Föll einfalda kóða

- Gera einn hlut, gera hann vel
- Óháð öðrum hlutum forrits

Fjölbinding (overloading) – fall hefur meira en eina gerð

```
double max(double x, double y)

int max(int x, int y)
```

Þýðandi fattar hvaða útgáfu af fallinu á að nota

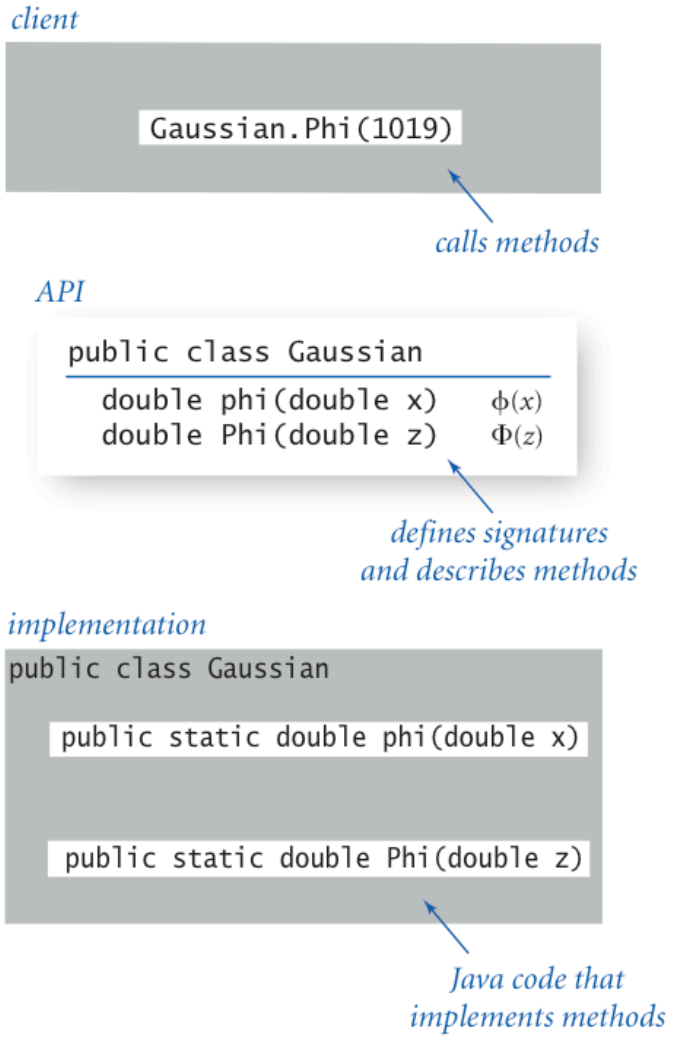
```
public class Gaussian {

   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;
   }
}
```

Föll í ólíkum skrám

Java keyrir aðeins eitt main fall í byrjun.

Getur kallað á önnur föll sem eru skilgreind í öðrum klösum (.java skrám)

Java þýðandinn (javac) sér um tengingar



Forritasöfn

Forritasöfn (libraries) hjálpa til við að endurnýta kóða

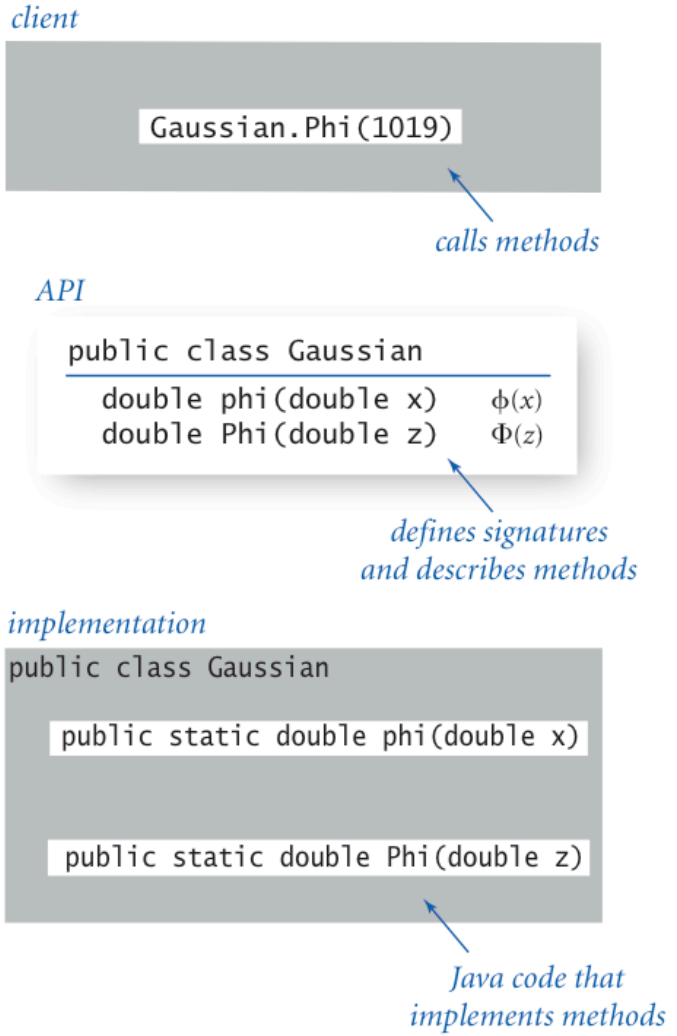
- Betra en copy/paste
- Sönnum og prófum kóða einu sinni, notum mörgu sinnum

API (Application Programming Interface)

- Lýsir **hvað** föll í forritasafni gera, ekki **hvernig**
- Java lýsing: skilatag, nafn á falli, viðföng
- Skilyrði; Notkun, Fyrir, Eftir

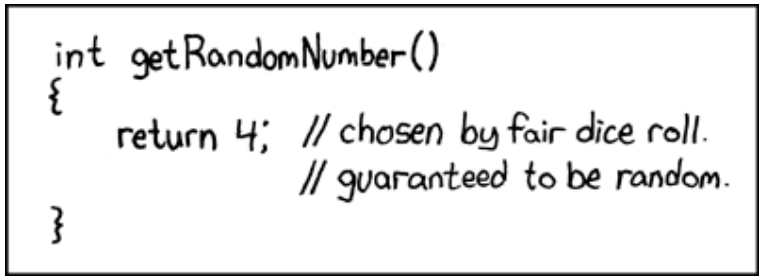
Útfærsla (implementation)

- Forrit (java kóði) fyrir föllin sem eru skilgreind í API



StdRandom forritasafn úr bókinni

public class StdRandom	
int uniform(int N)	*integer between* 0 *and* N-1
double uniform(double lo, double hi)	*real between* lo *and* hi
boolean bernoulli(double p)	true *with probability* p
double gaussian()	*normal, mean* 0*, standard deviation* 1
double gaussian(double m, double s)	*normal, mean* m*, standard deviation* s
int discrete(double[] a)	i *with probability* a[i]
void shuffle(double[] a)	*randomly shuffle the array* a[]



Slembitölur

```
public class StdRandom {

   // between a and b
   public static double uniform(double a, double b) {
      return a + Math.random() * (b-a);
   }

   // between 0 and N-1
   public static int uniform(int N) {
      return (int) (Math.random() * N);
   }

   // true with probability p
   public static boolean bernoulli(double p) {
      return Math.random() < p;
   }

   // gaussian with mean = 0, stddev = 1
   public static double gaussian()
      /* see Exercise 1.2.27 */

   // gaussian with given mean and stddev
   public static double gaussian(double mean, double stddev) {
      return mean + (stddev * gaussian());
   }

   ...
}
```

Einingaprófun

Notum main() í forritasafni til að prófa öll föll

```
public class StdRandom {
   ...

   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      for (int i = 0; i < N; i++) {
         StdOut.printf(" %2d " , uniform(100));
         StdOut.printf("%8.5f ", uniform(10.0, 99.0));
         StdOut.printf("%5b "  , bernoulli(.5));
         StdOut.printf("%7.5f ", gaussian(9.0, .2));
         StdOut.println();
      }
   }
}
```

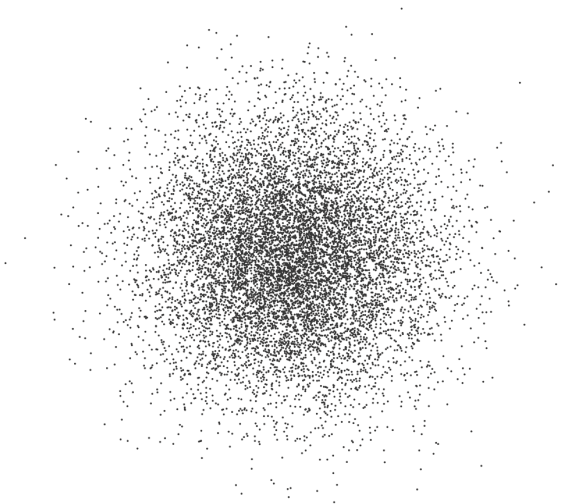
```
% java StdRandom 5
 61 21.76541  true 9.30910
 57 43.64327 false 9.42369
 31 30.86201  true 9.06366
 92 39.59314  true 9.00896
 36 28.27256 false 8.66800
```

Notkun forritasafna

StdRandom.java þarf að vera í sömu möppu

```
public class RandomPoints {
   public static void main(String args[]) {
      int N = Integer.parseInt(args[0]);
      for (int i = 0; i < N; i++) {
         double x = StdRandom.gaussian(0.5, 0.2);
         double y = StdRandom.gaussian(0.5, 0.2);
         StdDraw.point(x, y);
      }
   }
}
```

```
% javac RandomPoints.java
% java RandomPoints 10000
```



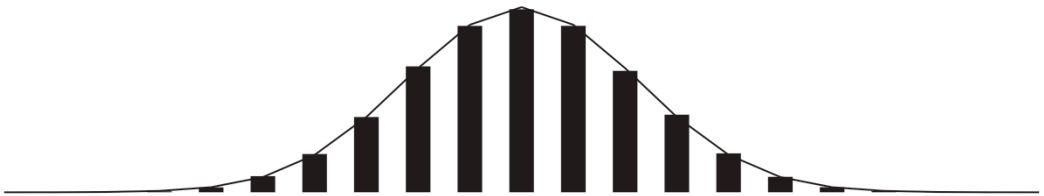
Einingaforritun

- Skiptum forriti niður í sjálfstæðar einingar
- Skrifum, sönnum og prófum hverja einingu fyrir sig
- Púslum einingum saman í stærra forrit

Dæmi: Köstum N peningum, hve of fáum við fisk?

- Lesum viðföng frá notanda
- Köstum einum pening
- Köstum N peningum og teljum fjölda fiska
- Endurtökum hermun, höldum utan um allar útkomur
- Teiknum mynd af niðurstöðum
- Berum saman við fræðilegu dreifinguna

```
% java Bernoulli 20 100000
```



Bernoullipróf

```
public class Bernoulli {
   public static int binomial(int N) {
      int heads = 0;
      for (int j = 0; j < N; j++)
         if (StdRandom.bernoulli(0.5)) heads++;
      return heads;
   }

   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      int T = Integer.parseInt(args[1]);

      int[] freq = new int[N+1];
      for (int i = 0; i < T; i++)
         freq[binomial(N)]++;

      double[] normalized = new double[N+1];
      for (int i = 0; i <= N; i++)
         normalized[i] = (double) freq[i] / T;
      StdStats.plotBars(normalized);

      double mean = N / 2.0, stddev = Math.sqrt(N) / 2.0;
      double[] phi  = new double[N+1];
      for (int i = 0; i <= N; i++)
         phi[i] = Gaussian.phi(i, mean, stddev);
      StdStats.plotLines(phi);
   }
}
```

Köstum N peningum
teljum fjölda fiska (heads)

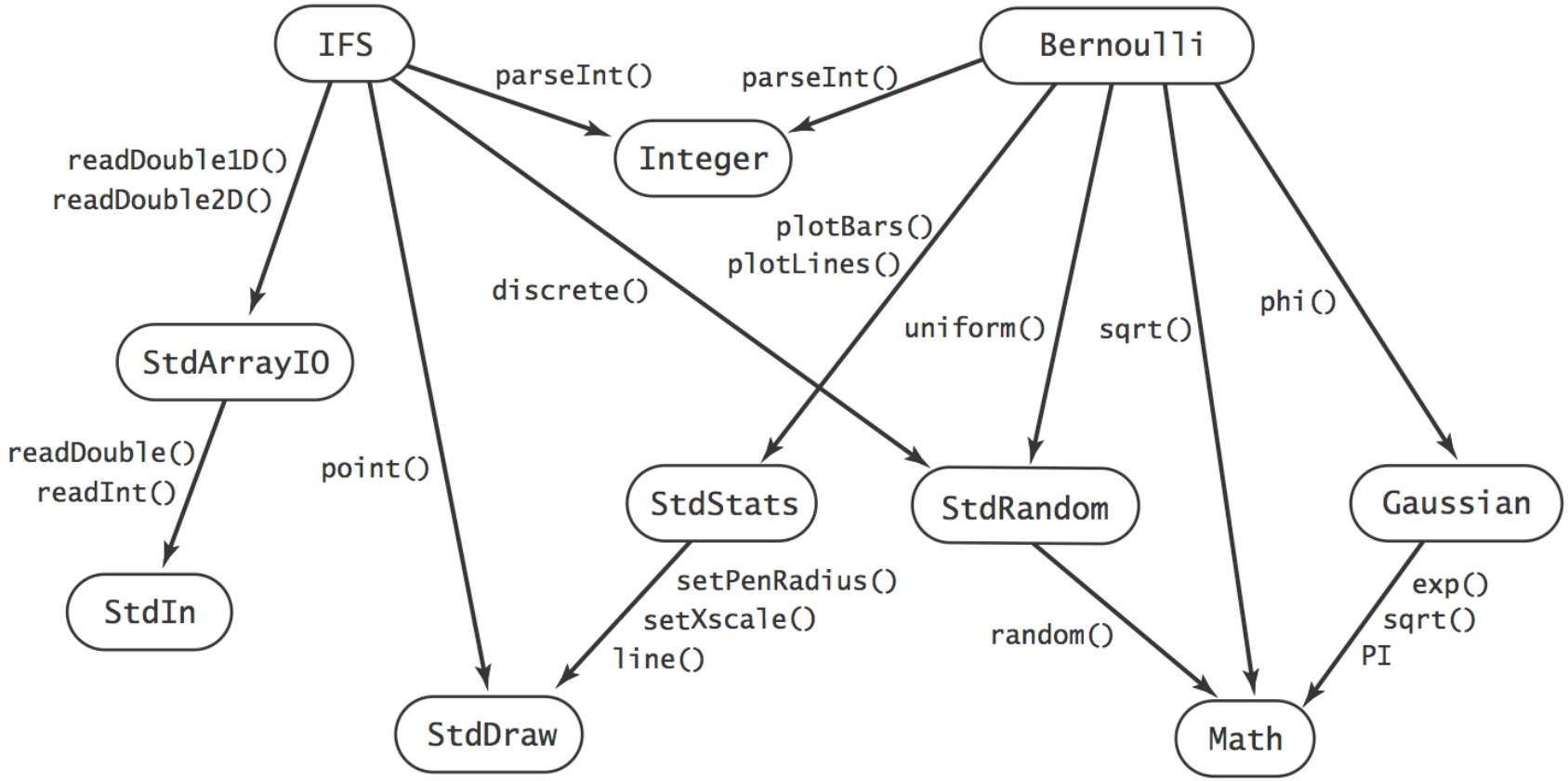
Framkvæmum T hermanir
hver með N endurtekningum

Hlutfallsleg dreifing

Fræðileg dreifing
(fyrir stór N)

Venslanet

Í einingaforritun byggjum við stærri forrit úr litlum endurnýtanlegum einingum



Gagnatög – Kafli 3.1

Gagnatag: mengi gilda og aðgerðir á gildum

Frumstæð gagnatög: örgjörvi getur unnið beint með

Tag type	Mengi gilda	lesgildi literal value	aðgerðir
`int`	heiltölur	`17` `12345`	samlagning, margföldun, …
`double`	“rauntölur”	`3.1415` `6.022e23`	samglagning, margföldun, …
`boolean`	sanngildi	`true` `false`	and, or, not

Gagnatög

Við viljum vinna með okkar eigin gagnatög

- Liti, myndir, strengi, strauma, …
- Tvinntölur, vektora, fylki, margliður, …
- Punkta, marghyrninga, …

Java leyfir okkur að skilgreina ný gagnatög (klasa) og gildi þeirra (hlutir)

Hingað til höfum við séð tvo hluti

- String – hlutir sem geyma strengi
- array – fylki, .length er hluti af klasanum!

Hlutir

Hlutur: gildi fyrir gagnatag, breytur vísa á hlut

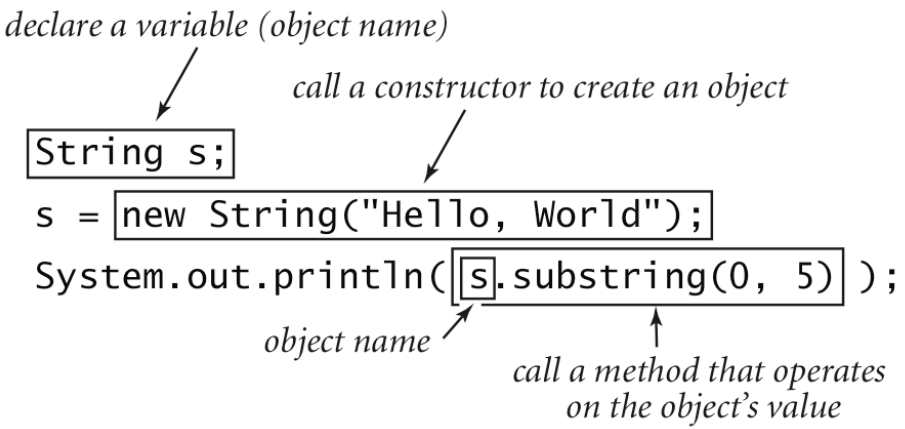
Getum skilgreint okkar eigin gagnatög og aðgerðir á þeim

Data Type	Set of Values	Operations
`Color`	24 bits	get red component, brighten
`Picture`	2D array of colors	get/set color of pixel (i, j)
`String`	sequence of characters	length, substring, compare

Smiðir og aðferðir

Búa til hlut í Java: nota `new` og nafnið á klasanum

Nota aðferð (method): nafnið á hlutnum (ekki klasa!)
punktur nafnið á aðferðinni



Myndvinnsla

Litur: skynjun augans á rafsegulbylgjum

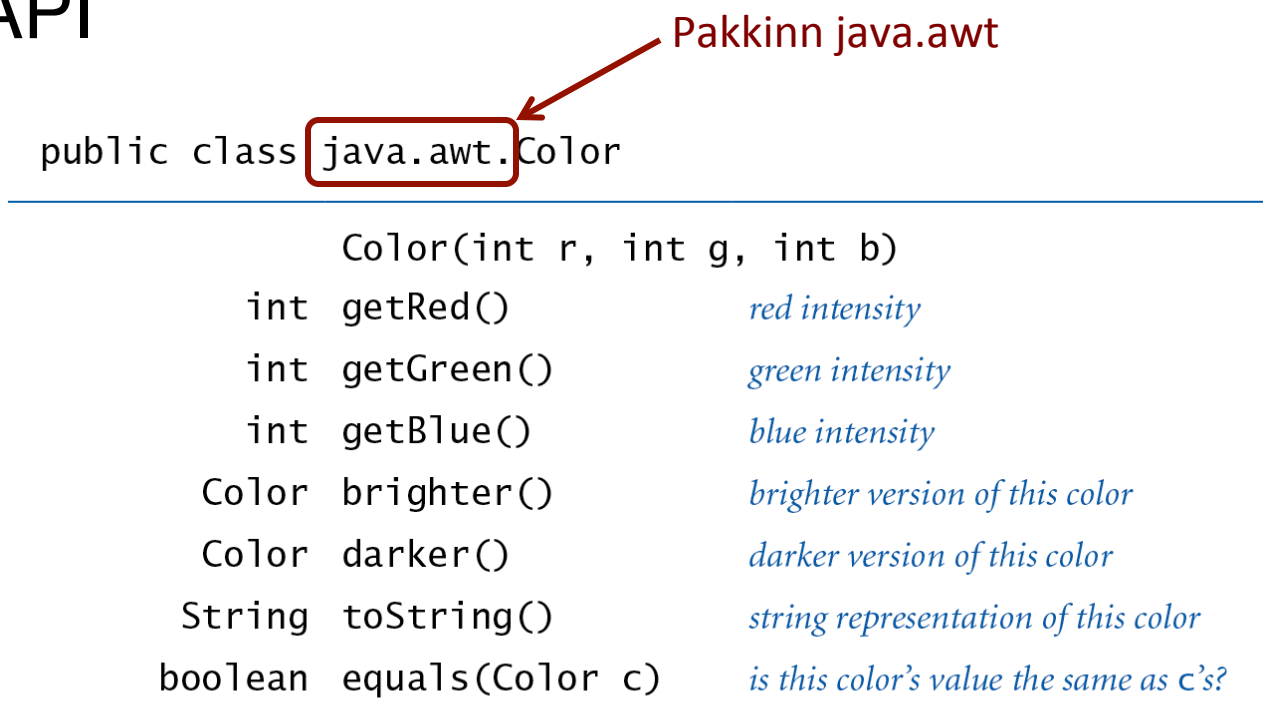
Gildi: (RGB) 256^3 mögulegar litasamsetningar,
segja til um magn af rauðum, grænum og bláum í lit á bilinu 0 til 255

R	G	B	Color
255	0	0	
0	255	0	
0	0	255	
255	255	255	
0	0	0	
255	0	255	
105	105	105	

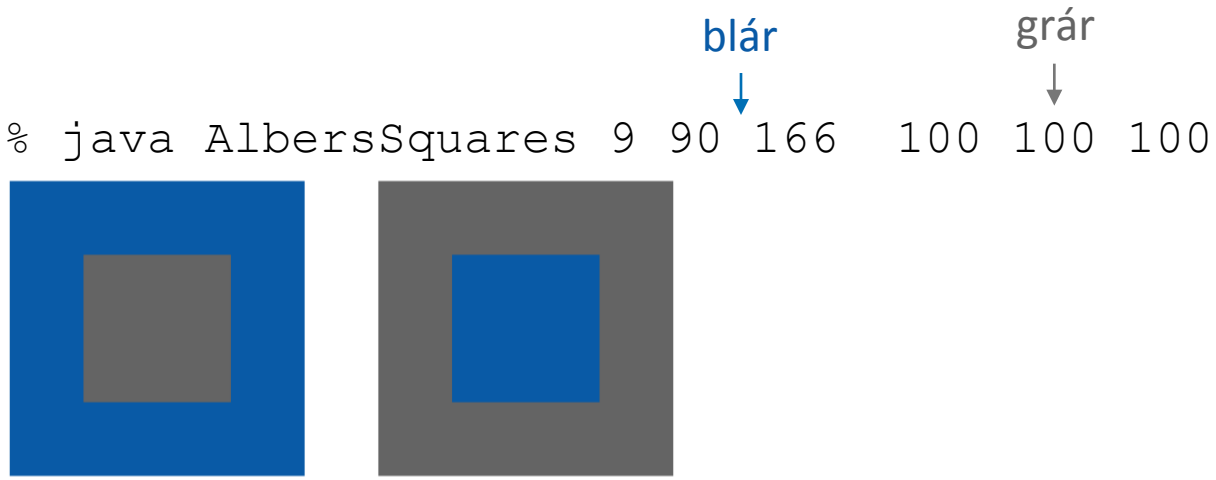
Lita klasinn

Gildi: (RGB) 256^3 mögulegar litasamsetningar, segja til um magn af rauðum, grænum og bláum í lit á bilinu 0 til 255

Java API



Albers ferningar



Litir í Java

```
import java.awt.Color;
```

notum Color safnið

```
public class AlbersSquares {
   public static void main(String[] args) {
      int r1 = Integer.parseInt(args[0]);
      int g1 = Integer.parseInt(args[1]);
      int b1 = Integer.parseInt(args[2]);
      Color c1 = new Color(r1, g1, b1);

      int r2 = Integer.parseInt(args[3]);
      int g2 = Integer.parseInt(args[4]);
      int b2 = Integer.parseInt(args[5]);
      Color c2 = new Color(r2, g2, b2);

      StdDraw.setPenColor(c1);
      StdDraw.filledSquare(.25, .5, .2);
      StdDraw.setPenColor(c2);
      StdDraw.filledSquare(.25, .5, .1);

      StdDraw.setPenColor(c2);
      StdDraw.filledSquare(.75, .5, .2);
      StdDraw.setPenColor(c1);
      StdDraw.filledSquare(.75, .5, .1);
   }
}
```

fyrsti litur

næsti litur

fyrsti ferningur

næsti ferningur

Birtustig

Birtustig litar – í réttu við hversu sterkt við skynjum lit

NTSC formúla: Y = 0.299r + 0.587g + 0.114b

```
import java.awt.Color;

public class Luminance {

   public static double lum(Color c) {
      int r = c.getRed();
      int g = c.getGreen();
      int b = c.getBlue();
      return .299*r + .587*g + .114*b;
   }

}
```

Gráskali

Gráskali: litur er “grár” þegar öll RGB gildin eru þau sömu

Breyta lit í gráskala: notum birtustig til að finna grátt gildi

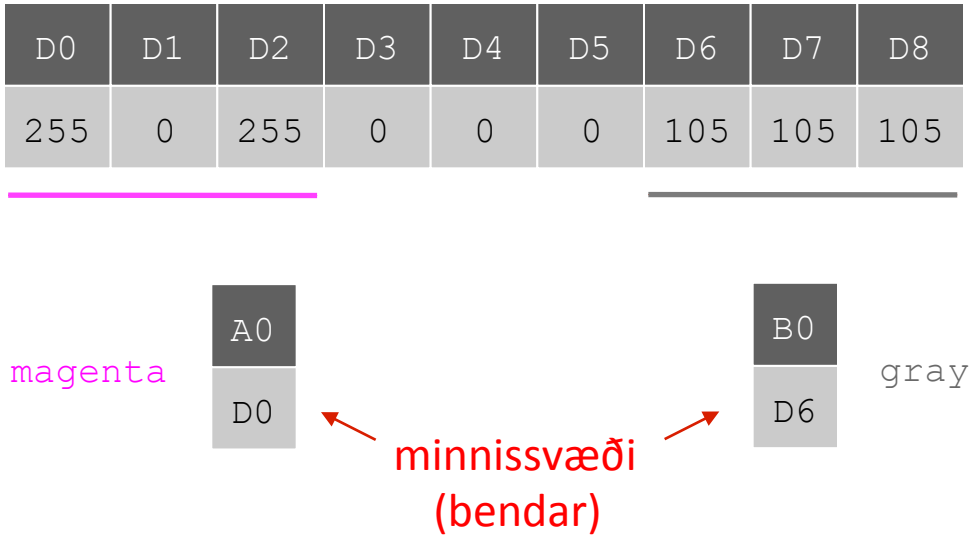
```
public static Color toGray(Color c) {
   int y = (int) Math.round(lum(c));
   Color gray = new Color(y, y, y);
   return gray;
}
```

red	*green*	*blue*	
9	90	166	*this color*
74	74	74	*grayscale version*
0	0	0	*black*

$0.299 * 9 + 0.587 * 90 + 0.114 * 166 = 74.445$

Minni og hlutir

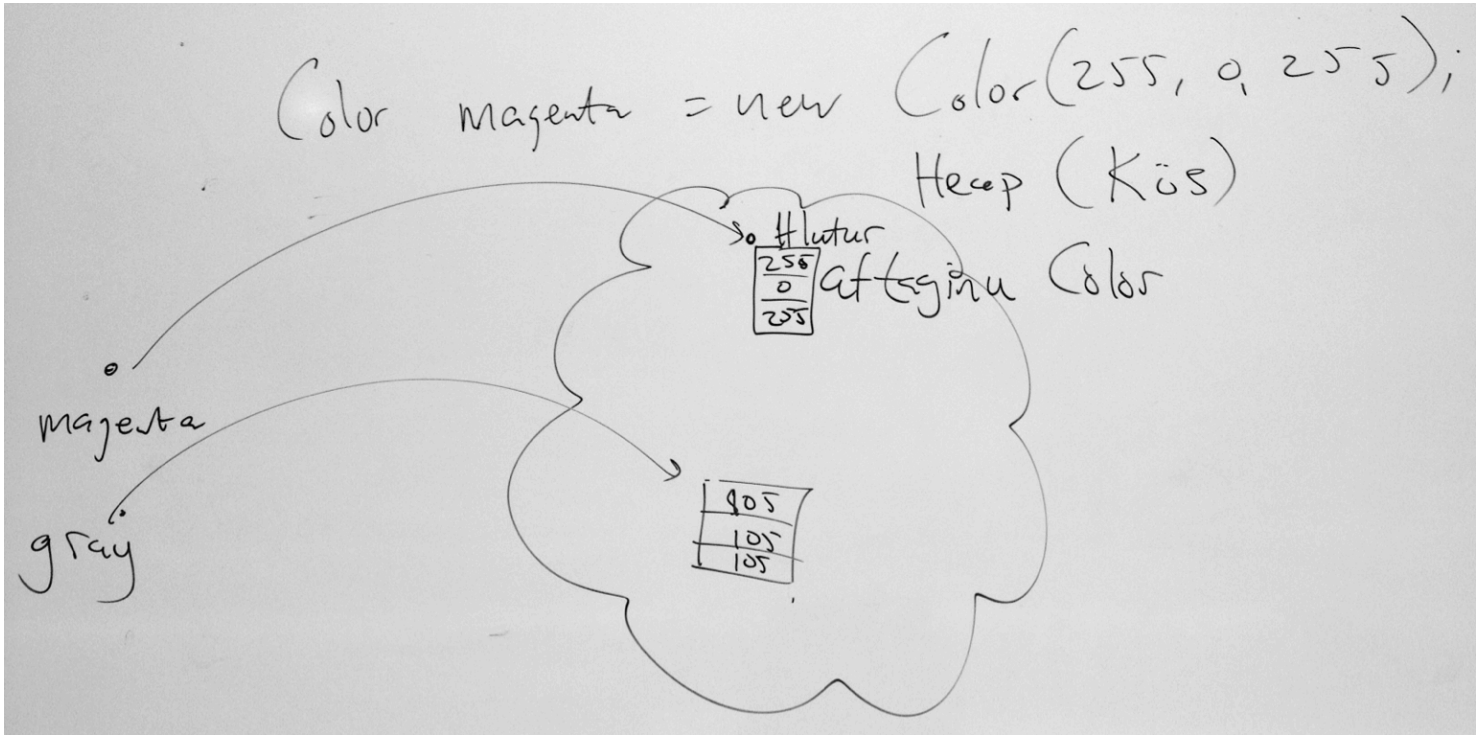
Minnisskipulag:



Breytur benda á hluti

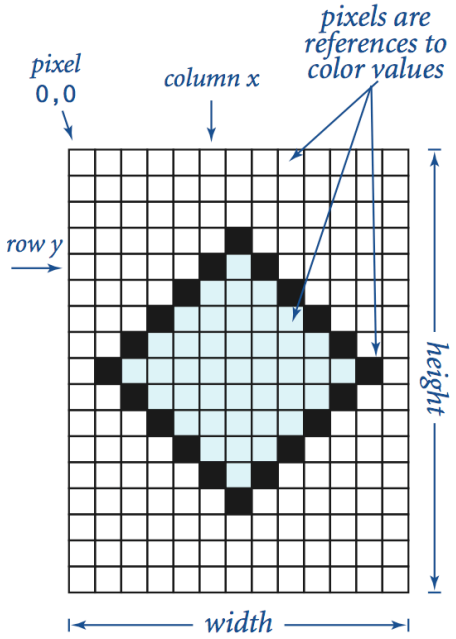
- Getum breytt gildinu á hlutnum
- Notað hlutinn sem viðfang og skilagildi
- Hlutirnir sjálfir eru geymdir í kös (heap) en breyturnar á stafla(stack)

Hlutir eru geymdir í kös, rétt eins og fylki



Myndir

Myndir: samsettar út pixlum (myndeiningum)



Gildi: “tvívítt fylki” af Color hlutum

API

```
public class Picture
```

`Picture(String filename)`	*create a picture from a file*
`Picture(int w, int h)`	*create a blank* `w`*-by-*`h` *picture*
`int width()`	*return the width of the picture*
`int height()`	*return the height of the picture*
`Color get(int x, int y)`	*return the color of pixel (x, y)*
`void set(int x, int y, Color c)`	*set the color of pixel (x, y) to* `c`
`void show()`	*display the image in a window*
`void save(String filename)`	*save the image to a file*

Myndvinnsla

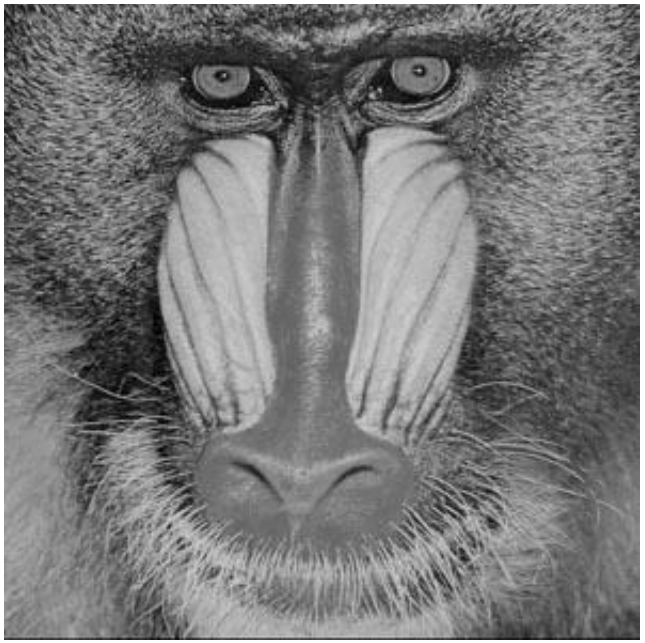
Breytum litamynd í gráskalamynd með réttu birtustigi

```
import java.awt.Color;

public class Grayscale {
   public static void main(String[] args) {
      Picture pic = new Picture(args[0]);
      for (int x = 0; x < pic.width(); x++) {
         for (int y = 0; y < pic.height(); y++) {
            Color color = pic.get(x, y);
            Color gray  = Luminance.toGray(color);
            pic.set(x, y, gray);
         }
      }
      pic.show();
   }
}
```

Myndvinnsla

Breytum litamynd í gráskalamynd með réttu birtustigi

mandrill.jpg



```
% java Grayscale mandrill.jpg
```