Point2D

Point2D, einfaldur klasi fyrir punkta í tvívíðu rúmi

```
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;}
}
```

Point2D

Point2D, einfaldur klasi fyrir punkta í tvívíðu rúmi

```
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;}
}
```

Hönnun klasa

Fyrir einföld gildi (t.d. punkta, strengi, tvinntölur) notum við oftast óbreytanlega hluti

- Engar hliðarverkanir
- Viljum við að hluturinn breytist þegar við notum aðferðir?
- Notum final fyrir tilviksbreytur

Eigi hlutir eiga að vera breytanlegir, þurfum að taka skýrt fram þegar aðferðir hafa hliðarverkanir.

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?

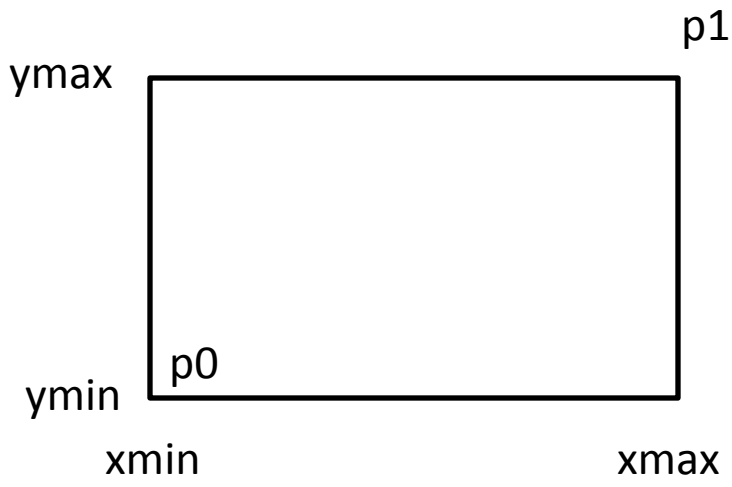
Rectangle

Hvaða tilviksbreytur?

- Breyturnar þurfa að lýsa ferhyrningnum fullkomlega
- Best er að aðeins ein möguleg lýsing komi til greina

Dæmi

- double x,y,width,height;
- double xmin,xmax,ymin,ymax;
- Point2D p0, p1;



Fastayrðing gagna

Fastayrðing gagna lýsir mögulegum gildum sem tilviksbreytur geta tekið

Dæmi

- Fyrir x,y,width,height þarf width >= 0 og height >= 0
- Fyrir xmin,xmax,ymin,ymax þarf xmin <= xmax, ymin <= ymax
- Fyrir p0,p1 þarf p0 að vera fyrir neðan og vinstra megin við p1

Í óbreytanlegum hlutum þarf smiður að tryggja að fastayrðing gagna sé uppfyllt **eftir** að hluturinn er búinn til.

Lýsingar á klösum

Fylgjum sömu reglum og fyrir föll

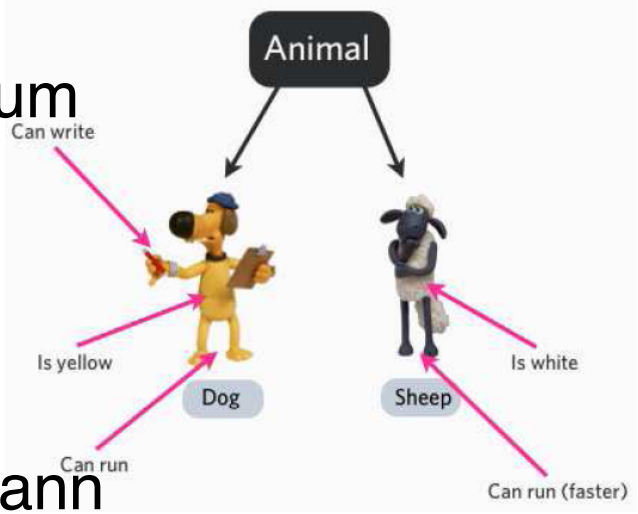
```
public class A {
   … // Tilviksbreytur
   //Fastayrðing gagna

   // Notkun: x = new A(…)
   // Fyrir:  hvað þarf að gilda um viðföngin
   // Eftir:  lýsing á hlutnum x sem búinn var til
   public A(…) { … }

   // Notkun: x = a.f(..)
   // Fyrir:  hvað þarf að gilda um viðföng aðferðarinnar
   //         og mögulega um ástand hlutsins
   // Eftir:  skilagildið x og hliðarverkanir
   public … f(…) { … }
}
```

Hlutbundin forritun (Object Oriented Programming eða OOP) byggir á eftirfarandi hugmyndafræði

- Öll forrit eru byggð á klösum og hlutum
- Finnum klasa/hluti sem tengjast vandamálinu
- Aðskilnaður: hlutir eru aðskildir frá öðrum hlutum (tilvísanir)
- Ástand: hlutur veit í hvaða ástandi hann er (tilviksbreytur)
- Hegðun: hlutur veit hvernig á að framkvæma eitthvað á sjálfum sér (aðferðir)



Hlutbundin forritun

Þrjár undirstoðir í hlutbundinni forritun

1. Upplýsingahuld (encapsulation): þurfum ekki að sjá innviði hlutar til að geta notað hann

2. Erfðir (inheritance): endurnýtum kóða með erfðum

3. Fjölbreytni (polymorphism): mismunandi hlutir geta útfært sömu aðferð á mism. hátt

Object-oriented programming is an exceptionally bad idea which could only have originated in California.

-- Edsger Dijkstra

Hlutbundin forritun

Upplýsingahuld (encapsulation)

- Felum tilviksbreytur með private
- Öll samskipti við hlut fara í gegnum API
- Getum breytt innri virkni klasa eftir á,
  aðrir hlutar forrits sjá ekki muninn
- Aðrir hlutar forrits geta ekki krukkað í klasanum

Í Rectangle klasanum skiptir ekki máli fyrir þann sem notar klasann hvernig hann er búinn til, bara hvernig hann virkar.

Hvernig er best að endurnýta kóða?

- copy/paste :)
- Með föllum
  - Við höfum séð hvernig á að raða heiltölufylkjum
    `sort(int[] a)`

Hvernig er best að endurnýta kóða?

- copy/paste :)
- Með föllum
  - Við höfum séð hvernig á að raða heiltölufylkjum
    `sort(int[] a)`

    Hvað með fleytitölur? útfærum
    `sort(double[] a)`

Hvernig er best að endurnýta kóða?

- copy/paste :)
- Með föllum
  - Við höfum séð hvernig á að raða heiltölufylkjum
    `sort(int[] a)`

    Hvað með fleytitölur? útfærum
    `sort(double[] a)`

    og ef við viljum raða String, Point2D, … ?

Hvernig er best að endurnýta kóða?

- copy/paste :)
- Með föllum
  - Við höfum séð hvernig á að raða heiltölufylkjum
    `sort(int[] a)`

    Hvað með fleytitölur? útfærum
    `sort(double[] a)`

    og ef við viljum raða String, Point2D, … ?

Við getum skrifað fall sem raðar hlutum í vaxandi röð án þess að vita hvaða hlut við erum að raða

Skil

Skil (interface) er notað til að lýsa hvaða aðferðir klasi útfærir.

Þetta er útfært í fjórum hlutum

- skilin sjálf eru skilgreind í sér .java skrá
- Klasi A tekur fram að hann útfæri skilin
- fall f tekur sem viðföng hvaða hlut sem er, sem útfærir skilin
- annar partur af forritinu kallar á f með tilviki af klasanum A

Dæmi: sort fallið fyrir hluti

1. Skilgreinum skilin Comparable með aðferðina compareTo í skránni Comparable.java

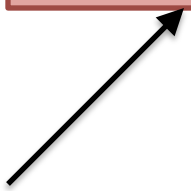
```
public interface Comparable {
  public int compareTo(Object o);
}
```

a.compareTo(b) Skilar heiltölu < 0 ef a < b, > 0 ef a > b og 0 annars

Dæmi: sort fallið fyrir hluti

1. Skilgreinum skilin Comparable með aðferðina compareTo í skránni Comparable.java

```
public interface Comparable {
  public int compareTo(Object o);
}
```



Allir hlutir **erfa** frá Object klasanum, meira um það síðar

2. Látum Klasa útfæra skil, t.d. Point2D

```
public class Point2D implements Comparable {
   …

   public int compareTo(Object o) {
       Point2D b = (Point2D) o;
       if (this.y < b.y) return -1;
       if (this.y > b.y) return +1;
       if (this.x < b.x) return -1;
       if (this.x > b.x) return +1;
       return 0;
   }
}
```

this er minna en b
ef y hnit this.y er minna
(x hnit ráða ef y hnit eru eins)

Hvernig mætti skilgreina "<" öðruvísi?

a.compareTo(b) Skilar heiltölu < 0 ef a < b, > 0 ef a > b og 0 annars

2. Látum Klasa útfæra skil, t.d. Point2D

```
public class Point2D implements Comparable {
   …

   public int compareTo(Object o) {
      Point2D b = (Point2D) o;
      if (this.y < b.y) return -1;
      if (this.y > b.y) return +1;
      if (this.x < b.x) return -1;
      if (this.x > b.x) return +1;
      return 0;
   }
}
```

this er minna en b
ef y hnit this.y er minna
(x hnit ráða ef y hnit eru eins)

Hvernig mætti skilgreina "<" öðruvísi?

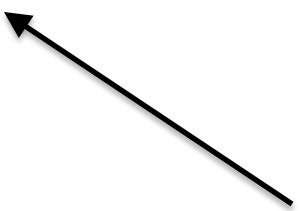
gamaldags Java kóði, þetta þurfti að gerast þar til í Java 1.5

3. sort fallið skrifað með tilliti til Comparable

```
public static void sort(Comparable[] a) {
   for (int i = 1; i < a.length; i++) {
       for (int j = i; j > 0; j--) {
          if (a[j-1].compareTo(a[j]) > 0) {
             swap(a,j-1,j);
          } else {
             break;
          }
       }
   }
}
```

3. sort fallið skrifað með tilliti til Comparable

```
public static void sort(Comparable[] a) {
   for (int i = 1; i < a.length; i++) {
       for (int j = i; j > 0; j--) {
          if (a[j-1].compareTo(a[j]) > 0) {
             swap(a,j-1,j);
          } else {
             break;
          }
       }
   }
}
```



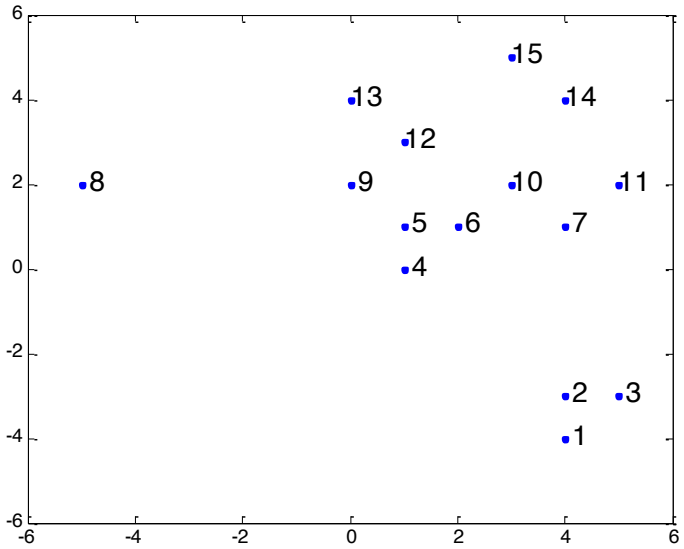
Þetta er það eina sem breytist í kóðanum, áður var a[j-1] > a[j]

Skil

4. compareTo aðferðin í sort er tengd við aðferðina í Point2D þegar við notum sort fallið

```
Point2D[] points = new Point2D[15];
… // fyllum fylkið af punktum

sort(points);// raðar í
             //  rétta röð
```



Fjölbreytni

Í Java getum við notað skil (interface) til að útfæra fjölbreytni og endurnýta kóða

Tvær aðferðir í viðbót

- Erfðir – inheritance (subtyping)

- Sniðmát - generics