IN1010

Repetisjon Uke 15

GUI 2

(Basert på Eyvind Axelsens presentasjon)

9. mai, 2025

I dag

- Litt om design
- Layout i Java Swing/AWT
- Et populært designmønster for GUI:

  Model-View-Controller

Litt om design

- Det er mange ting å tenke på når vi skal lage et brukergrensesnitt. Vi vil gjerne at det skal se fint ut.
- Men vel så viktig så må det være *enkelt*, *forståelig* og *brukbart*
- Når vi lager grensesnitt for andre, så må vi også tenke på *tilgjengelighet*, alle skal helst kunne bruke det
- Noen ser dårlig, hører dårlig, er fargeblinde, vanskelig for å bevege seg, og noen har knapt brukt en datamaskin før

Litt om design

Det finnes nå etterhvert mange verktøy for å planlegge utseende

Men det viktigste er å tenke gjennom hva programmet skal gjøre

Hvilke funksjoner, hva slags struktur, hvordan skal det flyte

Når du vet hva programmet skal gjøre er det smart å lage enkle tegninger og figurer som forklarer hvordan det skal se ut

Layout og utseende

Hvis du ikke vil teste programmet på alle forskjellige type operativsystemer,

så kan du be om at Java prøver å lage det så likt som mulig

Layout og utseende

Vi har flere måter i Java for å bestemme hvordan vi plasserer komponentene i GUI'et vårt

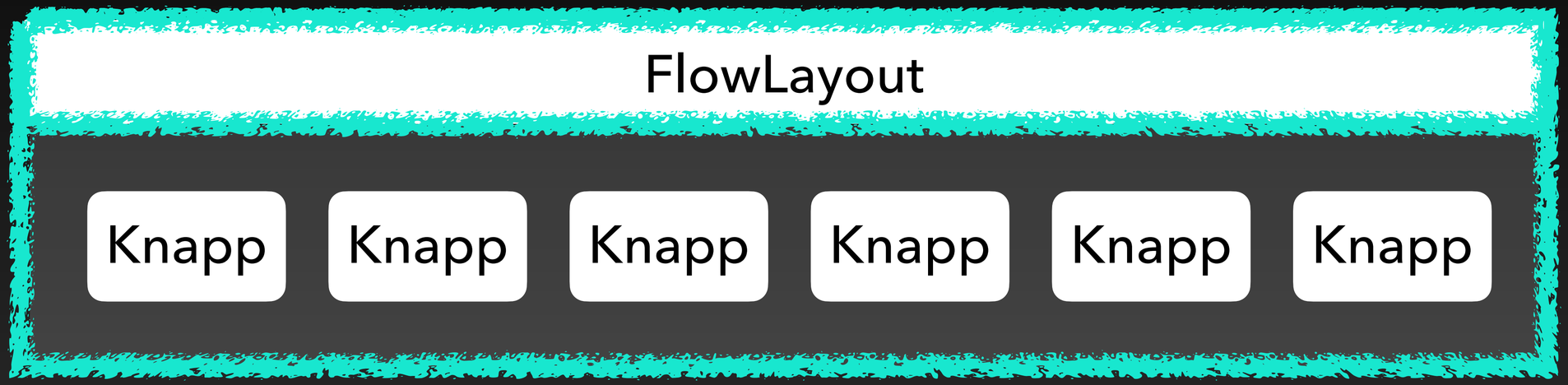
- FlowLayout
- BorderLayout
- GridLayout

I tillegg har vi et par andre som vi ikke går gjennom

CardLayout, GridBagLayout, BoxLayout, GroupLayout, SpringLayout, OverLayout

og i tillegg "null"-layout – hvor alt plasseres helt manuelt

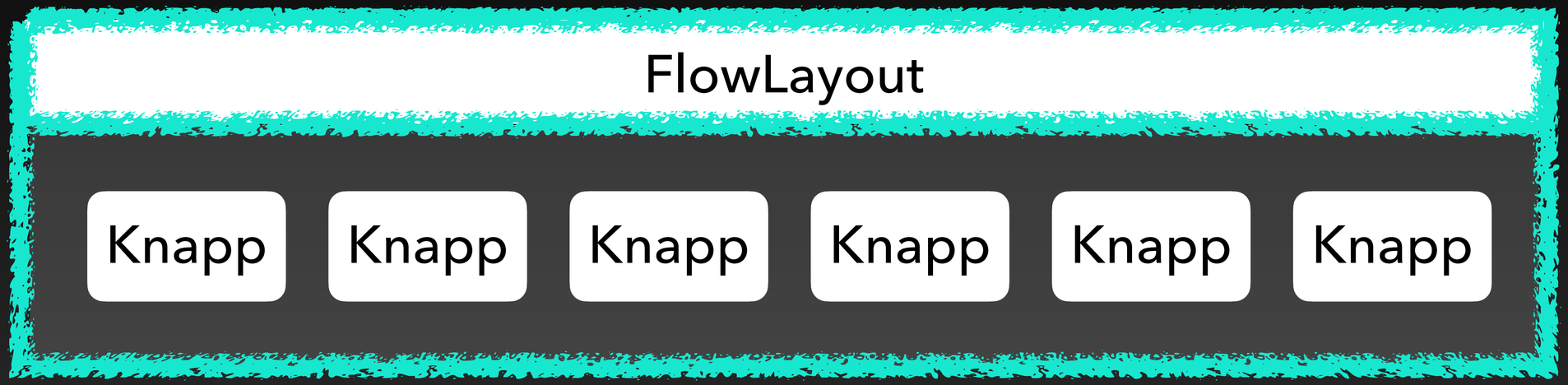
Layout og utseende



Med FlowLayout, som er det som er standard hvis du ikke bruker noe annet i Java, lar komponentene "flyte" fra venstre til høyre og deretter ovenfra og ned, etterhvert som de blir lagt til

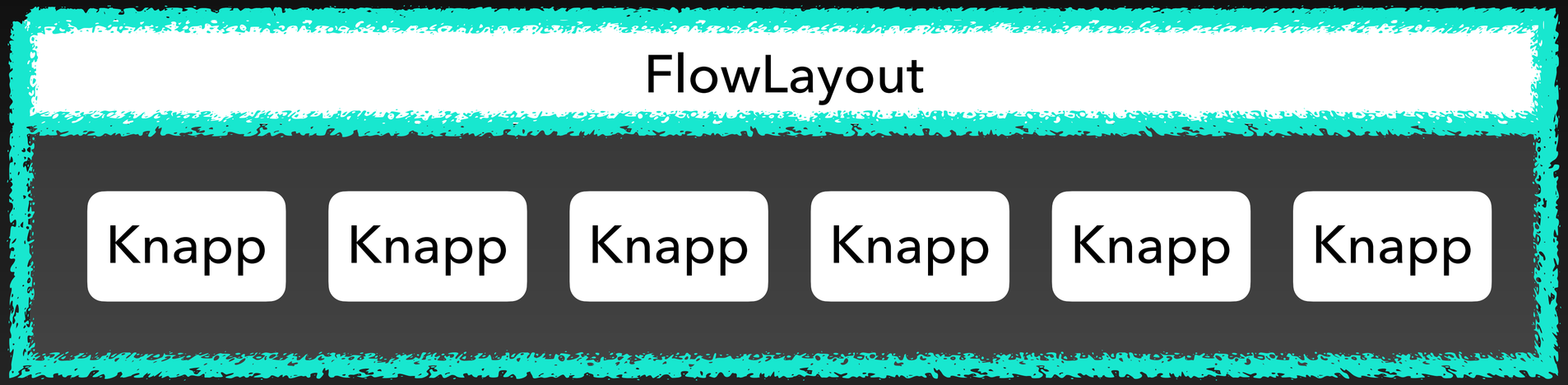
Hvis vi endrer størrelsen på vinduet så hopper de rundt, og Java prøver så godt som mulig å plassere de slik at de passer

Layout og utseende



Vi kan bruke FlowLayout med en del innstillinger

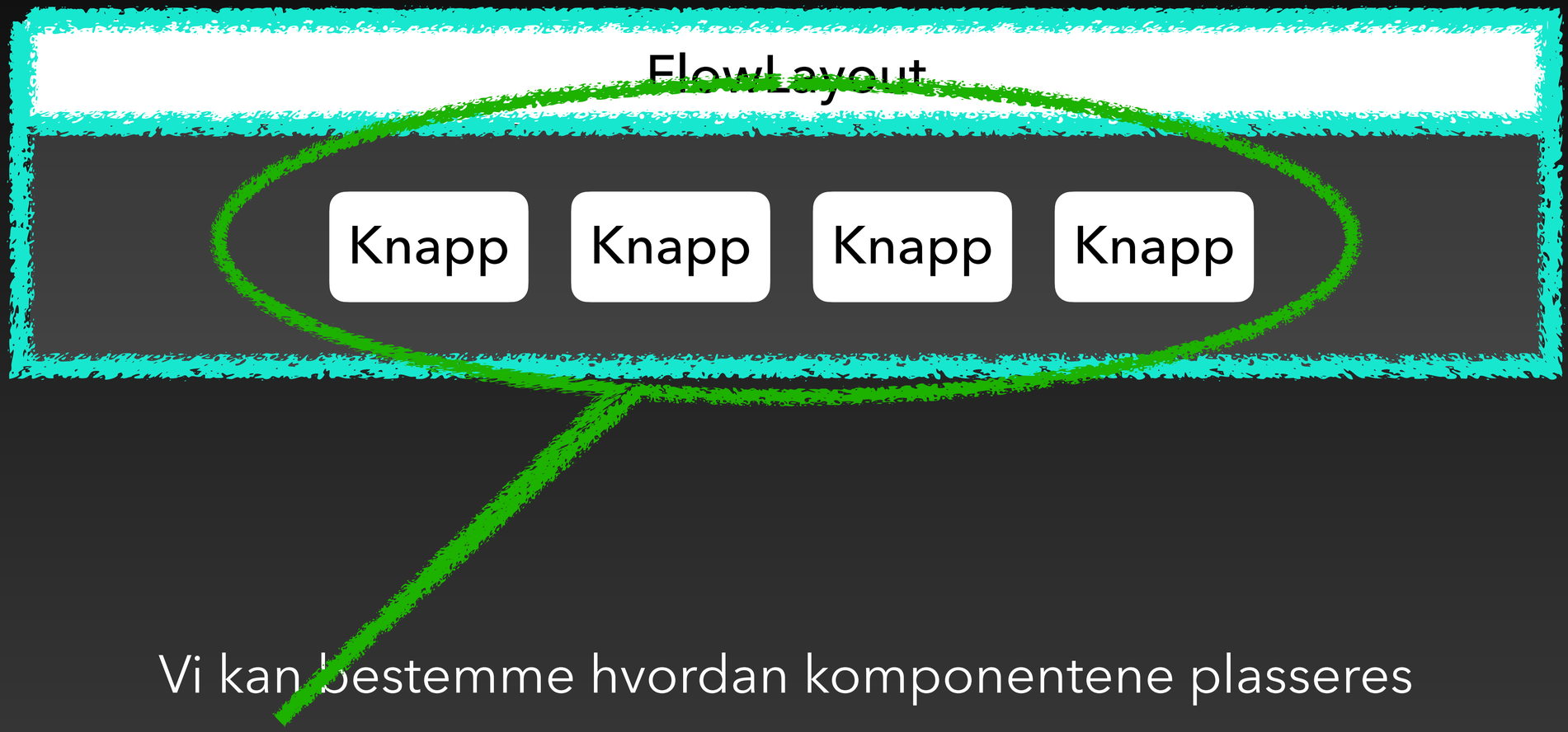
Layout og utseende



Vi kan bestemme hvordan komponentene plasseres

CENTER – midtstilt, LEFT – venstrejustert, RIGHT – høyrejustert

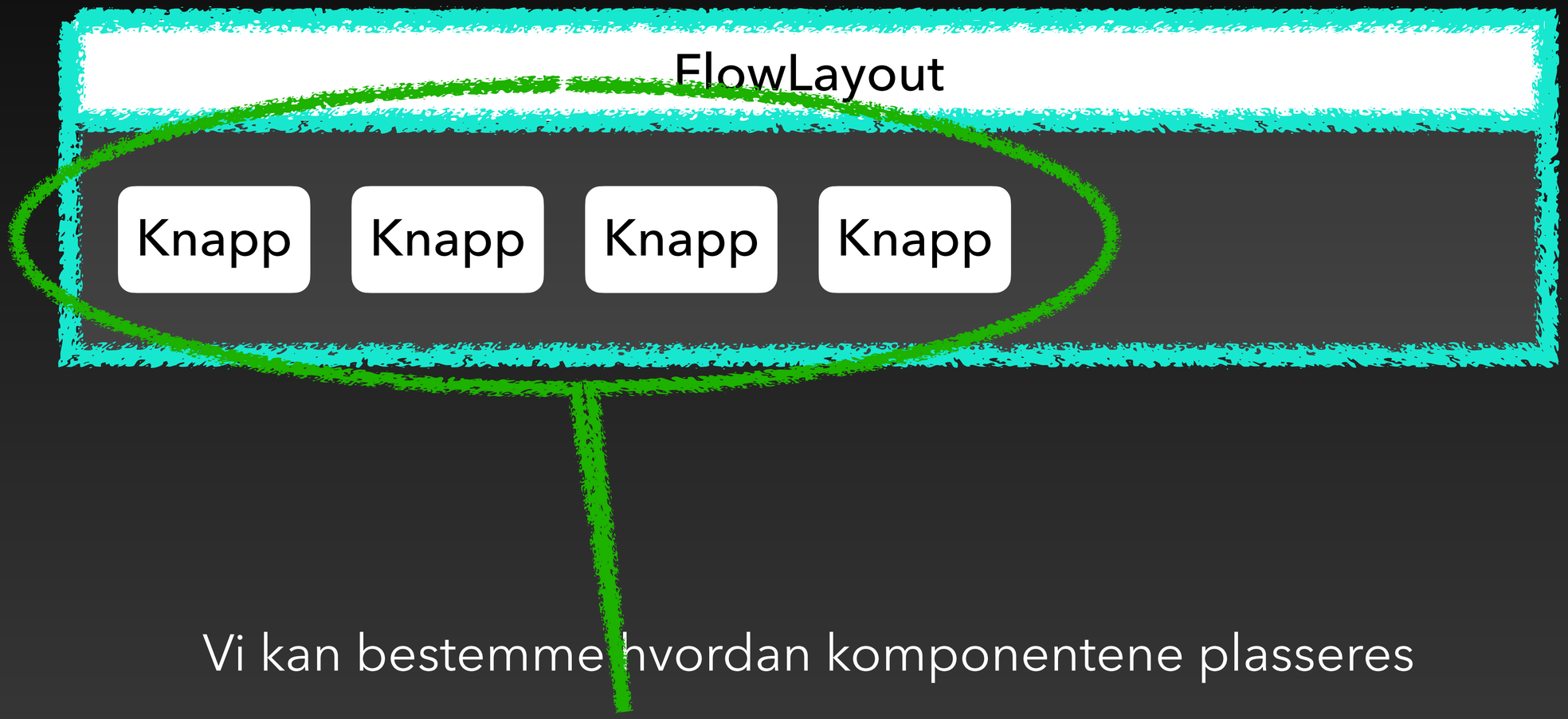
Layout og utseende



Vi kan bestemme hvordan komponentene plasseres

CENTER – midtstilt, LEFT – venstrejustert, RIGHT – høyrejustert

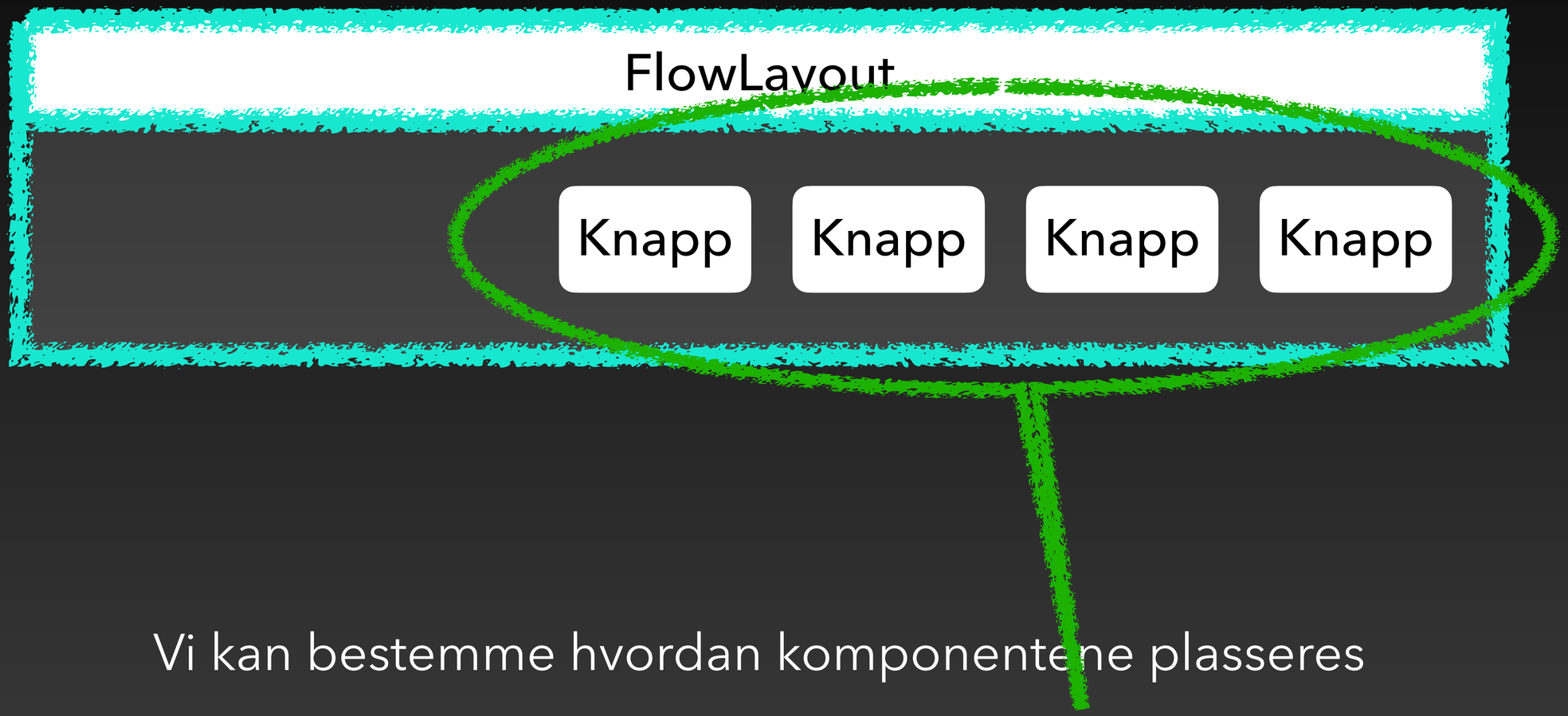
Layout og utseende



Vi kan bestemme hvordan komponentene plasseres

CENTER – midtstilt, LEFT – venstrejustert, RIGHT – høyrejustert

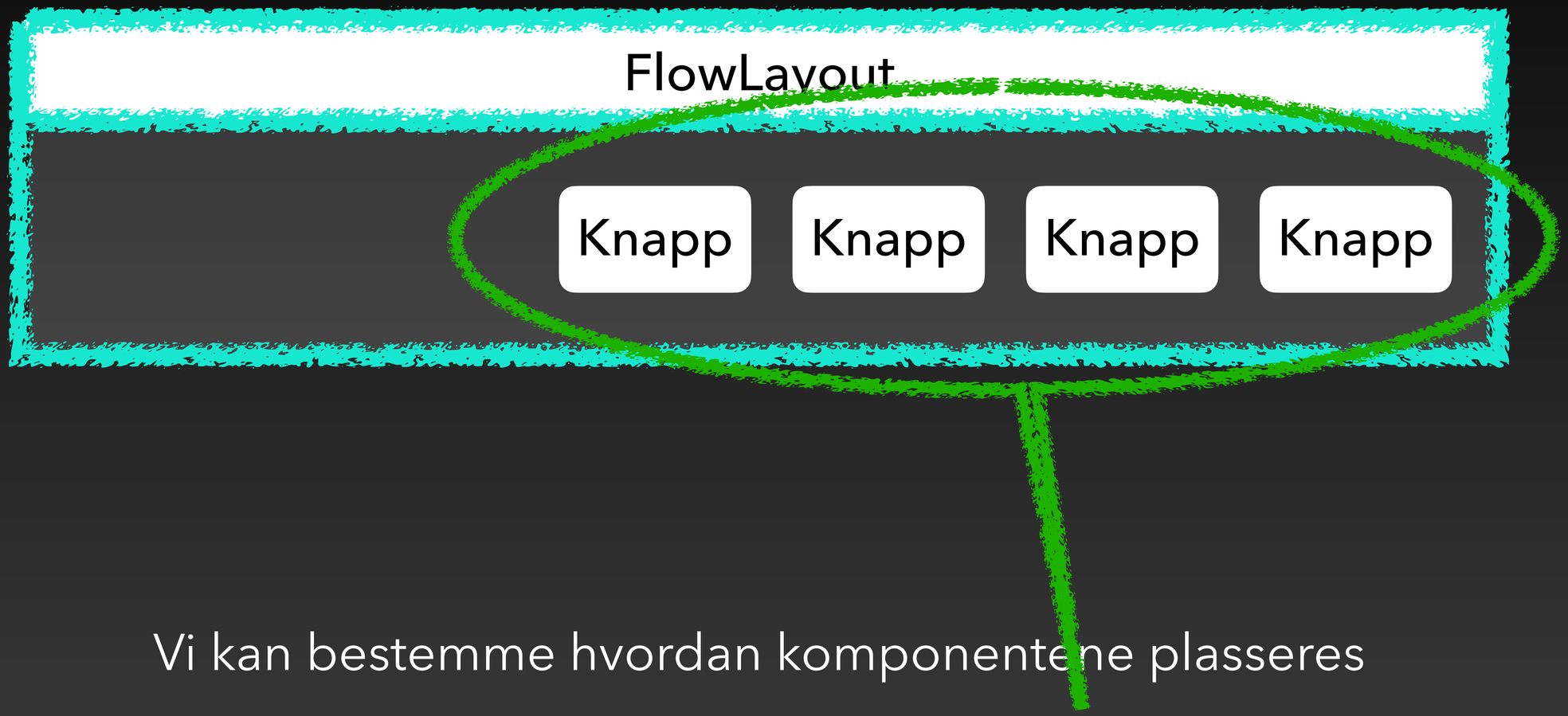
Layout og utseende



Vi kan bestemme hvordan komponentene plasseres

CENTER – midtstilt, LEFT – venstrejustert, RIGHT – høyrejustert

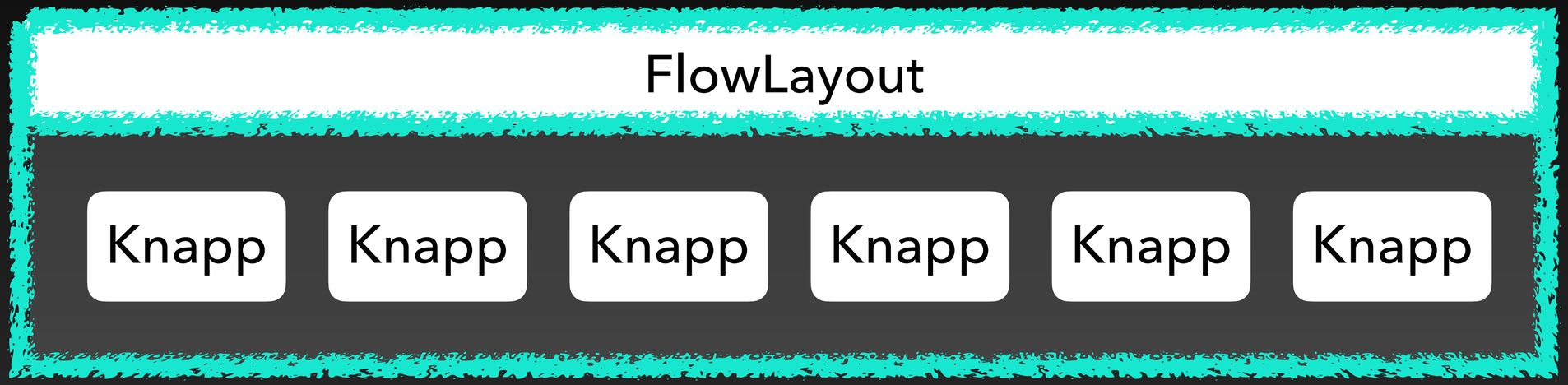
Layout og utseende



Vi kan bestemme hvordan komponentene plasseres

CENTER – midtstilt, LEFT – venstrejustert, RIGHT – høyrejustert

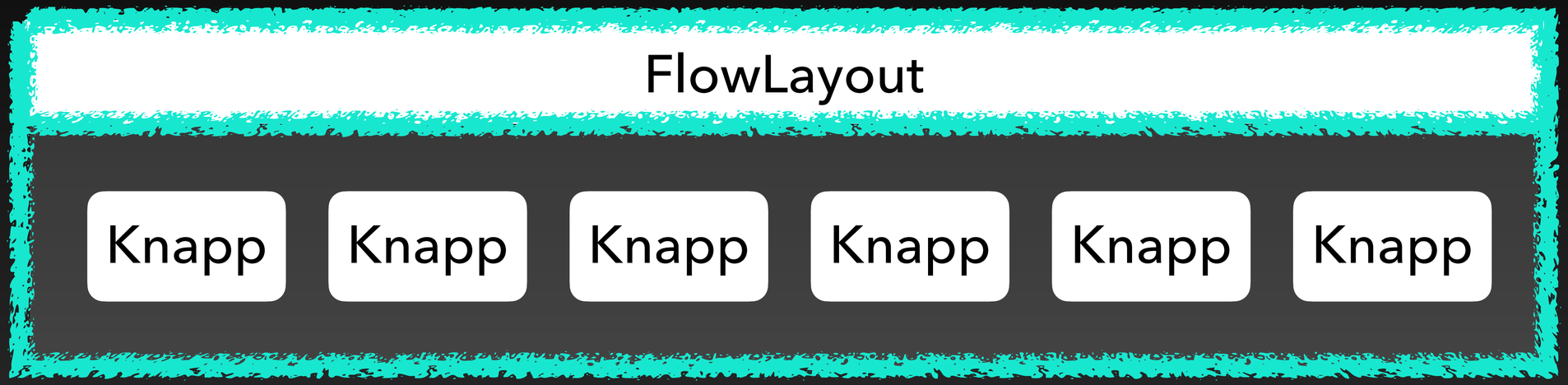
Layout og utseende



Vi kan spesifisere avstanden mellom komponentene,

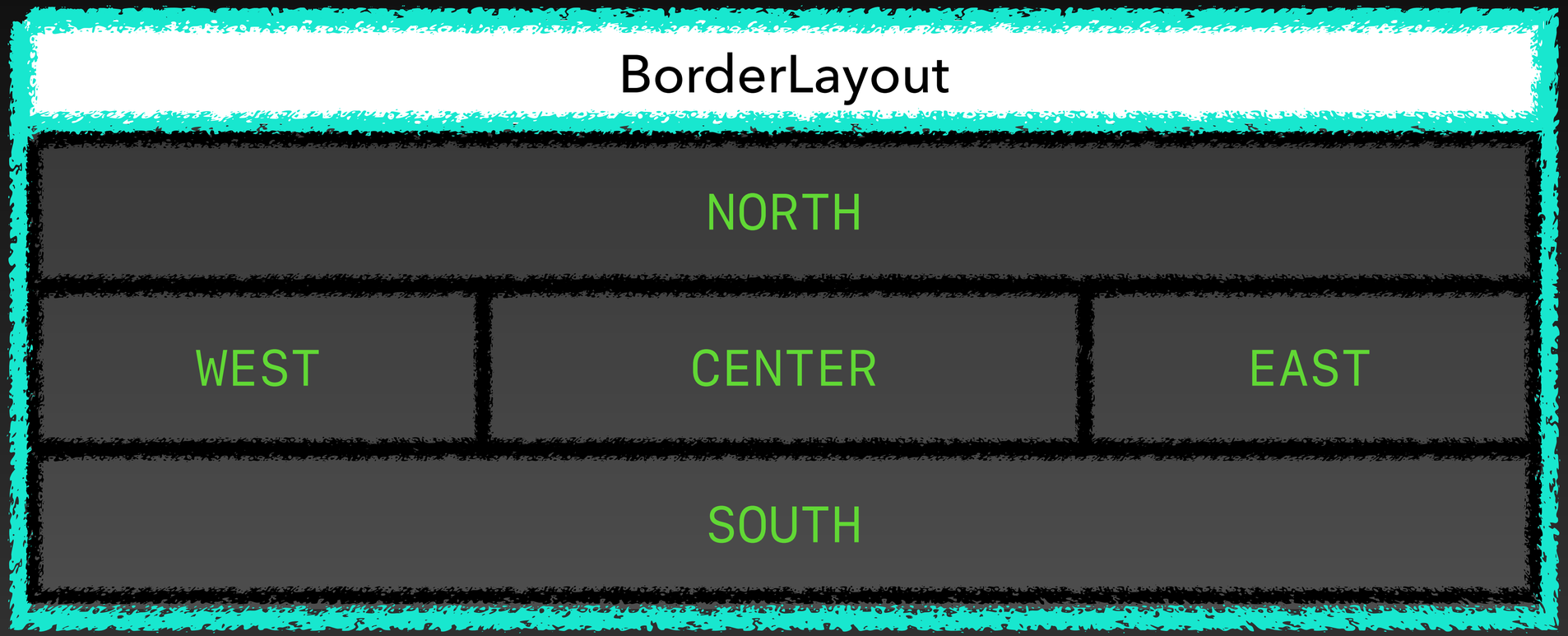
den første angir antall punkter horisontalt, og den andre er vertikalt

Layout og utseende



Det finnes også de relative LEADING og TRAILING, som justerer avhengig av orienteringen komponenten har (ComponentOrientation)

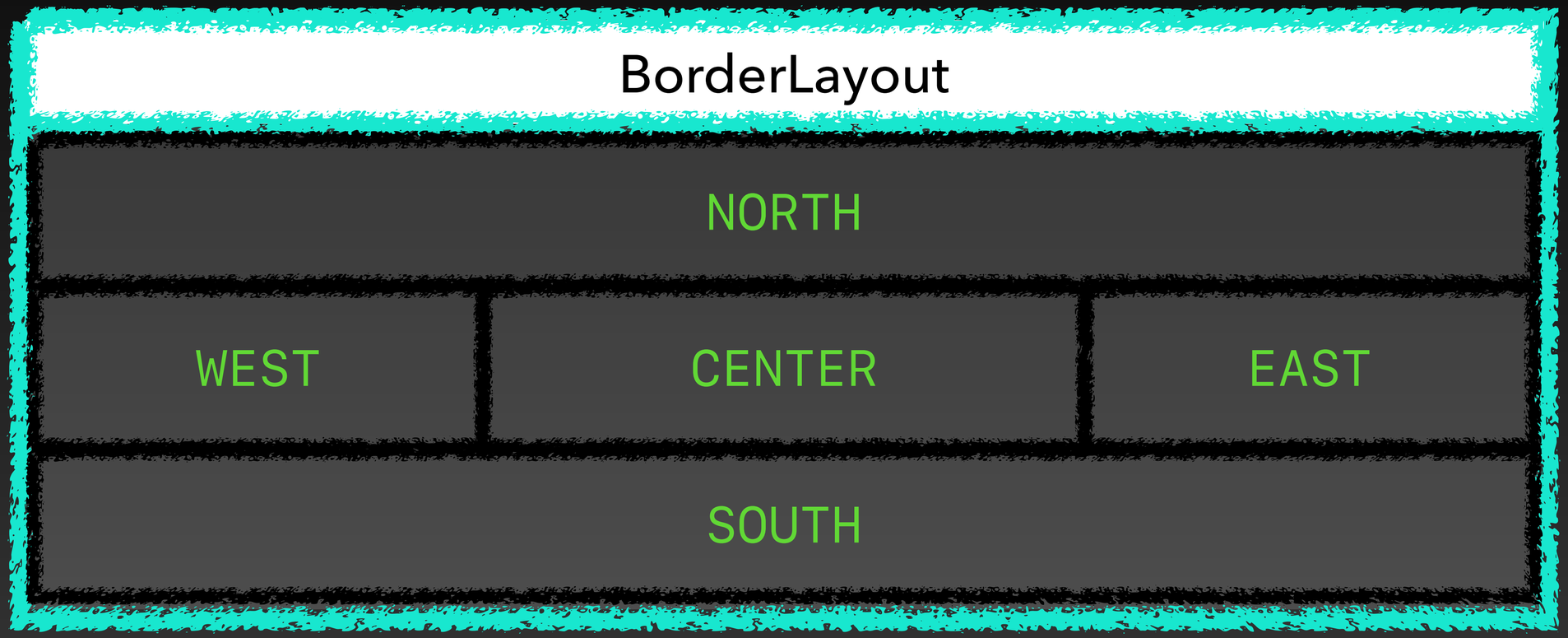
Layout og utseende



Når vi vet omtrent hvordan vi vil plassere komponentene, så kan vi bruke BorderLayout

Det som er viktig å huske på er at kun kan ha en komponent per "himmelretning"

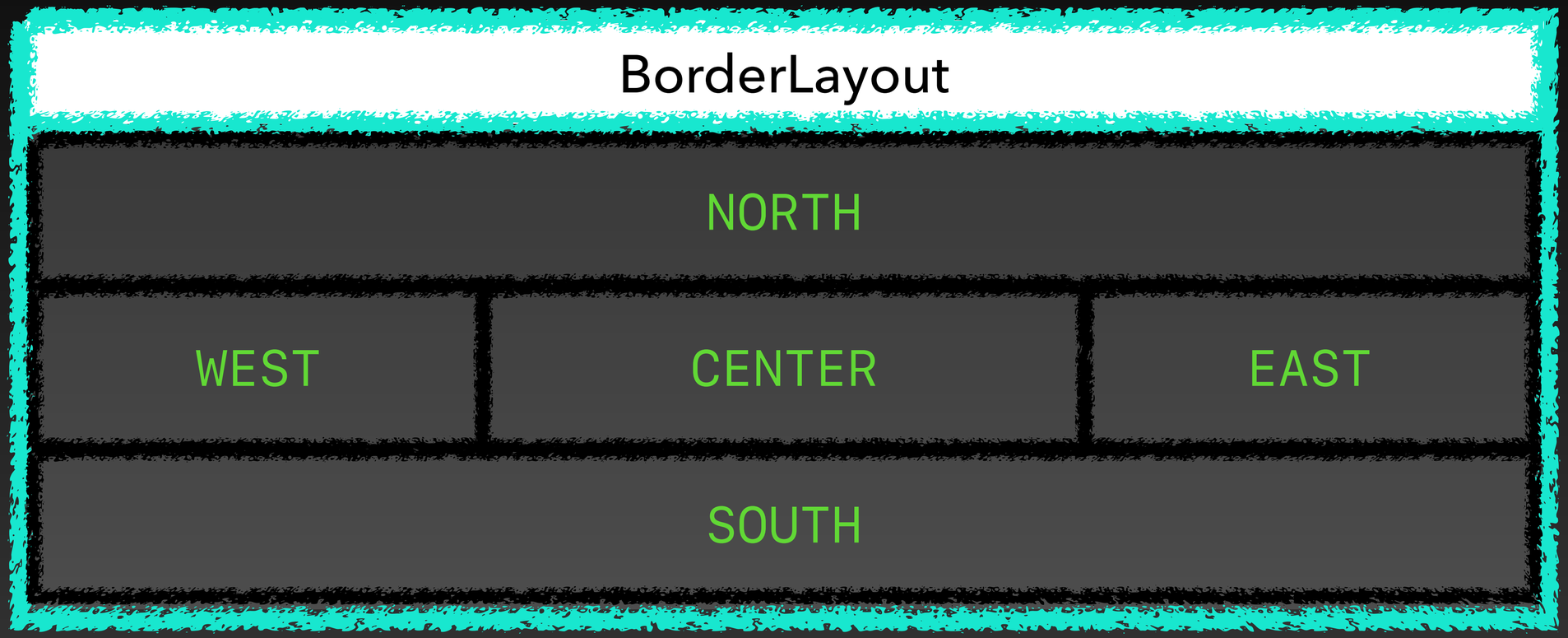
Layout og utseende



Men vi kan bruke et panel, som kan ha flere komponenter med en annen layout

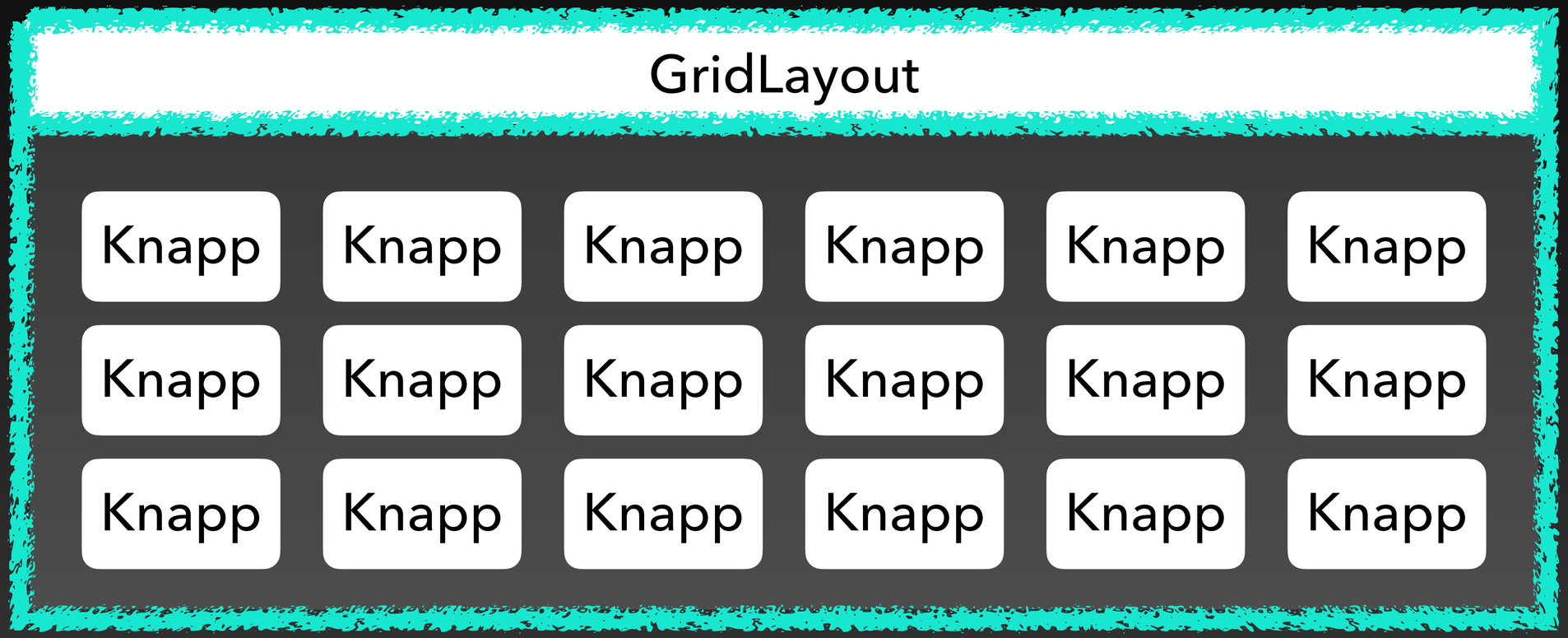
I de fleste tilfellene så kan vi nøste layout, for å lage ganske avansert GUI

Layout og utseende



Vi må ikke bruke alle. De ulike komponentene vil prøve å fylle opp plassen, nord/sør horisontalt, vest/øst vertikalt, mens senter fyller opp resten

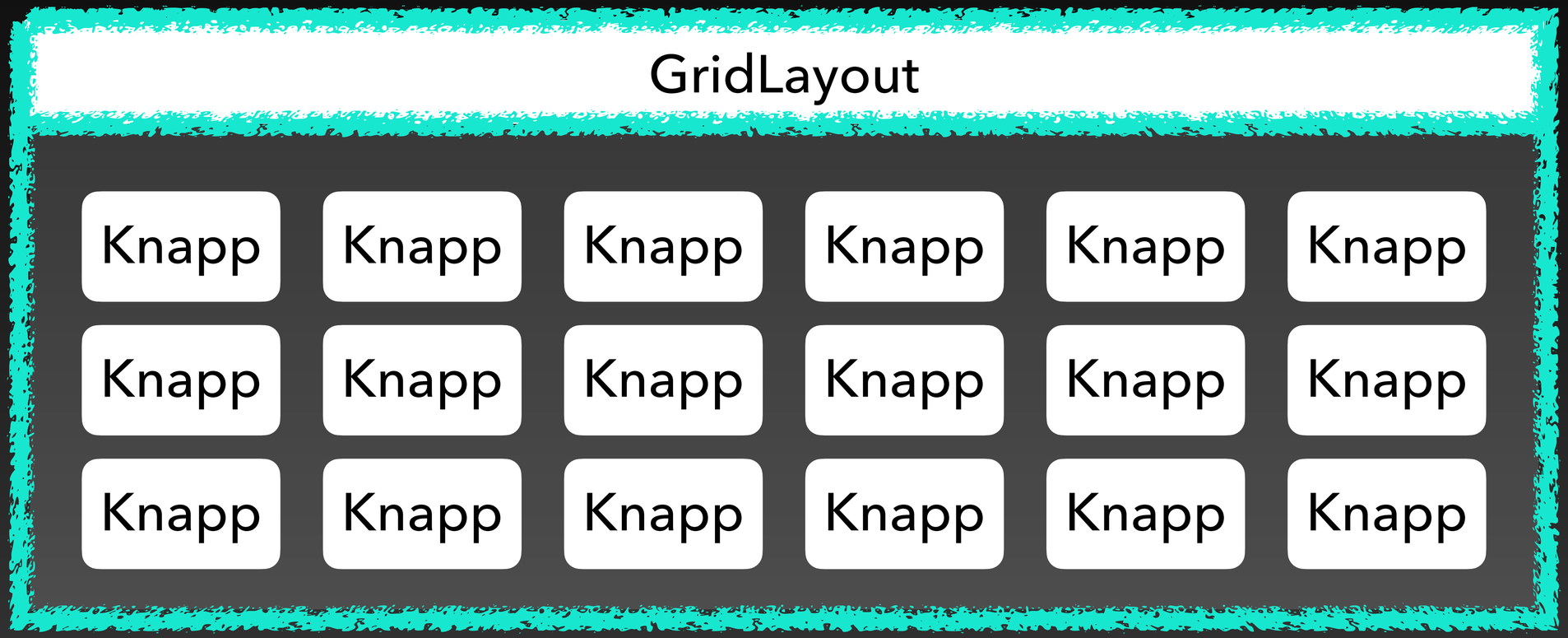
Layout og utseende



Hvis vi vil ha komponentene i et rutenett, så bruker vi GridLayout

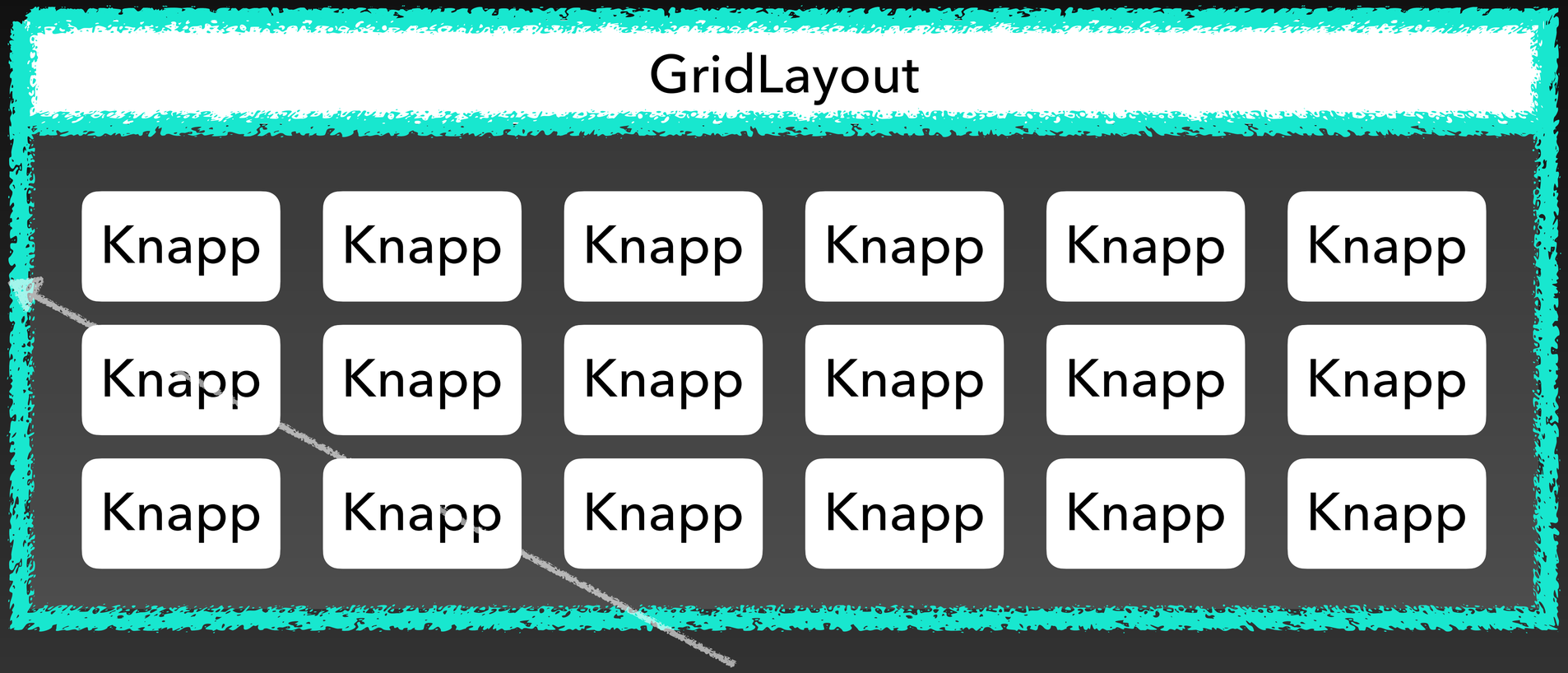
Alle komponentene vil ha samme størrelse

Layout og utseende



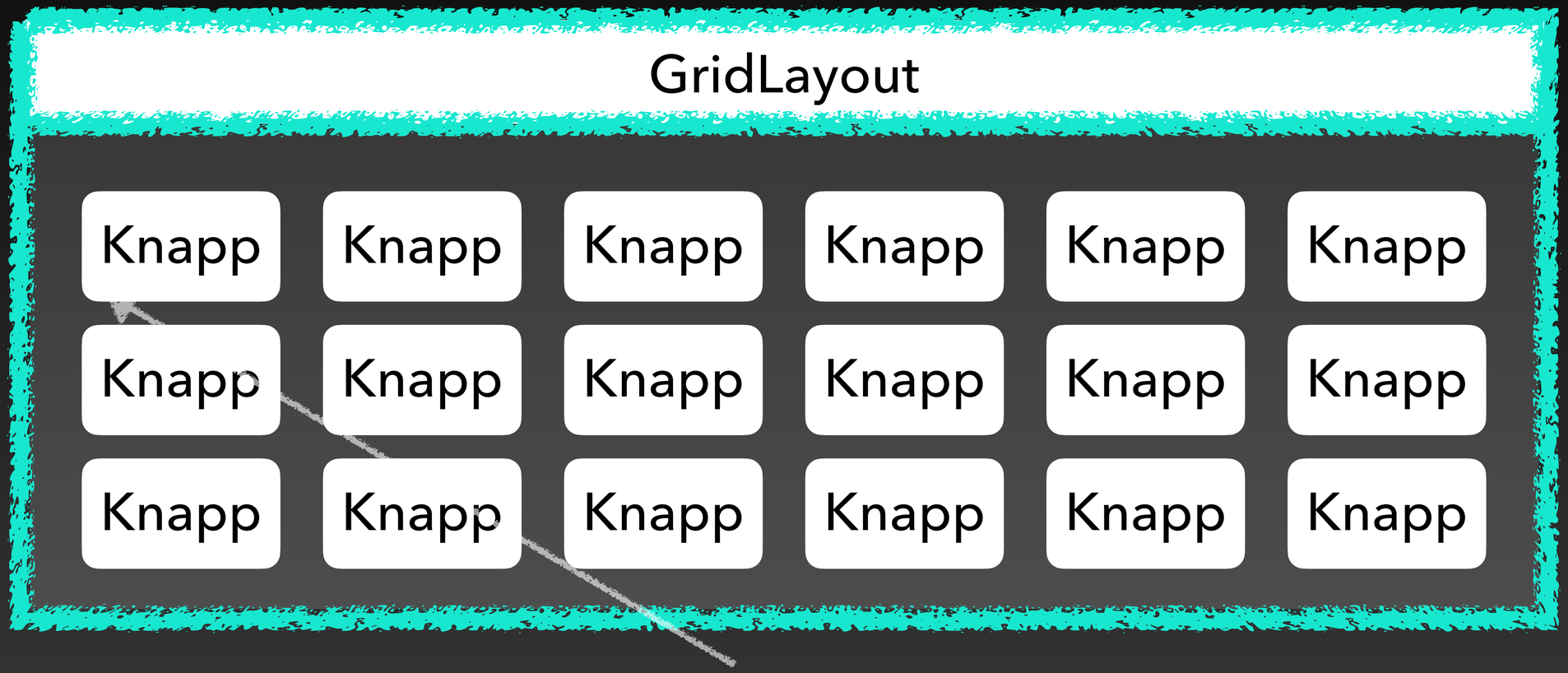
Hvis vi vil ha komponentene i et rutenett, så bruker vi GridLayout

Layout og utseende



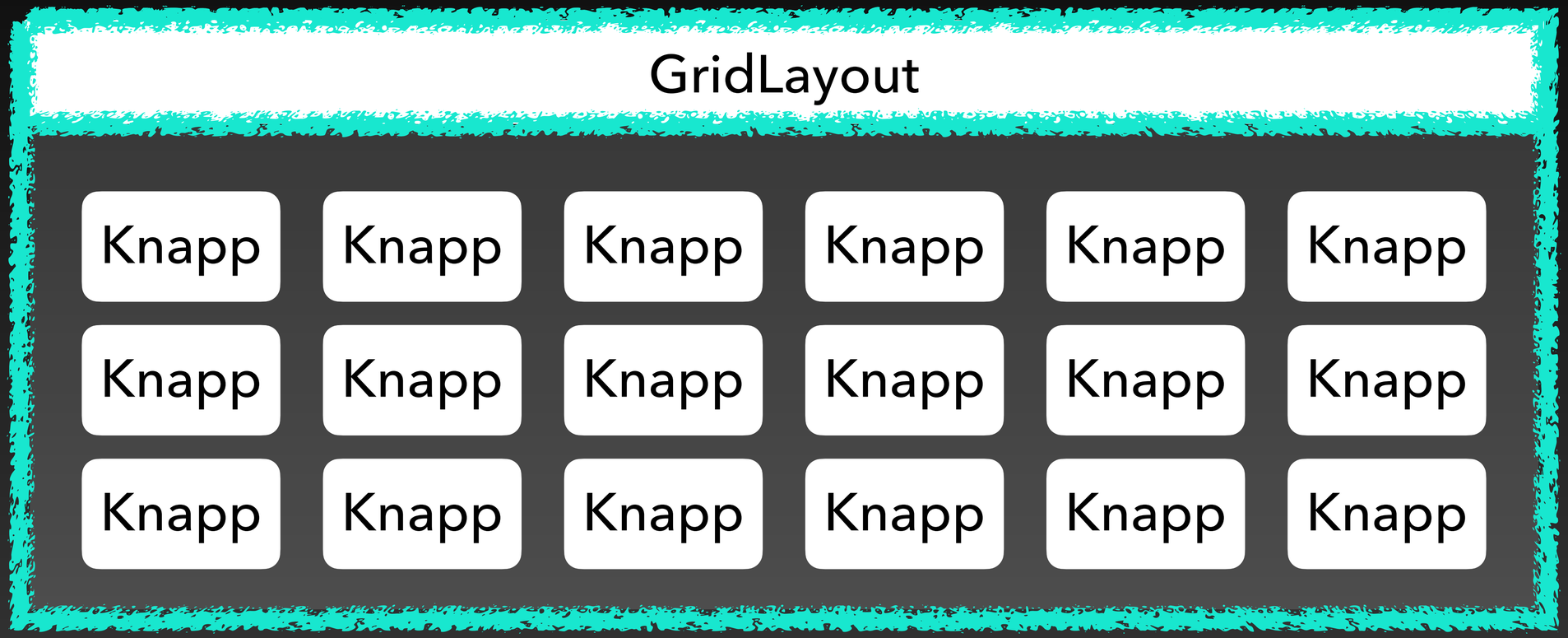
Første tallet bestemmer antall rader

Layout og utseende



Og det andre antallet kolonner

Layout og utseende



Hvis du vil at enten antall rader eller kolonner ikke skal være bestemt, så kan du sette det til 0. Men for kun en av de, i eksempelet her vil antall rader øke

Layout og utseende

FlowLayout endrer ikke størrelsen på komponentene som standard.

Hvis du har komponenter som vokser for store når du bruker de andre layout-typene. Så kan en måte være å lage et panel, som bruker FlowLayout, som du plasserer der du har komponenten som er for stor, og så setter du den inn i panelet.

Layout og utseende

Tekst og fonter

I alle komponetene som lager tekst, så kan du endre fonten som brukes.

Det er fem innebygde skrifttyper eller font familier:

Serif, SansSerif, Monospaced, Dialog, og DialogInput

For hver kan du spesifisere stil, som kan være Plain, Bold, Italic, eller Bold + Italic.

Layout og utseende

For å velge font, så bruker vi metoden setFont()

Layout og utseende

For standard-fontene i Java, så finnes det String-konstanter for å velge de, men du kan også velge andre som du har på maskinen din, men husk at andre kanskje ikke har de

Layout og utseende

Hvis du vil kombinere stiler, så kan du bruke eller-operatoren "|"

Layout og utseende

Størrelsen defineres i punkter, som kan tilsvare en piksel, men kan også være avhengig av oppsettet til maskinen din

Layout og utseende

Farger

Omtrent alle kompnentene kan både forgrunn (typisk teksten) og bakgrunnen endres.

Det er mange måter å definere en farge på.

Java har følgende konstanter

BLACK, BLUE, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW

Layout og utseende

Vi kan bruke en av konstantene for å velge farger

Layout og utseende

Det er også mulig å bruke hex-koder

Layout og utseende

Flyttall mellom 0 og 1, for hver farge-komponent evt. pluss også transparens

Men heltall mellom 0 og 255 går også

Model-View-Controller

Et vanlig design-mønster for å lage et GUI

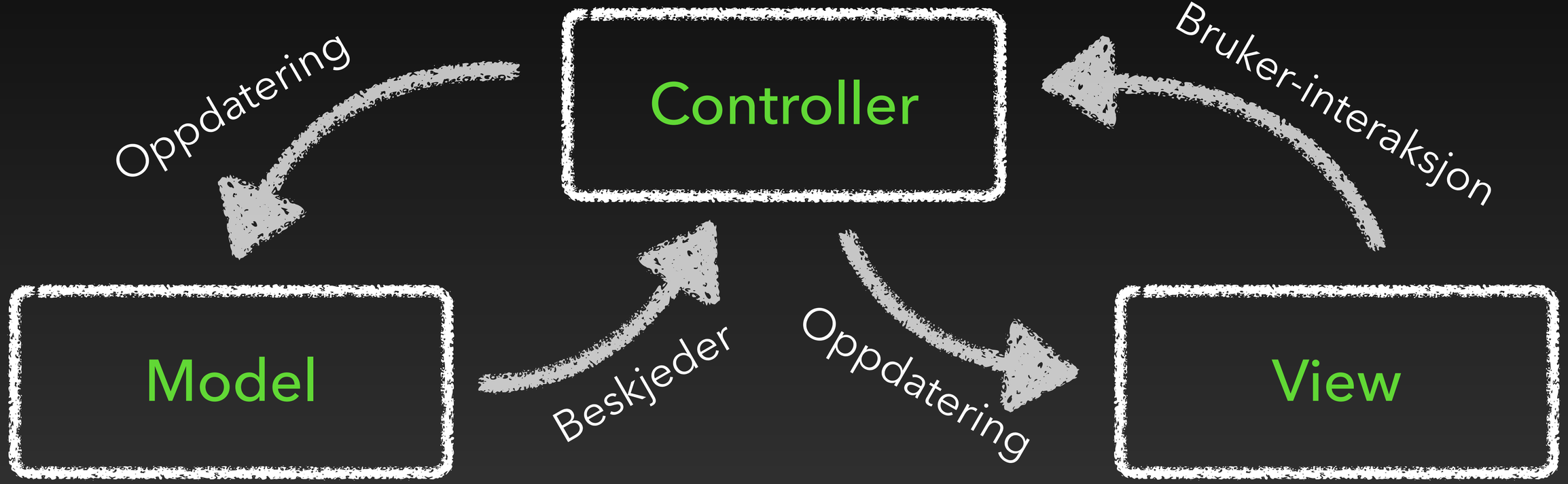
Består av

Model – Data og kode som ikke har med interaksjon/presentasjon

View – Det brukeren ser og kan kommunisere gjennom

Controller – Det som binder sammen Model med View

Model-View-Controller



Model-View-Controller

Model

Ideen er at modellen innkapsles.

Det skal bestå av data, logikken og behandlingen av dataene.

Modellen trenger ikke å være et objekt eller en klasse, men all kommunikasjon mellom de andre objektene skal ikke påvirkes av brukeren og heller ikke vises.

Model-View-Controller

View

Er alt som presenteres for brukeren, og også all input til programmet.

Gjør det enkelt å endre brukergrensesnittet, uten at det påvirker modellen på noen måte.

Model-View-Controller

Controller

For å kunne vise dataene fra modellen, så henter kontrolleren dette fra modellen. Det blir så bearbeidet slik at det kan vises gjennom View.

Oppdatering av modellen fra brukeren skjer også ved at endringer hentes fra View, og så oppdateres det i modellen.

View og Controller er ofte hver sine klasser, en-til-en, med hver sine objekter.