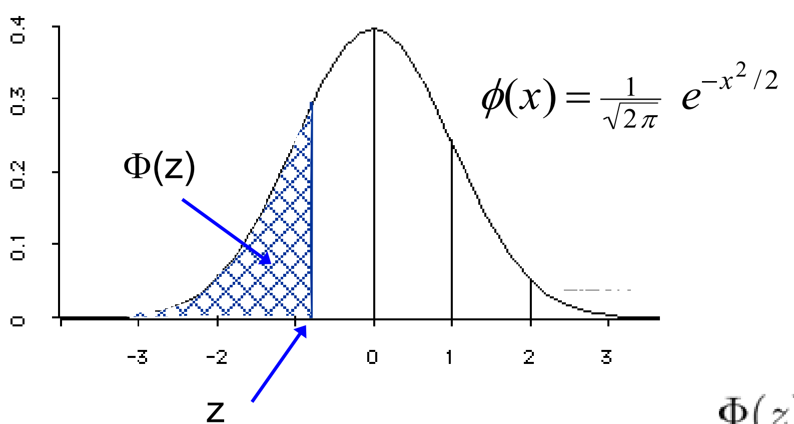
$$\phi(x, \mu, \sigma) \;=\; \phi\left(\tfrac{x-\mu}{\sigma}\right) / \;\sigma$$

Yfirhleðsla (*overloading*):
Tvö föll með sama nafn en mismunandi mynsturfar (*signature*)

```
double phi(double)
double phi(double, double, double)
```

Uppsafnaða dreififallið

- Ef við viljum fá að vita hversu hátt hlutfall hefur lægra gildi en *x*
- Dæmi:
  - Segjum að hæð sé Gauss-dreifð með μ = 178cm og σ = 8cm
  - Hve hátt hlutfall er lægri en 165cm?



Ekki til lokuð formúla fyrir Φ, notum Taylor röð

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

Java fall

```
public class Gaussian {

    public static double phi(double x)
        // eins og áður

    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);
    }
}
```

Útreikningur á Taylor röð

Yfirhlaðin útgáfa fyrir almenn μ og σ

Notkun á `Gaussian` forritasafninu

- Höfum nú forritasafnið **`Gaussian`**
  - Getum notað það svipað og **`Math`**
  - Þurfum að vísu að hafa **`Gaussian.java`** í sömu möppu
- Hæð Gauss-dreifð með með μ = 178cm og σ = 8cm
  - Hve margir eru lægri en 165cm?

```
% java Gaussian 165 178 8
0.05208127941521934
```

5% eru lægri en 165cm
(miðað við þessar forsendur)

- Einkunnir Gauss-dreifðar með μ = 70 og σ = 15
  - Hve margir falla?

```
% java Gaussian 50 70 15
0.09121121972586782
```

9% fall
(miðað við þessar forsendur)

Notkun fylkja í föllum

- Java föll geta fengið fylki sem viðföng og skilað fylkjum sem skilagilum
- Dæmi:
  - Reikna meðaltal staka í einvíðu fylki:

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

Best að nota hér **`a.length`** frekar en annað viðfang

Geymum mikið af gögnum í fylkjum − þægilegt að vinna á þeim í föllum

Föll geta breytt fylkjum!

- Viðföng eru send niður í föll sem gildi (*call-by-value*)
  - Breytingar á viðfangsbreytunni skila sér ekki upp
  - Dæmi:
    - Breytan **`a`** breytist ekkert, þó viðfangsbreytunni **`i`** sé breytt í fallinu

```
...
fall(a);
...
```

```
public static void fall(int i)
{
   i++;
}
```

- Þetta gildir ekki með fylki!
  - Það er hægt að breyta innsendum fylkjum í falli

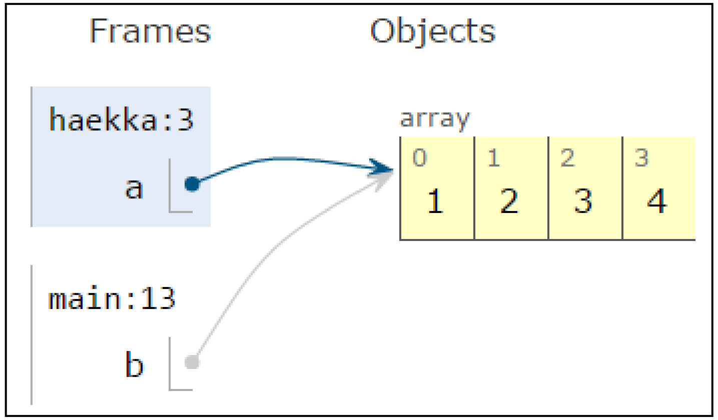
Dæmi um breytingu á fylki

- Einfalt dæmi þar sem gildi í fylki er breytt:

(Java Visualizer)

```
public static void haekka(int[] a) {
   for (int i=0; i<a.length; i++)
      a[i]++;
}
```

  - Virkar vegna þess að fylki eru samsett úr tveimur hlutum:
    - Tilvísun á gögnin
    - Gögnin sjálf



Fyrirlestraræfing

1. Hversu hátt hlutfall er utan við tvö staðalfrávik frá meðaltalinu miðað við Gauss dreifingu?

2. Skrifið fallið **`addTo(int[] a, int b)`**, sem leggur töluna **`b`** við öll stök fylkisins **`a`** (ekkert skilagildi)

3. Hvers vegna viljum við ekki afrita fylkin sjálf yfir í viðfangsbreytu falls?

Stokkun fylkja

- Getum víxlað á stökum fylkja:

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

Breytum fylkinu `a`: víxlum á stökum `a[i]` og `a[j]`

- Notum þetta fall til að stokka fylki:

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

Skilar slembiheiltölu frá 0 til `N-i-1`

HÁSKÓLI ÍSLANDS
IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI- OG TÖLVUNARFRÆÐIDEILD

Að skila fylki

- Föll geta líka skilað fylkjum sem þau búa sjálf til

(Java Visualizer)

```
public static double[] slembifylki(double a, double b, int n) {
  double[] m = new double[n];
  for (int i=0; i<n; i++)
     m[i] = a + Math.random()*(b-a);
  return m;
}
```

Fylkið er búið til inni í fallinu

Setja slembigildi í öll sætin

Skila fylkinu

Réttara að segja:
Skila tilvísun á fylkið

Sjáum meira að þessari hegðun þegar við förum í klasa (kafli 3)

Fleiri dæmi fylki í föllum

- Finna hágildi staka í fylki:

```
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;
}
```

Breytan heitir sama nafni og fallið

Góð hugmynd?

- Reikna punktfeldi (*dot product*) tveggja fylkja

```
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;
}
```

Getum svo notað þetta fall í fylkjamargföldun (*matrix multiplication*)

Skipulagning forrita

- Við lausn flókina verkefna er best að brjóta þau niður í smærri, leysanlegri, undirverkefni
- Dæmi: Baka köku
  - Búa til kökudeig
  - Setja í mót og baka
  - Skreyta köku
- Sum skrefin þarf síðan að brjóta niður í undirskref:
  - Búa til kökudeig:
    - Blanda saman þurrefnum
    - Bræða smjör
    - Hræra eggjum saman við, o.s.frv.

Uppbrot forrita

- Mikilvægt að finna hvar er hægt að brjóta upp í föll
- Algengt skipulag:
  - Innlestur (eða smíði) á gögnum
  - Útreikningur með gögnin
  - Birting á niðurstöðum

Má oft brjóta þennan þátt meira upp

- Stundum er aðalforritið (`main`-fallið) aðeins nokkrar línur
  - Nota fallsnöfnin til að lýsa því hvað þau gera
  - Fáum grófa mynd af skipulagi forritsins með því að lesa `main`-fallið

Dæmi um uppbrot forrits

- Safnaraverkefnið (*coupon collector*)
  - *N* ólíkar gerðir af spilum, hvað þarf að safna mörgum þar til þú hefur amk eitt spil af hverri gerð?
  - Notum hermun til að áætla þetta
  - Það er til stærðfræðformúla fyrir þetta verkefni:

    Væntur fjöldi er NH_N

H_N er *N*-ta þýðtalan (*harmonic number*)

Notum rökfylkið `found` til þess að athuga hvort allar tölur séu komnar:

`found[i]` er `true` ef tala `i` er komin

Uppbrot safnaraforrits

- Aðalforritið les, reiknar og skrifar:

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

Fallið `collect` gerir alla vinnuna

- Fallið `collect` notar fallið `getCoupon(N)` til fá nýtt spil

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

Eða kaupa nýjan pakka af fótboltaspilum

([Java Visualizer])

```
public class Coupon {

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

   public static int collect(int N) {
      boolean[] found = new boolean[N];
      int cardcnt = 0;
      int valcnt  = 0;

      while (valcnt < N) {
         int val = getCoupon(N);
         cardcnt++;
         if (!found[val]) valcnt++;
         found[val] = true;
      }
      return cardcnt;
   }

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

Skila slembispili á milli 0 og *N*-1

Fá nýtt spil

Ef ekki sést áður þá hækka teljara

Setjum alltaf sætið í `found` sem `true`

```
% java Coupon 100
753

% java Coupon 1000
7647

% java Coupon 10000
90640
```

Miðmisserispróf

- Haldið laugardag 10. okt. kl. 10:00 - 11:30
- Efni til prófs:
  – Kaflar 1.1 til 1.5
    - Sleppa dæmi í lok kafla 1.4 (*Self-avoiding walks*)
    - Ekki ítarlega í `StdAudio` í lok kafla 1.5
- Má koma með eitt "svindlblað" (má skrifa á báðar hliðar!)
  – Búið ykkur til blaðið sjálf − græðið mest á því
- Prófið sjálft:
  – Bæði krossaspurningar og forritunarspurningar

Undirbúningur fyrir próf

- Útbúa gott svindlblað
  - Fara yfir efnið og skrifa niður það sem þið eruð óörugg á
- Æfa sig í að skrifa stutt Java forrit á blað
  - Nota gamlar fyrirlestraæfingar, tímadæmi og heimadæmi
  - Mikið að æfingum í bókinni (*exercises*) og á heimasíðu bókarinnar
- Nota "virka" (*active*) yfirferð
  - Ekki bara lesa!
  - Skrifa niður áhugaverð atriði
  - Leysa verkefni
  - Breyta forritum

Ég heyri − ég gleymi
Ég sé − ég man
Ég geri − ég skil

Kínverskur málsháttur

Fyrirlestraræfing

4. Skrifið fallið **`min(double[] a)`**, sem er sambærilegt við fallið **`max`** hér á undan

5. Virkar þetta fall í Java?
   Ef ekki hvernig er þá hægt að víxla á tveimur heiltölum í Java?

```
public static void swap(int a, int b)
{
    int t = a;
    a = b;
    b = t;
}
```

6. Eruð þið að svindla í prófinu ef þið notið svindlblað?

Föstudagsspurning!

HÁSKÓLI ÍSLANDS
IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI- OG TÖLVUNARFRÆÐIDEILD

Samantekt

- Í þessum tíma:
  - Einkenni falla í Java
  - Stærri dæmi um föll

Kafli 2.1

- Í næsta tíma:
  - Forritasöfn
  - Einingaforritun

Kafli 2.2