



IN1010 uke 13

Gruppe 7

Agenda

- Design og testbarhet
 - Hva man skal tenke på
 - Bruke interfaces
- Å skrive tester
 - Arrange, act, assert
 - Kodeeksempel: enhetstest
- Jobbe med oppgaver/trix/oblig!

Design og testbarhet



Tenke på når man skriver kode

- Oftest skal kode leses/endres masse av deg/andre, ikke bare noe man skriver 1 gang og aldri ser på igjen
- For å lett kunne lese og endre:
 - Leselig (kommentarer, variabelnavn, ...)
 - God design/struktur
- Hvis ikke: Spaghettikode :(
 - Vanskelig å følge,
for mange deler er viklet inn i hverandre



SOLID-prinsippene

Viktigst i IN1010 / hovedpoenget:

- Hver klasse skal ha **1** ansvarsområde
 - Skal være tydelig hva ansvaret er
- Ting bør avhenge av abstraksjoner (interface) og ikke konkrete implementasjoner (eksempel snart)

Hvorfor bruke interfaces

```
class SpillKontroller {  
    SpillTerminalView view;  
    SpillModell modell;  
  
    public SpillKontroller(SpillTerminalView view) {  
        this.view = view;  
        this.modell = new SpillModell();  
    }  
}
```

Hvis vi vil at Spillet skal kunne vises i GUI istedenfor....?

```
class SpillKontroller {
    SpillView view;
    SpillModell modell;

    public SpillKontroller(SpillView view) {
        this.view = view;
        this.modell = new SpillModell();
    }
}

interface SpillView {
    /* metoder som alle SpillViews skal tilby */
}
```

Hvorfor bruke interfaces

- Bruker `SpillView interface` istedenfor den direkte `klassen` (implementasjonen)
 - Enkelt bytte ut view til annen klasse
 - Gjenbruk av koden, spill for både GUI og terminal
 - Veldig tydelig hva en `SpillView`-klasse skal gjøre

Å skrive tester



Enhetstesting

- Tester bare én del/klasse
 - Skal være uavhengig av resten av programmet
-
- Sikre at delen/klassen fungerer som den skal
 - Bruke testen senere for å se at delen fortsatt fungerer etter endringer i programmet

Kodeeksempel

*Enhetstest for
Katt-klasse*



Jobb med oppgaver!

IN1010 Emnesiden → Grupper →
Gruppe 7 → Uke13

