Interaktiv grafikk

INF101 forelesning 28. februar 2023

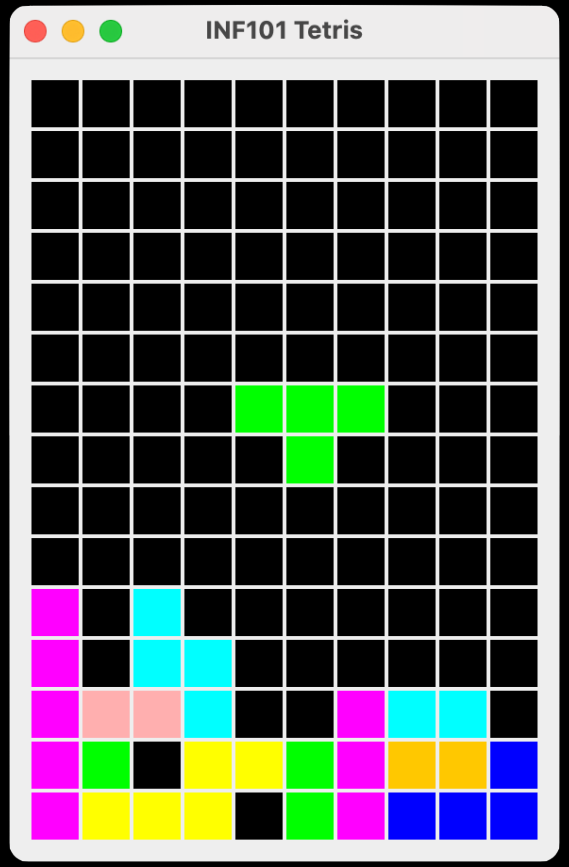
Torstein Strømme

Stikkord: model-view-controller, static, record, enum, UML

I dag

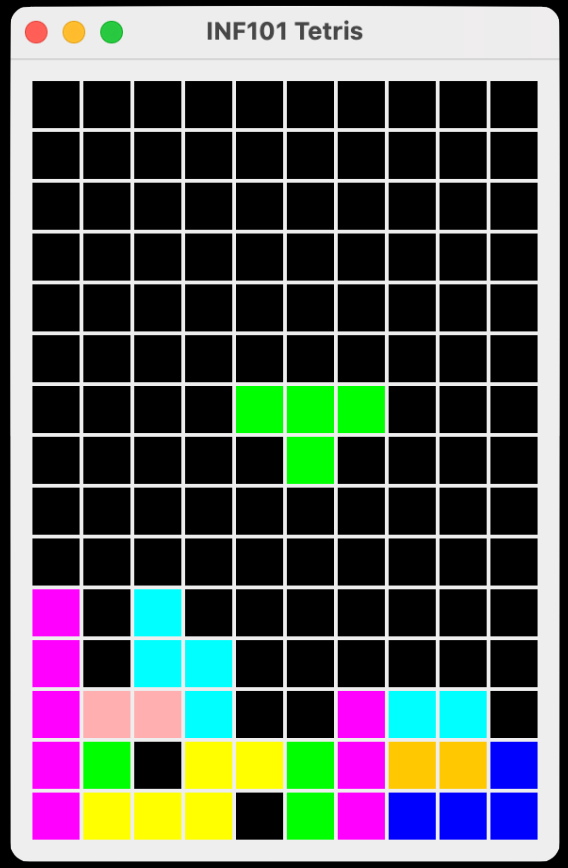
- Tetris
- Småting som kan dukke opp
  - Static
  - Record
  - Enum
- Interaktiv grafikk
  - Model-view-controller
  - Eksempler
- Hvis tid: UML

Semesteroppgave 1



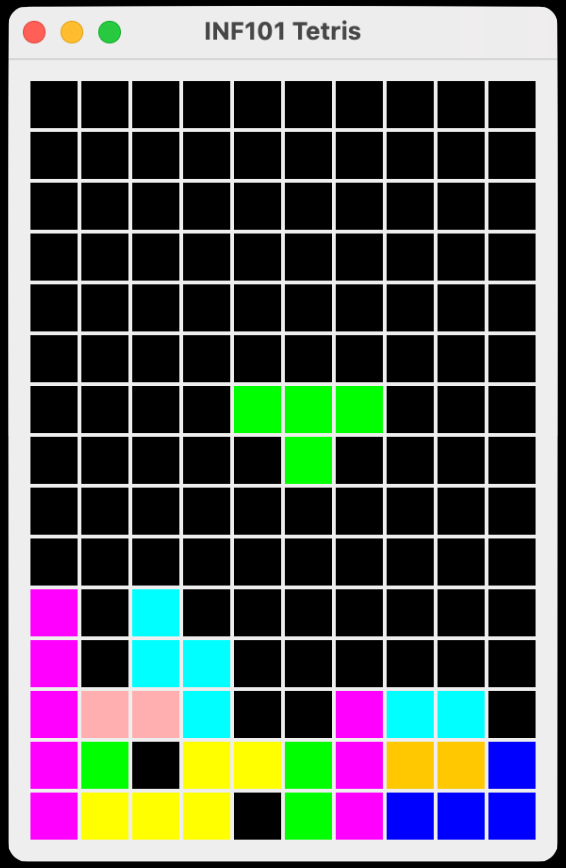
Semesteroppgave 1

- Lag Tetris fra scratch
- Følg en guide
  - Opprett en forenklet modell
  - Tegn modellen
  - Forbedre modellen
  - Tegn forbedret modell
  - Endre modellen med tastetrykk
  - Forbedre modellen
- Start tidlig!!!!!!!!!!!!



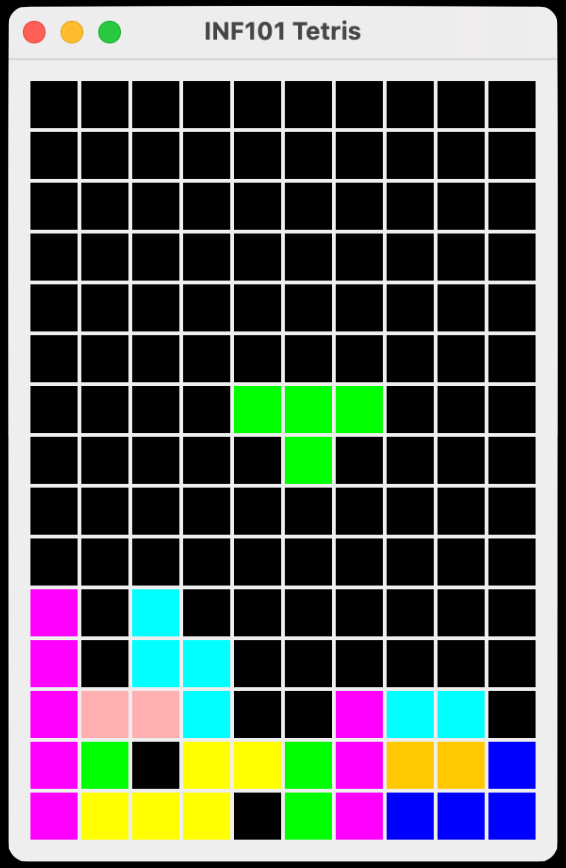
Semesteroppgave 1

- Vurdering
  - Funksjonalitet (5 poeng)
    - Timer
    - Fjerne fulle rekker
    - Dropping/liming/game over
    - Rotasjon/flytting
    - Tegning
  - Dokumentasjon (2 poeng)
    - Javadocs for metoder i grensesnitt og public metoder
    - Selv-dokumenterend metode- og variabelnavn



Semesteroppgave 1

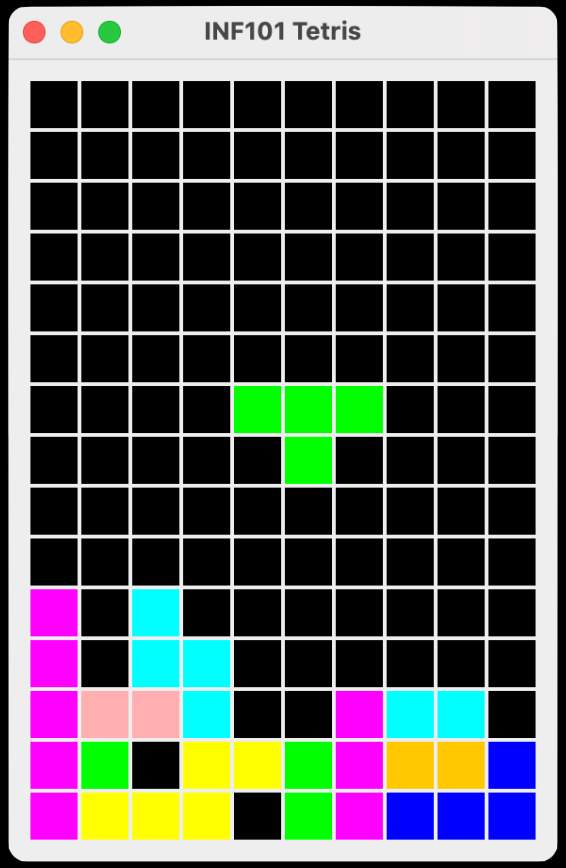
- Vurdering
  - Kodestil (2 poeng)
    - Modularitet og innkapsling
    - Fornuftig bruk av hjelpemetoder
    - God gjenbruk av kode
  - Testing (3 poeng)
    - Modellen er tilrettelagt for testing
    - Modellen blir testet
  - Besvarelser av refleksjonsspørsmål (3 poeng)



Semesteroppgave 1

- Samarbeid
  - Arbeidet er individuelt
  - Det er lov å feilsøke sammen
  - Det er lov å dele små utdrag fra koden på discord for å be om hjelp

  - Det er ikke lov å **skrive av** kode
  - Det er ikke lov å dele koden din slik at andre kan få tilgang til den uten at du er en aktiv part.
    - Du kan: dele skjerm
    - Du kan ikke: sende hele koden din til noen



static

Tilhører klassen

Klassemetoder

- Velegnet til *funksjoner*
  (metoder som ikke benytter *this*)

```
Klasse.metodeNavn();
```

Klassevariabler

- Globale variabler, tilgjengelig for “alle”
- Velegnet til konstanter
- Bør ellers unngås
- Vanskelig å teste med

uten static

Tilhører objekter

Instansmetoder

- Kan se og endre instansvariabler
  (har alltid tilgang til objektet *this*)

```
objekt.metodeNavn();
```

Instansvariabler

- Variabler som er unike for hvert objekt
- Velegnet for å holde på tilstander/state som logisk hører sammen.

Record

- En klasse som fungerer som en slags tuple
- Alle feltvariabler er final
- Konstruktør, gettere, toString, equals og hashCode er med automatisk

```
public record CellPosition(int row, int col) { }
```

```
CellPosition pos = new CellPosition(3, 5);
int a = pos.row();
int b = pos.col();
pos.row = 4; // ikke lov, row er final (og private)
```

Record

```java
public record CellPosition(int row, int col) { }
```

omtrent det samme som

```java
public final class CellPosition {

  private final int row;
  private final int col;

  public CellPosition(int row, int col) {
    this.row = row;
    this.col = col;
  }

  public int row() {
    return this.row;
  }

  public int col() {
    return this.col;
  }

  @Override
  public boolean equals(Object o) {
    // ... blah blah blah
  }

  @Override
  public int hashCode() {
    // ... blah blah blah
  }

  @Override
  public String toString() {
    // ... blah blah blah
  }
}
```

Enum

- En klasse med et fast definer sett av objekter
- Umulig å lage nye objekter run-time

```
public enum GameState {
  RUNNING(), PAUSED(), GAME_OVER();
}
```

```
GameState state = GameState.RUNNING;
state = GameState.PAUSED; // OK, GameState.PAUSED er en GameState
state = "PAUSED";         // ikke lov, "PAUSED" er ikke en GameState
state = new GameState();  // ikke lov, GameState er en enum
```

Enum

```
public enum GameState {
  RUNNING(), PAUSED(), GAME_OVER();
}
```

omtrent det samme som

```
public final class GameState {
  public static final GameState RUNNING = new GameState();
  public static final GameState PAUSED = new GameState();
  public static final GameState GAME_OVER = new GameState();

  private MyGameState() {}
}
```

super-duper-

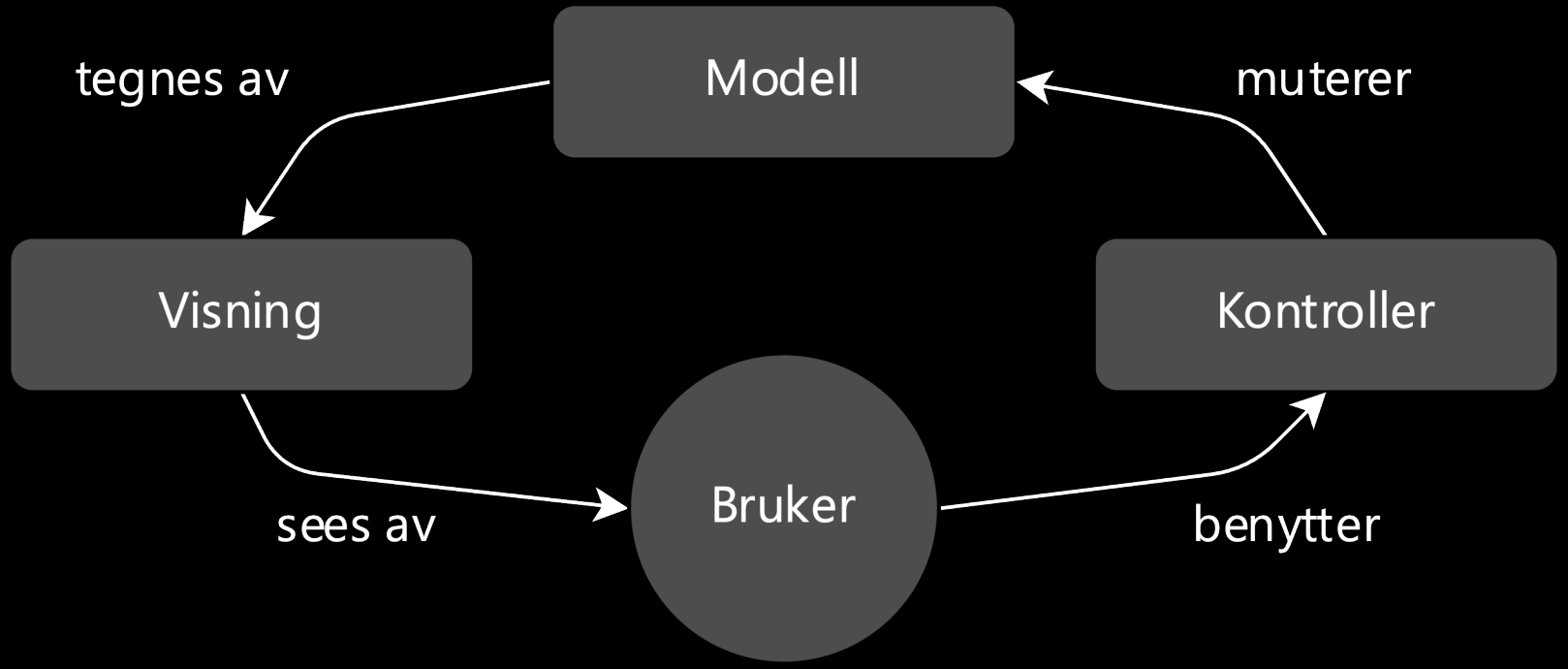
Enum og record er *klasser*

- Både enum og record er «vanlige» klasser med ekstra begrensninger
  - Record: alle instansvariabler er final og har gettere
  - Enum: konstruktøren er superhemmelig, og objektene er konstanter (public final klassevariabler)

- Er forvøvrig som helt vanlige klasser:
  - Definerer en type
  - Kan implementere grensesnitt
  - Kan ha både instansmetoder, konstruktører, klassemetoder og klassevariabler
  - Enum kan ha vanlgie instansvariabler
  - Record-objekter kan opprettes som vanlige objekter

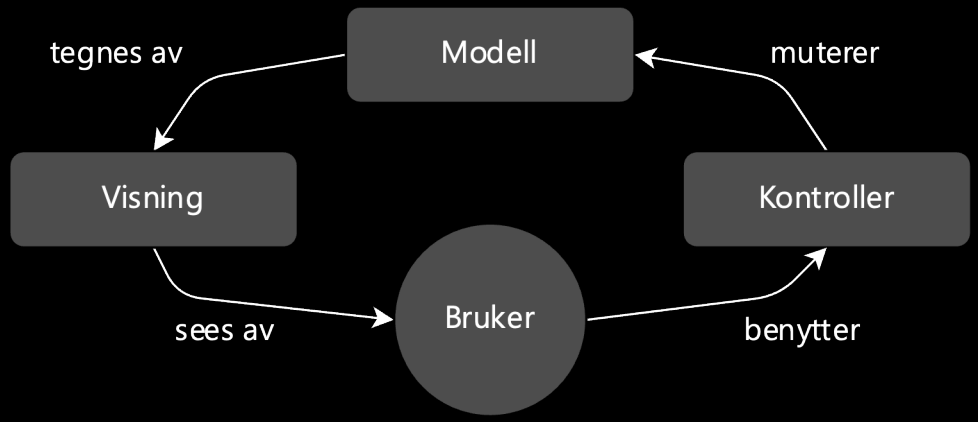
Interaktiv grafikk

Model-View-Controller



Model-view-controller

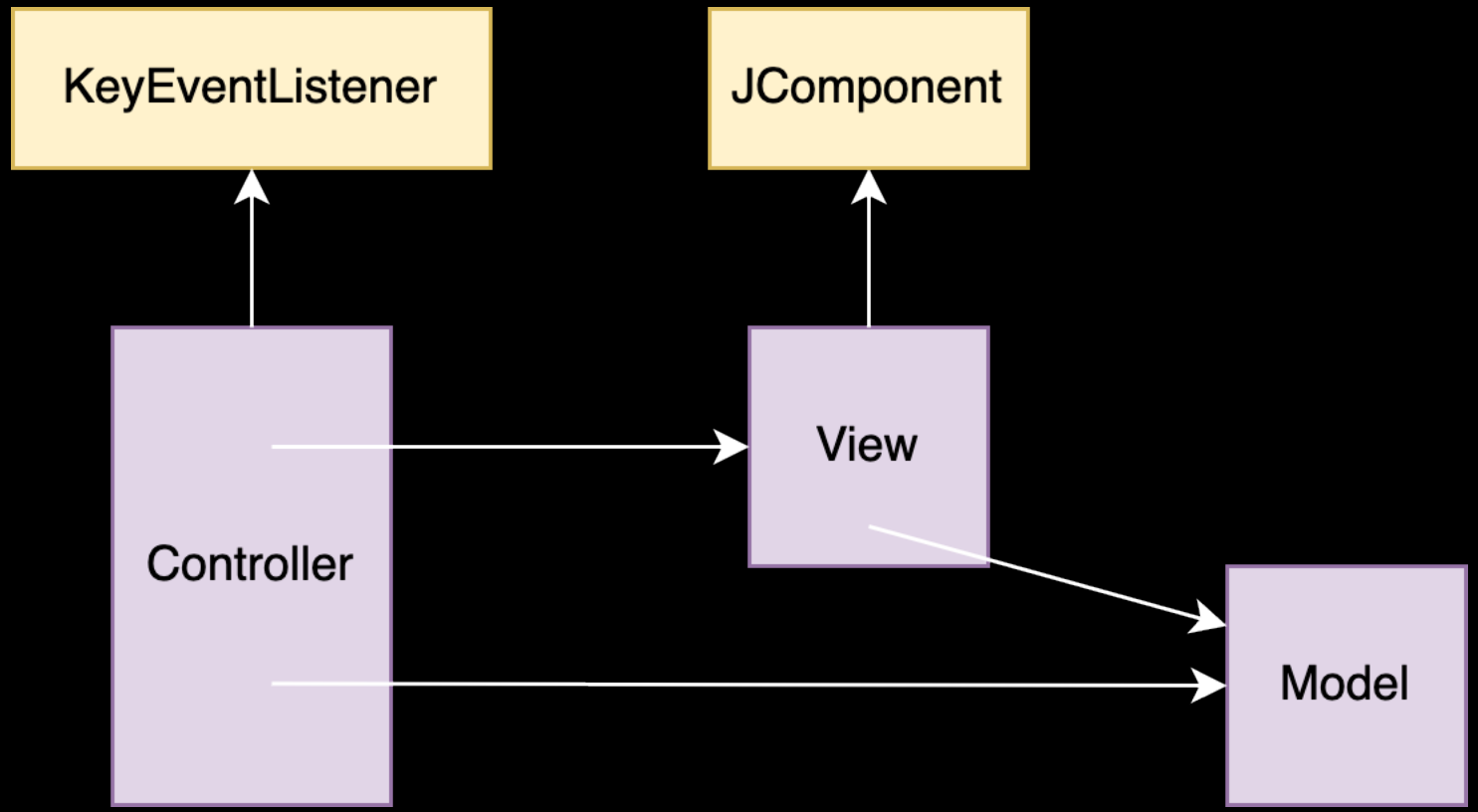
- Prinsipp: lag et tydelig skille mellom
  - modellen som holder tilstand
  - kode som tegner modellen
  - kode som modifiserer modellen



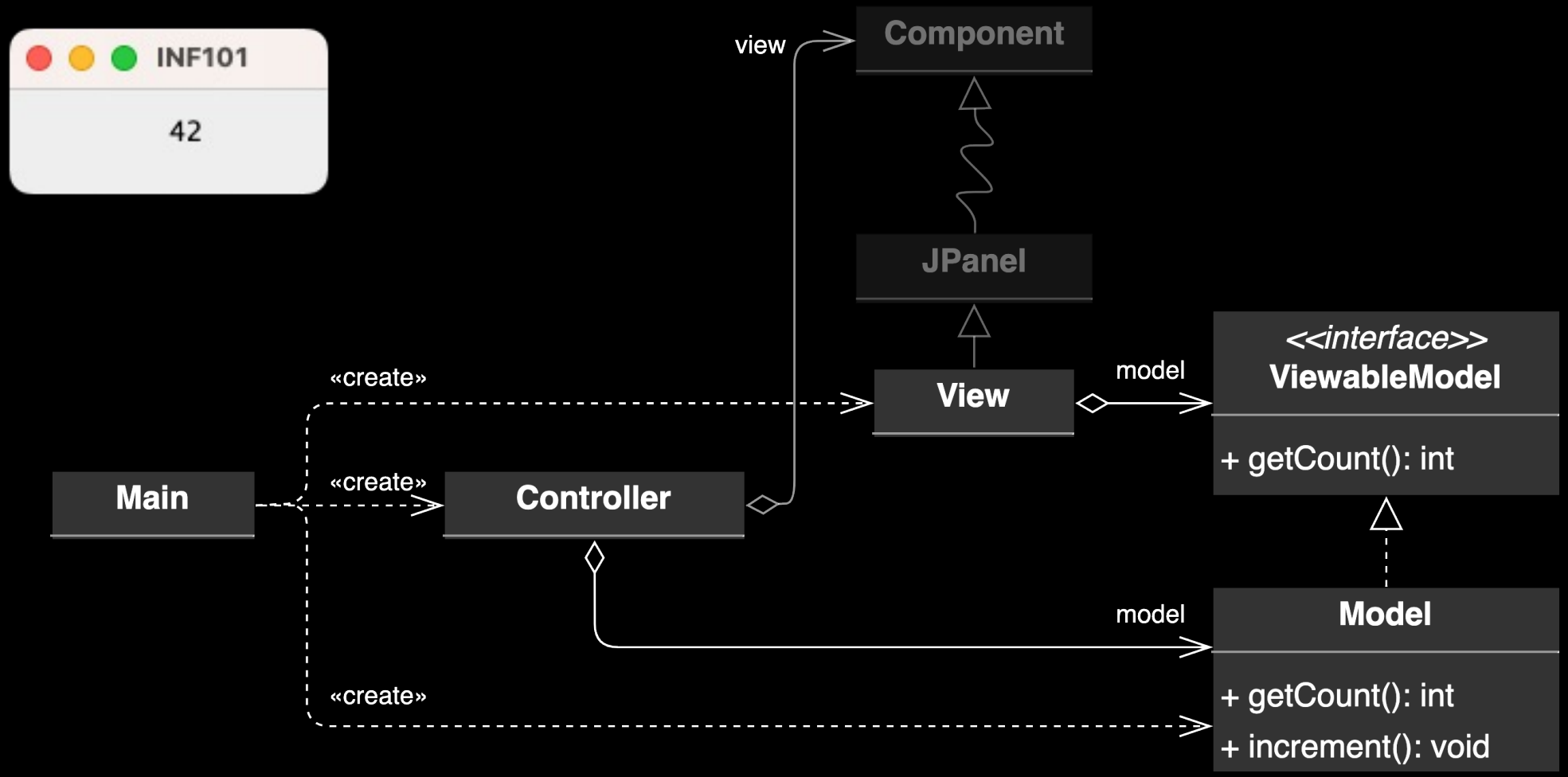
Ikke modifiser modellen når du tegner den!

- For å hjelpe oss å overholde dette, kan vi la visningskode se på modellen via et «read-only» grensesnitt.

Model-view-controller med tastetrykk i swing



Første eksempel: en teller



Første eksempel: en teller

1. Opprett en klasse *View* og en klasse *Main* og tegn din favoritt-figur
   - Følg mal fra kursnotatene om grafikk

2. Opprett en klasse *Model*. La den ha en instansvariabel som skal huske hvor mange ganger noen har trykket på tastaturet

3. Opprett et grensesnitt *ViewableModel* og la grensesnittet ha en metode for å returnere hvor mange ganger noen har trykket. La Model implementere grensesnittet

Første eksempel: en teller

4. Legg til en metode *increment* i Model som øker telleren med én.

5. La *View* ha en instansvariabel av typen *ViewableModel*, og initier den med en verdi gitt som argument til konstruktøren.

6. Tegn verdien i paintComponent

Første eksempel: en teller

7. Opprett en klasse *Controller* med to instansvariabler initert fra parametre i konstruktøren: en *Model*-parameter for modellen og en *Component*-parameter (importeres fra java.awt) for den visuelle komponenten hvor vi skal lytte til tastetrykk vi kaller *view*.

8. La *Controller* implementere *KeyListener* (importes fra java.awt.event), og legg disse linjene til i konstruktøren:

```
view.setFocusable(true);
view.addKeyListener(this);
```

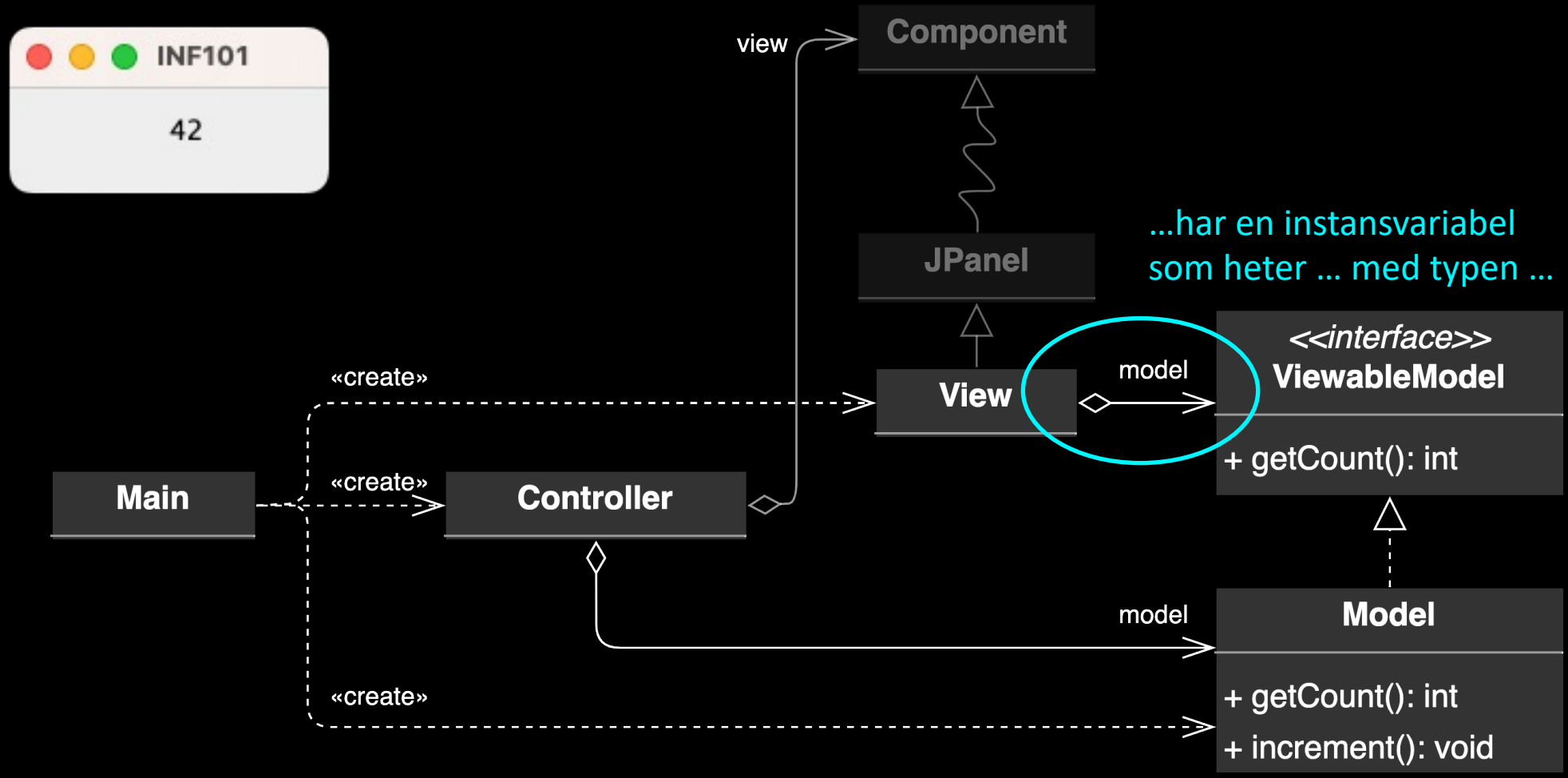
Første eksempel: en teller

9. Kall *increment* på modellen i *keyReleased*-metoden i *Controller*

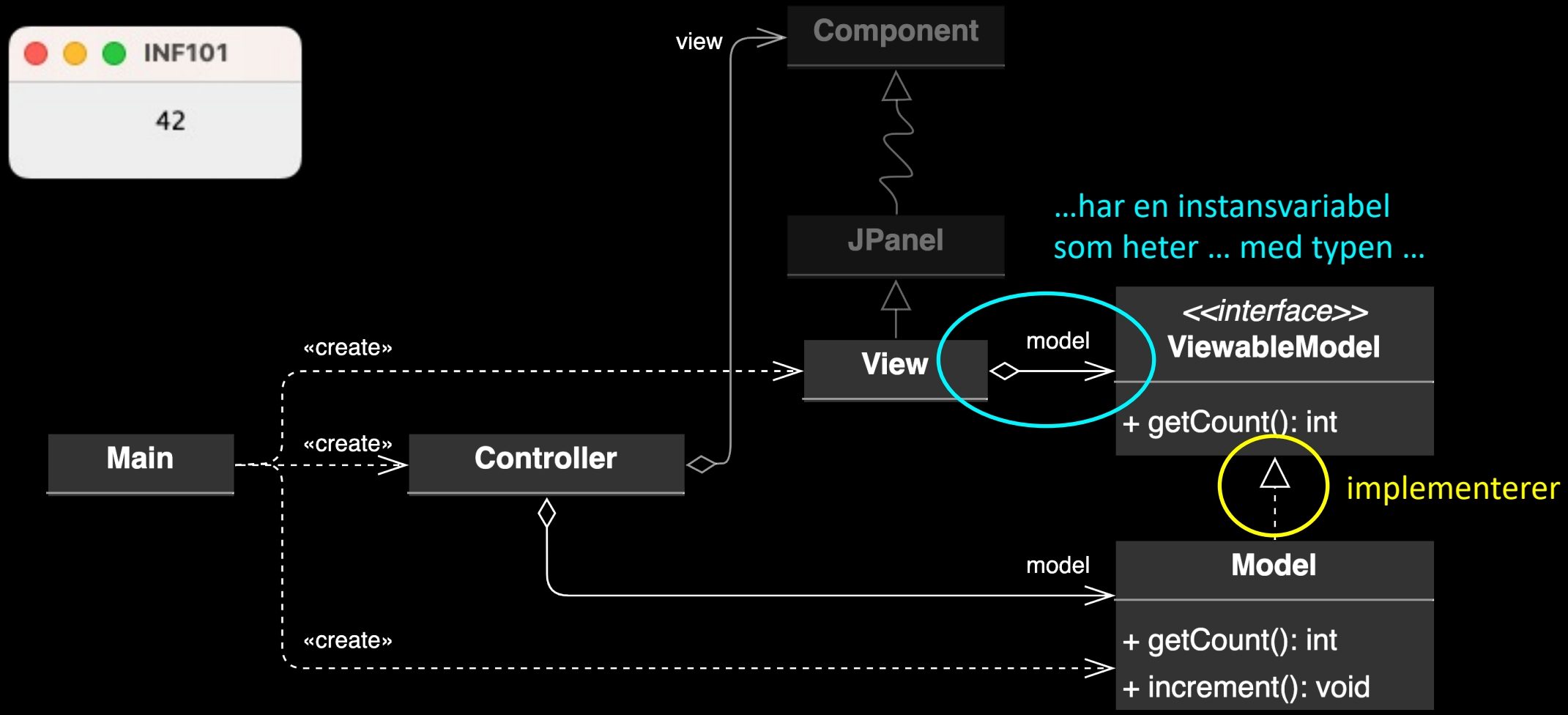
10. Kall *repaint* på visningen i *keyReleased*-metoden i *Controller*

11. Vinn!

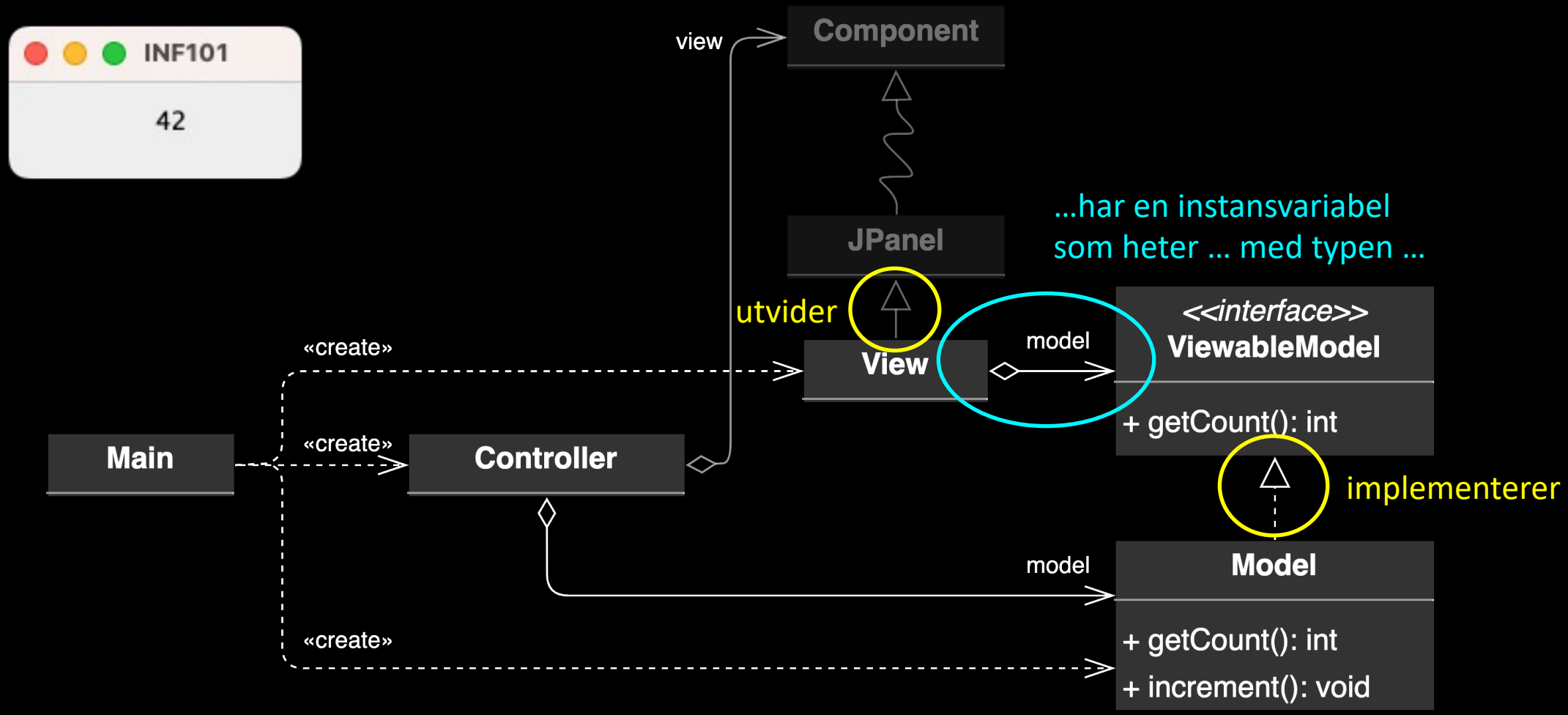
Første eksempel: UML-diagram



Første eksempel: UML-diagram



Første eksempel: UML-diagram



Første eksempel: UML-diagram