Tvinntölur

Viljum gagnatag fyrir tvinntölur

Mengi gilda: tvær rauntölur, raunhluti og þverhluti

API

public class Complex		
	Complex(double real, double imag)	
Complex	plus(Complex b)	*sum of this number and* b
Complex	times(Complex b)	*product of this number and* b
double	abs()	*magnitude*
String	toString()	*string representation*

$a = 3 + 4i,\ b = -2 + 3i$

$a + b = 1 + 7i$

$a \times b = -18 + i$

$|\ a\ | = 5$

Tvinntölur - prufuforrit

Einfalt prufuforrit: notar gagnatagið prófar ýmsar aðgerðir

```
public static void main(String[] args) {
   Complex a = new Complex( 3.0, 4.0);
   Complex b = new Complex(-2.0, 3.0);
   Complex c = a.times(b);
   StdOut.println("a = " + a);
   StdOut.println("b = " + b);
   StdOut.println("c = " + c);
}
```

```
% java TestClient
a = 3.0 + 4.0i
b = -2.0 + 3.0i
c = -18.0 + 1.0i
```

Megum ekki skrifa c = a * b, :(

Ekki hægt að endurskilgreina +,*,/ og [] í Java

Tvinntölur - útfærsla

```
public class Complex {

   private final double re;
   private final double im;

   public Complex(double real, double imag) {
      re = real;
      im = imag;
   }

   public String toString() { return re + " + " + im + "i"; }

   public double abs() { return Math.sqrt(re*re + im*im); }

   public Complex plus(Complex b) {
      double real = re + b.re;
      double imag = im + b.im;
      return new Complex(real, imag);
   }

   public Complex times(Complex b) {
      double real = re * b.re - im * b.im;
      double imag = re * b.im + im * b.re;
      return new Complex(real, imag);
   }
}
```

tilviksbreytur

smiður

Býr til nýja tvinntölu og skilar tilvísun

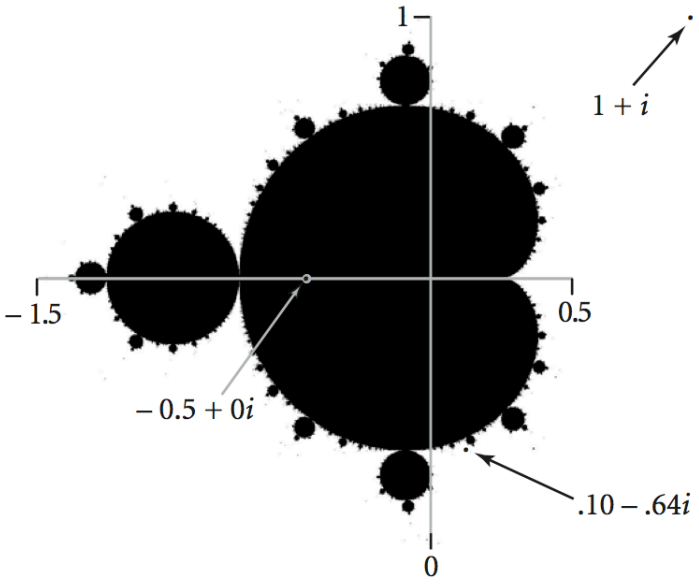
vísar í tilviksbreytu í b

aðferðir

Mandelbrot mengi

Mandelbrot mengi er mengi tvinntalna

Birting: teiknum (x,y) svart ef z = x + yi er í menginu, annars hvítt

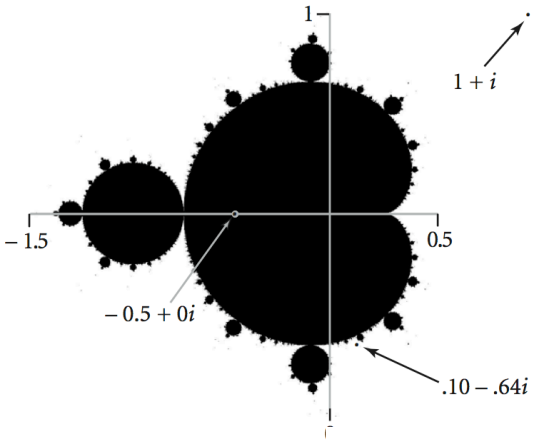


- Engin einföld formúla fyrir menginu
- Lýsum því með reikniriti

Mandelbrot mengi

Mandelbrot mengi: er z_0 í menginu?

- Ítrum: $z_{t+1} = (z_t)^2 + z_0$
- Ef $|z_t|$ er ósamleitin þá er z_0 ekki í menginu annars er z_0 í menginu



t	z_t
0	-1/2 + 0i
1	-1/4 + 0i
2	-7/16 + 0i
3	-79/256 + 0i
4	-26527/65536 + 0i
5	-1443801919/4294967296 + 0i

z = -1/2 er í mandelbrot menginu

t	z_t
0	1 + i
1	1 + 3i
2	-7 + 7i
3	1 - 97i
4	-9407 - 193i
5	88454401 + 3631103i

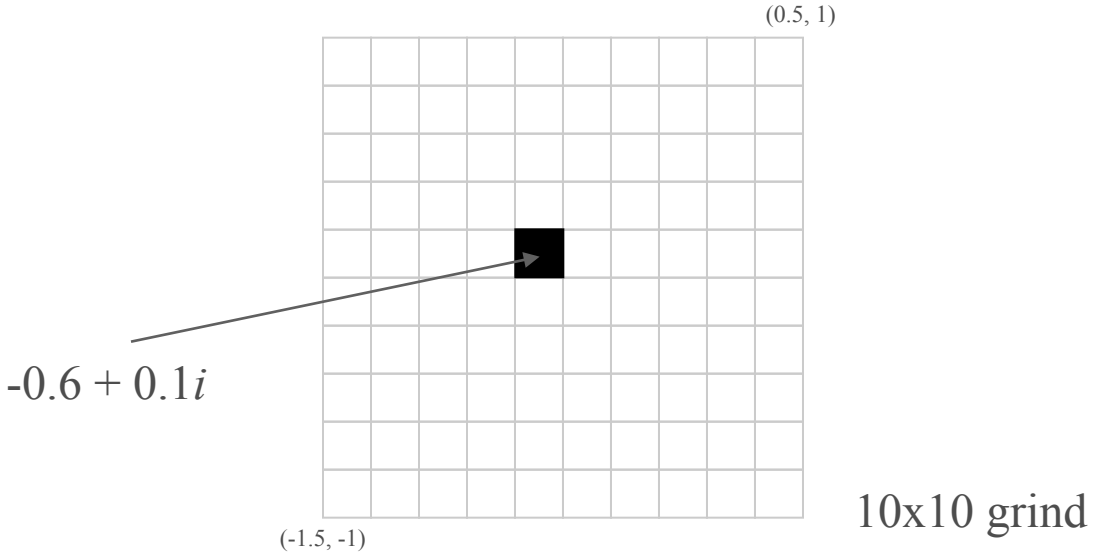
z = 1 + i ekki í Mandelbrot menginu

Birting Mandelbrot mengisins

- Getum ekki birt óendanlega marga punkta
- Getum aðeins ítrað endanlega oft

Nálgun

- Birtum fyrir NxN grind
- Ef $|z_t| > 2$ fyrir eitthvað t, þá er z ekki í Mandelbrot menginu
- Ef $|z_{255}| < 2$ þá er z "mjög líklega" í Mandelbrot menginu



Mandelbrot útfærsla

Mandelbrot fallið varpar tvinntölum í lit

- Er z_0 í Mandelbrot menginu?
- Skilar hvítum ef nei, annars svörtum

```
public static Color mand(Complex z0) {
   Complex z = z0;
   for (int t = 0; t < 255; t++) {
      if (z.abs() > 2.0) return StdDraw.WHITE;
      z = z.times(z);
      z = z.plus(z0);
   }
   return StdDraw.BLACK;
}
```

- Flottari mynd: skilum gráskala t.d. `new Color(255-t, 255-t, 255-t)`

Mandelbrot útfærsla

Plottum Mandelbrot í gráskala

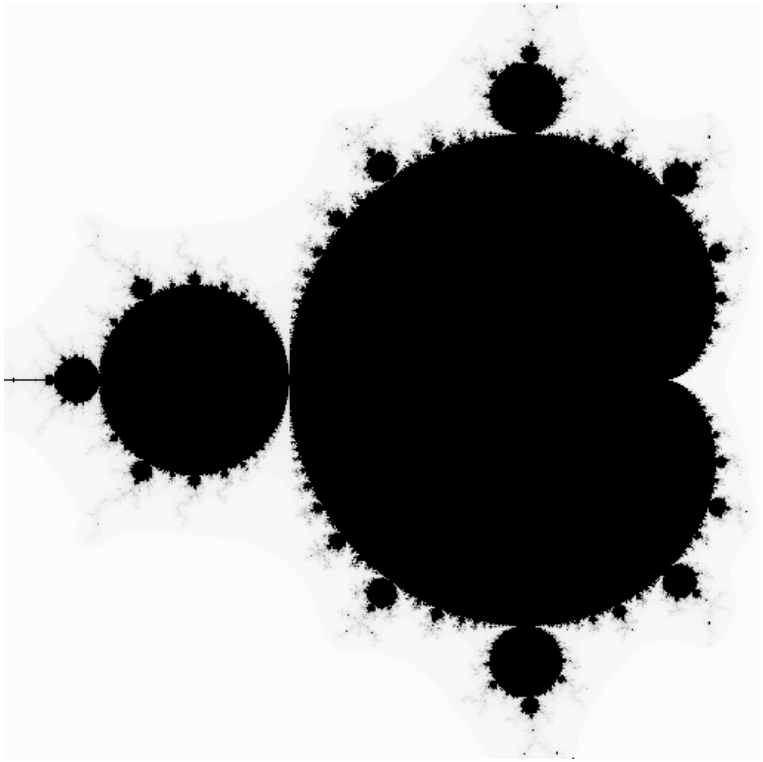
```
public static void main(String[] args) {
   double xc   = Double.parseDouble(args[0]);
   double yc   = Double.parseDouble(args[1]);
   double size = Double.parseDouble(args[2]);
   int N = 512;
   Picture pic = new Picture(N, N);

   for (int i = 0; i < N; i++) {
      for (int j = 0; j < N; j++) {
         double x0 = xc - size/2 + size*i/N;
         double y0 = yc - size/2 + size*j/N;
         Complex z0 = new Complex(x0, y0);
         Color color = mand(z0);
         pic.set(i, N-1-j, color);
      }
   }
   pic.show();
}
```

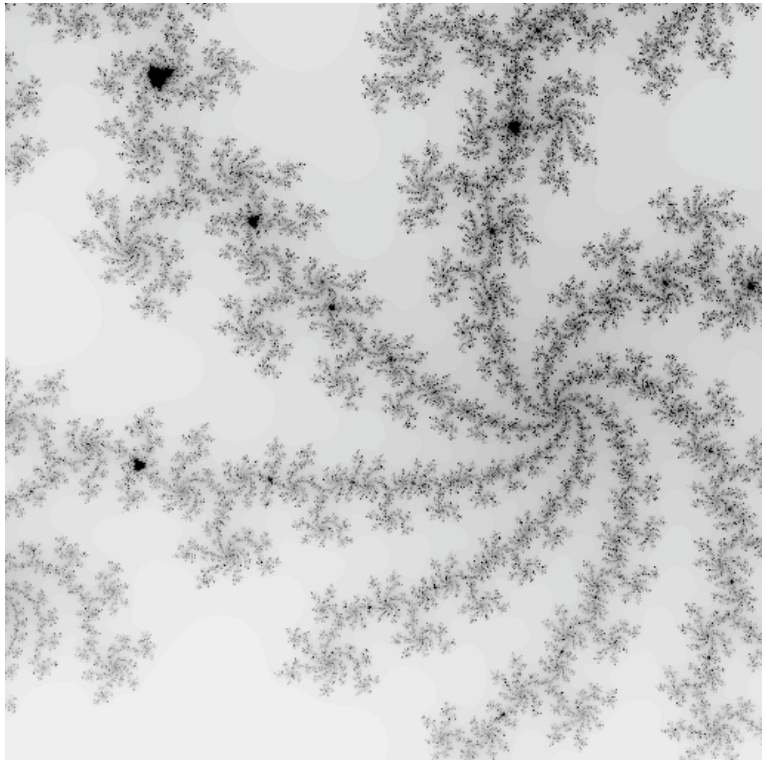
skölum myndina
til að passa á skjá

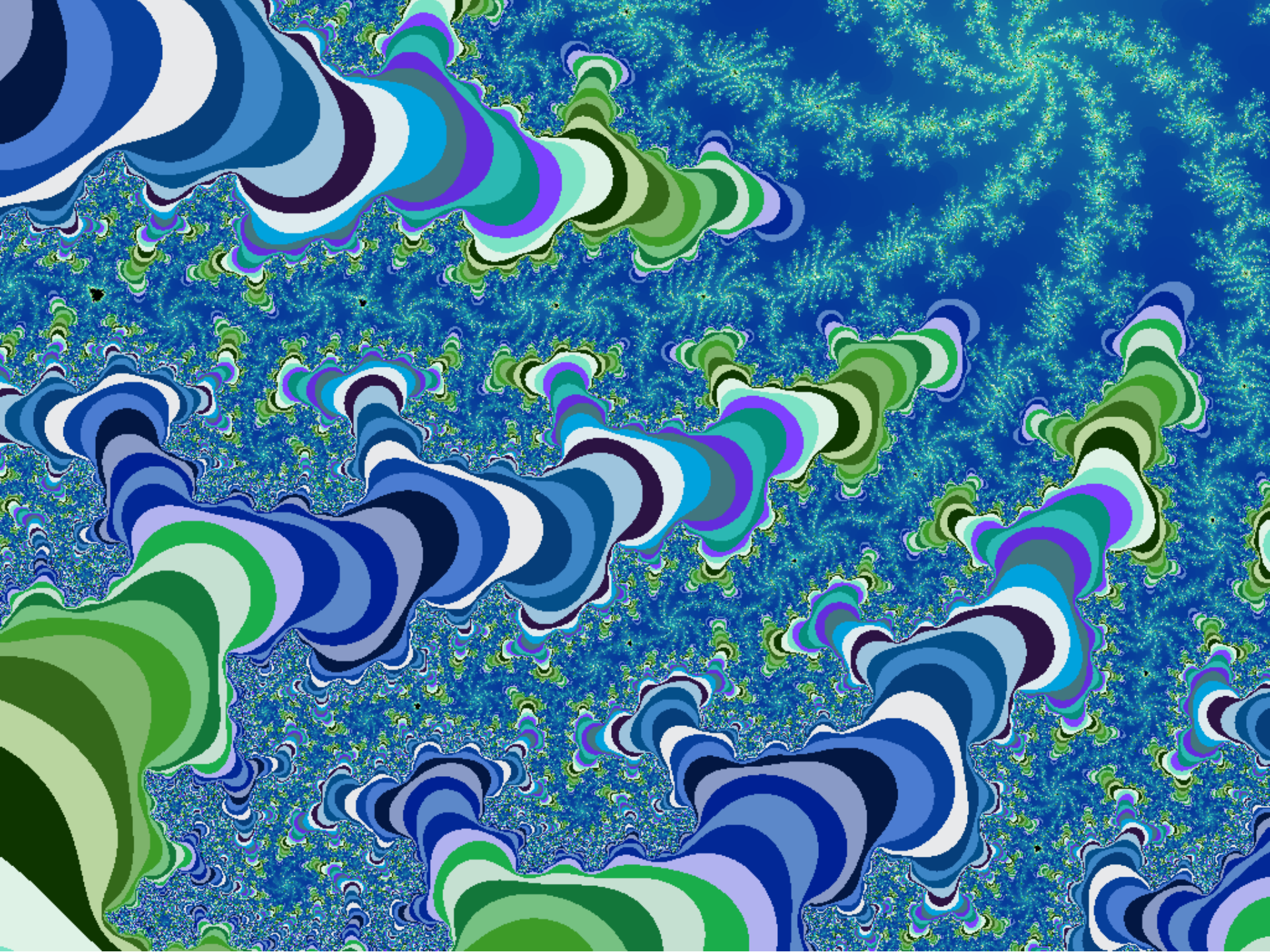
Mandelbrot mengið

```
% java Mandelbrot -.5 0 2
```



```
% java Mandelbrot .1045 -.637 .01
```





Point2D

Einfalt dæmi, gagnatag fyrir tvívíð hnit.

```
public class Point2D {
  public double x,y;
  public Point2D() {
     x = 0.0;
     y = 0.0;
  }
}
```

Point2D

Einfalt dæmi, gagnatag fyrir tvívíð hnit.

```
public class Point2D {
  public double x,y;
  public Point2D() {
     x = 0.0;
     y = 0.0;
  }
}
```

Vandamál: aðrir partar forrits geta breytt hlutnum

Point2D

```
public class Point2D {
  public double x,y;
  public Point2D() {
     x = 0.0;
     y = 0.0;
  }
}
…
Point2D p = new Point2D();
Point2D r = p;
r.y = 3.0; // nú er p líka breyttur!
```

Point2D

```
public class Point2D {
   private double x,y;
   public Point2D(double x, double y) {
       this.x = x;
       this.y = y;
   }
   public double getX() { return x;}
   public double getY() { return y;}
   public void setX(double x) { this.x = x;}
   public void setY(double y) { this.y = y;}
}
```

Vandamálið leyst?

Point2D

```
public class Point2D {
   private double x,y;
   Point2D(double x, double y) {
       this.x = x;
       this.y = y;
   }
   public double getX() { return x;}
   public double getY() { return y;}
   public void setX(double x) { this.x = x;}
   public void setY(double y) { this.y = y;}
}
```

Vandamálið leyst?

```
Point2D p = new Point2D(1.0,3.0);
r = p;
r.setX(3.0); // alveg eins og áður!
```

Point2D

Af hverju er þetta vandamál?

Ef við skilgreinum p og höldum að hluturinn sé einhverju ástandi þá geta aðrir hlutar forrits breytt því án þess að við tökum eftir því!

```
Point2D p = new Point2D(x,y);

// athugum að p sé í [0,1] x [0,1]
fall_af_hinu_illa(p) // breytir gildinu á p, p
  gæti verið
                     // hvað sem er

… // nú er p ekki í löglegu ástandi
```

Point2D

Lausnin er að láta Point2D vera óbreytanlegan (Immutable).

```
public class Point2d {
  private final double x,y;
  public Point2D(double x, double y) {
      this.x = x;
      this.y = y;
  }
  public double getX() { return x;}
  public double getY() { return y;}
}
```

Nú er ekki hægt að breyta p, við getum bara búið til nýja hluti.

String klasinn

String klasinn sem geymir strengi í Java er óbreytanlegur.

```
String s = “Hello”;
String r = s;
for (int i = 0; i < 10; i++) {
  s += i;
}

System.out.println(s);
System.out.println(r);
```

Hvað prentast út?

Óbreytanlegir hlutir kosta aðeins meira

- Þurfum að búa til nýjan hlut í hvert skipti sem við viljum breyta einhverju
- Nýir hlutir verða til í kös (heap) og kosta minni
- Ruslasafnarinn fær nóg að gera

- Auðveldara að skrifa rétt forrit með óbreytanlegum hlutum
- Margir partar forrits geta deilt sama óbreytanlega hlut (þ.e. tilvísun) án þess að hafa áhyggjur.

Rectangle

Notum Point2D til að búa til ferhyrningaklasa

```
public class Rectangle {
   // tilviksbreytur …

   public Rectangle(double x0, double y0,
                    double x1, double y1) { … }
   public Rectangle(Point2D p1, Point2D p2) { … }

   public double getWidth() { … }
   public double getHeight() { … }
   public Point2D getPos() { … }
}
```

Hvernig eigum við að geyma gildin? Hvaða tilviksbreytur?