Et hierarki av typer

INF101 forelesning 10. februar 2023

Torstein Strømme

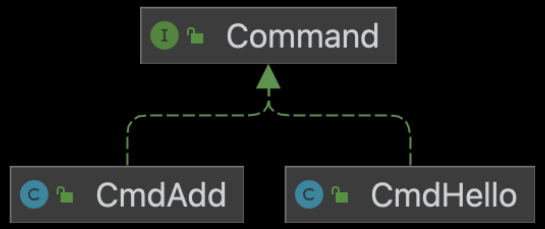
Stikkord: arv, Object, rutenett, grafikk

Grensesnitt

tilbakeblikk

```java
public String executeCommand(String cmdName, String[] args) {
  Command cmd = this.allCommands.get(cmdName);
  if (cmd != null) {
    return cmd.run(args);
  }
  // ...
}
```

klientkoden fungerer for alle objekter med typen Command




```java
public interface Command {
  String run(String[] args);
  String getName();
}
```

implementerer

```java
public class CmdHello implements Command {
  @Override
  public String run(String[] args) {
    return "Hello, world!";
  }
  @Override
  public String getName() {
    return "hello";
  }
}
```

```java
public class CmdSum implements Command {
  @Override
  public String run(String[] args) {
    int sum = 0;
    for (String arg : args) {
      sum += Integer.parseInt(arg);
    }
    return String.valueOf(sum);
  }

  @Override
  public String getName() {
    return "sum";
  }
}
```

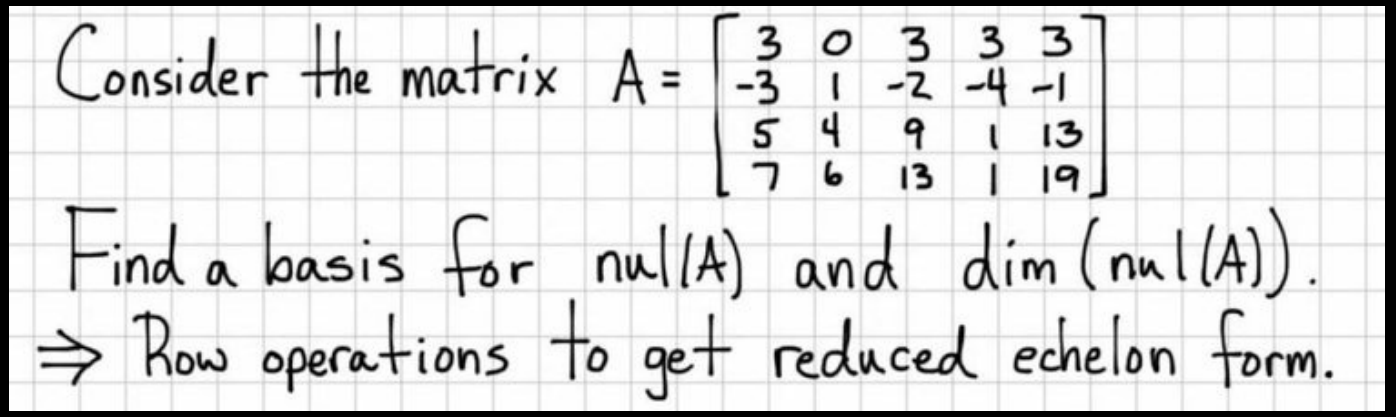
I dag

- Rutenett
- Arv
  - Arv mellom grensesnitt
  - Arv mellom klasser
- Grafikk

Matriser

padlet.com/torsteins/matrise

- En matrise er et rutenett med tall



- Hvilke metoder bør et grensesnitt for en matrise ha?

Matriser

En metode for å

- endre tall på en posisjon
- se på tallet i en posisjon
- få vite antall rader
- få vite antall kolonner
- iterere gjennom alle verdier

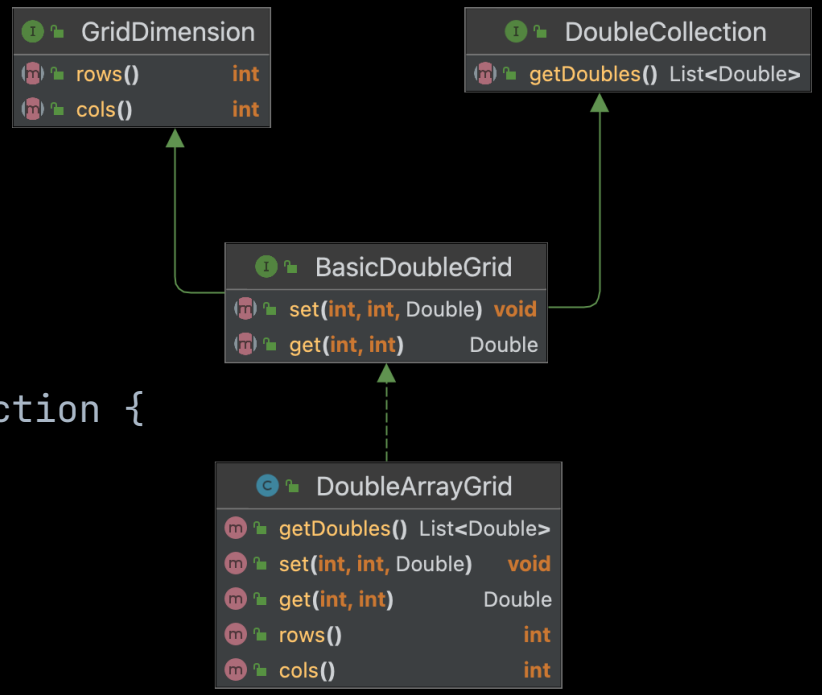
- summere med annen matrise
- trekke fra en annen matrise
- multiplisere med annen matrise
- finne egenverdi, egenvektor, nullrom, egenrom
- transponere
- finne invers
- løse ligningssystem
- ...

Rutenett

```java
public interface BasicDoubleGrid {

  /** Get the number of rows in the grid. */
  int rows();

  /** Get the number of columns in the grid. */
  int cols();

  /** Get the value at the given row and column. */
  Double get(int row, int col);

  /** Set the value at the given row and column. */
  Double set(int row, int col, Double value);

  /** Get all the values in the grid as a list. */
  List<Double> getDoubles();
}
```

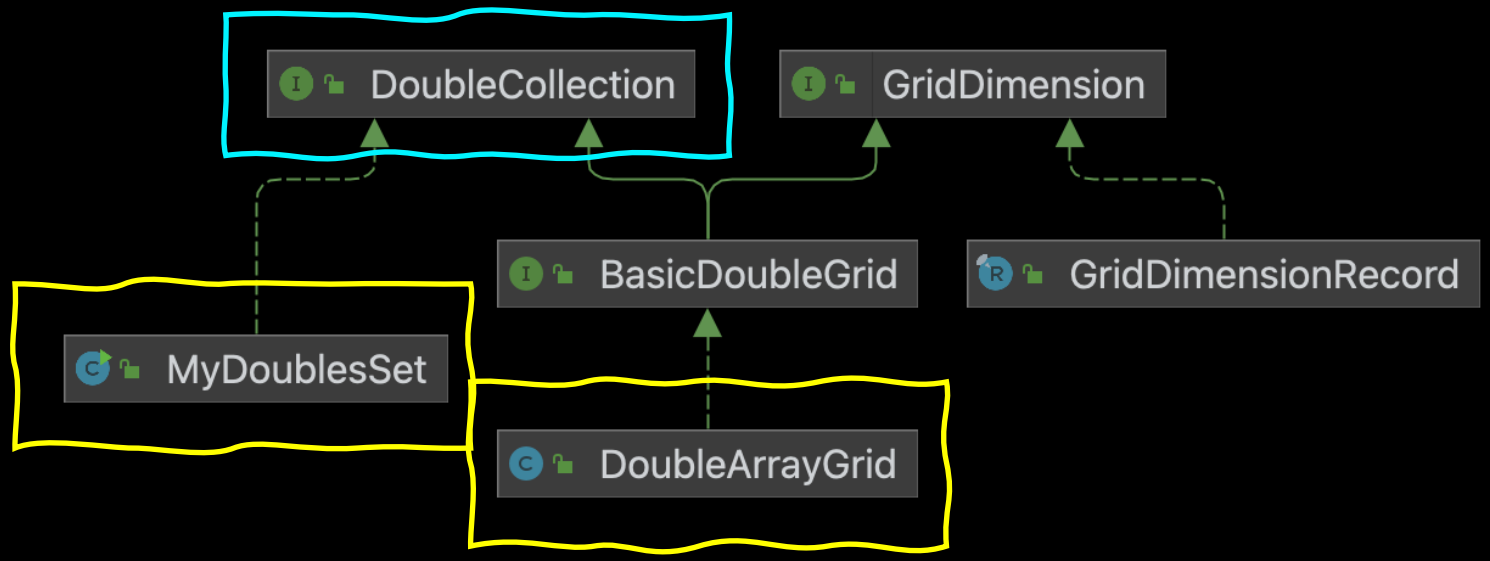
Arv mellom grensesnitt

```java
public interface GridDimension {
  int rows();
  int cols();
}

public interface DoubleCollection {
  List<Double> getDoubles();
}

public interface BasicDoubleGrid extends GridDimension, DoubleCollection {
  void set(int row, int col, Double value);
  Double get(int row, int col);
}


public class DoubleArrayGrid implements BasicDoubleGrid {
  /* ... */
  @Override public void set(int row, int col, Double value) { /* ... */ }
  @Override public Double get(int row, int col) { /* ... */ }
  @Override public int rows() { /* ... */ }
  @Override public int cols() { /* ... */ }
  @Override public List<Double> getDoubles () { /* ... */ }
}
```



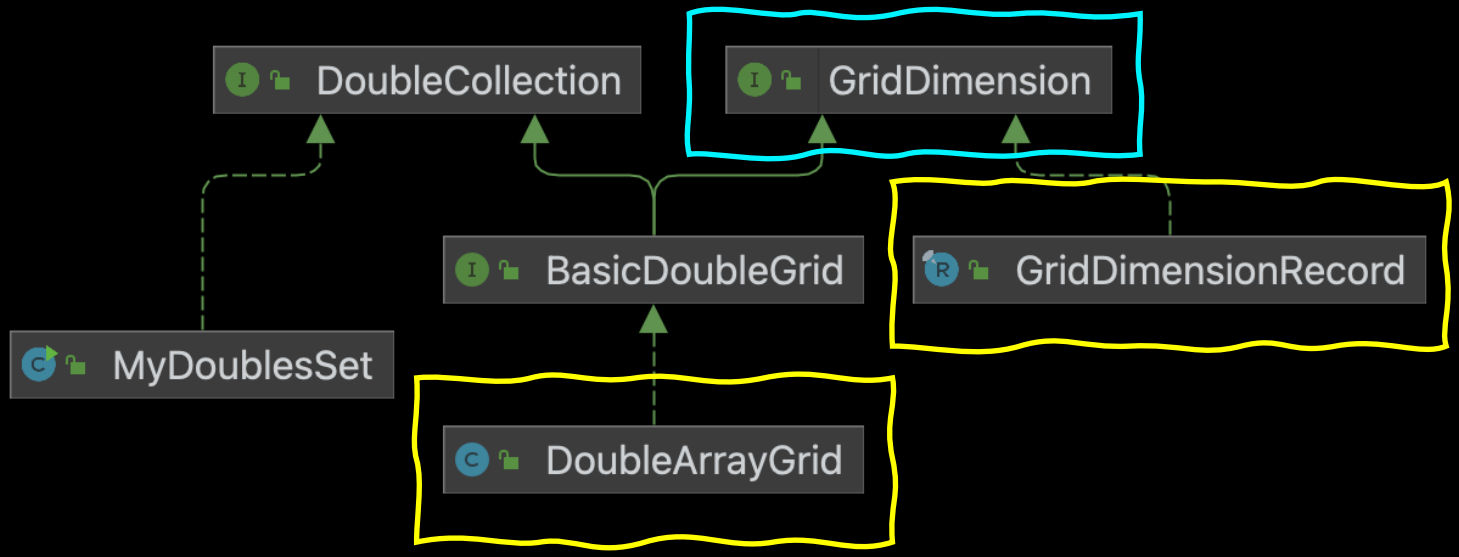
Klassen som implementerer må implementere *alle* metoder i arvede grensesnitt

Et hierarki av typer




```
public static Double findLargest(DoubleCollection collection) {
  Double largest = null;
  for (Double d : collection.getDoubles()) {
    if (largest == null || d > largest) {
      largest = d;
    }
  }
  return largest;
}
```

Et hierarki av typer



Klientkoden benytter type så høyt oppe i hierarkiet som mulig, slik at koden kan gjenbrukes

```
public static double getWidthPerCell(double totalWidth, GridDimension dim, int margin) {
  return totalWidth - ((double) (margin * (dim.cols() + 1))) / dim.cols();
}
```

Prinsipper for bruk av grensesnitt

«Open for extension, closed for modification»

- Det er bedre å være avhengig av et grensesnitt enn en klasse.

«Dependency inversion principle»

- Klientkodens behov bestemmer hva grensesnittet bør være.

«Interface segregation principle»

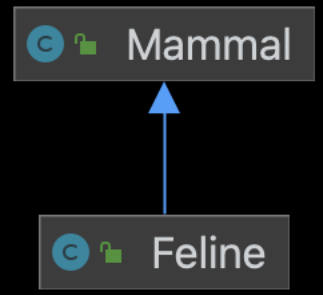
- Ikke bruk et grensesnitt som er større enn nødvendig.
- Mange små grensesnitt er bedre enn ett stort.

Arv mellom klasser

- En klasse B kan *utvide* en annen klasse A
  - Klassen A er *superklassen* til B
  - Klassen B er en *subklasse* av A
  - Klassen B får automatisk alle metoder og feltvariabler definert i A ferdig implementert
  - Typen B er en undertype av A i typehierarkiet

Arv

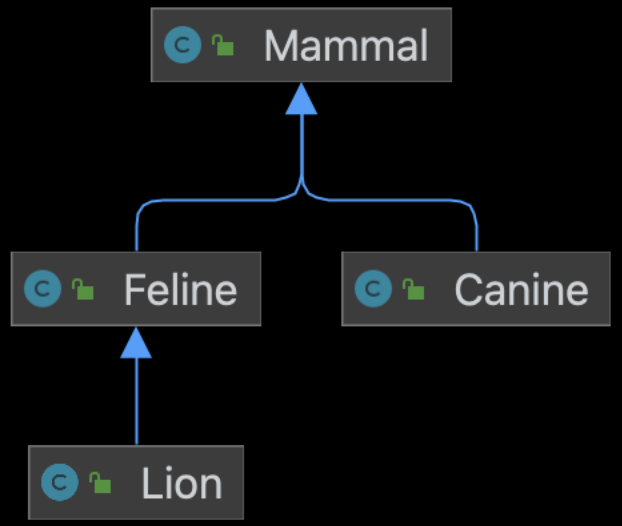
```java
public static void main(String[] args) {
  Mammal adam = new Mammal("Adam");
  Mammal kitty = new Feline("Kitty");

  eatThinkSleep(adam);
  eatThinkSleep(kitty);
}

public static void eatThinkSleep(Mammal m) {
  m.eat("meat");
  m.eat("veggies");
  m.speak();
  m.drinkMilk();
}
```




```java
public class Mammal {
  protected final String name;

  public Mammal(String name) {
    this.name = name;
  }
  public void eat(String food) {
    System.out.println(this.name + ": Yum, this " + food + " is delicious!");
  }
  public void speak() {
    System.out.println(this.name + ": (sits in silence)");
  }
  public void drinkMilk() {
    System.out.println(this.name + ": Mmmmm, milk!");
  }
}
```

```java
public class Feline extends Mammal {

  public Feline(String name) {
    super(name);
  }

  @Override
  public void eat(String food) {
    if (food.equals("veggies")) {
      System.out.println(this.name + ": Yac
      return;
    }
    super.eat(food);
  }
}
```

Arv blir en del av typehierarkiet



Arv mellom klasser

- Arv mellom klasser kan spare mye kode, men gjør særlig koden i superklassen krevende å vedlikeholde/modifisere.
- Kravene for å kunne bruke arv uten å potensielt innføre alvorlige bugs er krevende å verifisere

Liskovs substitusjonsprinsipp

> Hvis S er en subklasse av T, da må alt som kan bevises om T også kunne bevises om S

Arv i Java

- Java benytter arv en del i standardbiblioteket
- Alle objekter arver
  - Hvis man ikke angir noe selv, arver man klassen Object
  - Object sitter helt øverst i typehierarkiet
- Det er vanskelig å skrive klasser som blir arvet – benytt gjerne final for å angi at klassen du skriver ikke er laget for å arves.

- Vi må arve for å bruke rammeverket Swing for å lage desktop-applikasjoner

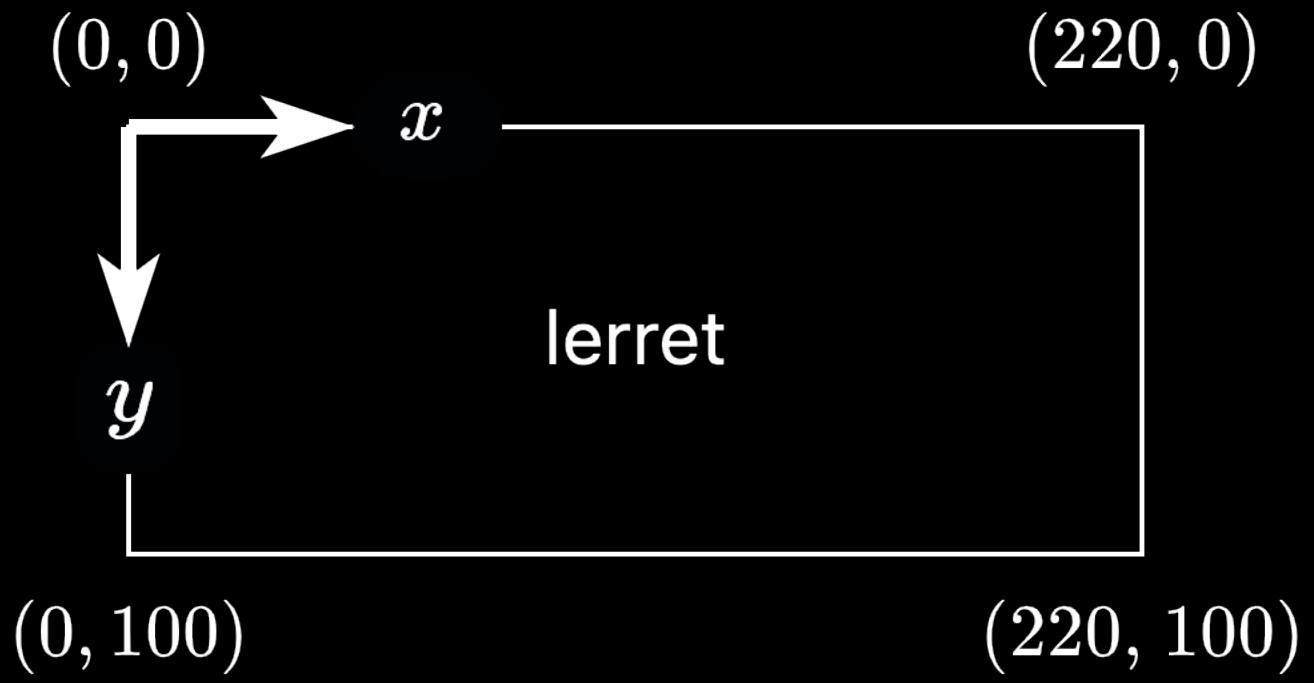
Grafikk

INF101 forelesning 10. februar 2023

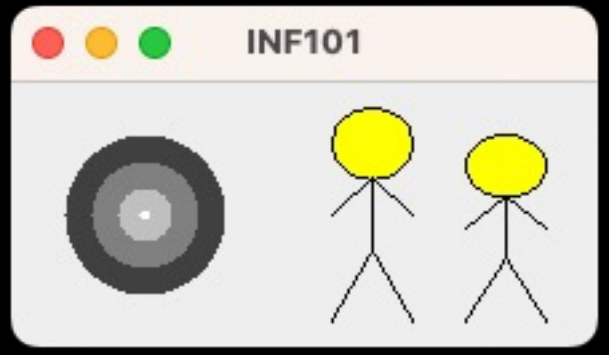
Torstein Strømme

Stikkord: arv, Object, rutenett, grafikk

Grafikk



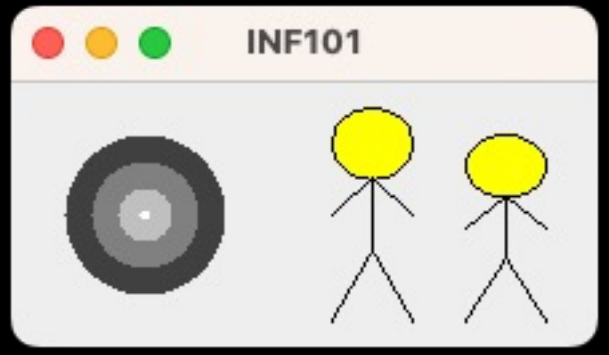
Hjelpemetoder for grafikk




```java
@Override
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  Graphics2D g2 = (Graphics2D) g;

  fillCenteredCircle(g2, 50, 50, 30, Color.DARK_GRAY);
  fillCenteredCircle(g2, 50, 50, 20, Color.GRAY);
  fillCenteredCircle(g2, 50, 50, 10, Color.LIGHT_GRAY);
  fillCenteredCircle(g2, 50, 50, 2, Color.WHITE);

  drawMan(g2, 120, 10, 30, 80);
  drawMan(g2, 170, 20, 30, 70);
}
```

```java
private static void fillCenteredCircle(Graphics2D g2, double cx, double cy, double r, Color color);

private static void drawMan(Graphics2D g2, double xLeft, double yTop, double width, double height);
```

Hjelpemetoder for grafikk



Gode parametre for hjelpefunksjoner:

Farge til figuren

Posisjon til sentrum og størrelsen til figuren

```
private static void fillCenteredCircle(Graphics2D g2, double cx, double cy, double r, Color color);
private static void drawMan(Graphics2D g2, double xLeft, double yTop, double width, double height);
```

Beskrivelse av rektangelet som omslutter figuren

Typehierarki for Shape