

Velkommen til IN2000!

IN2000

Software Engineering med prosjektarbeid

Introduksjon til emnet

Faglærere og gruppelærere: Se [Kurssiden](#)

Forelesningsplan

- Forelesninger: onsdager og fredager kl. 12.15 - 14.00

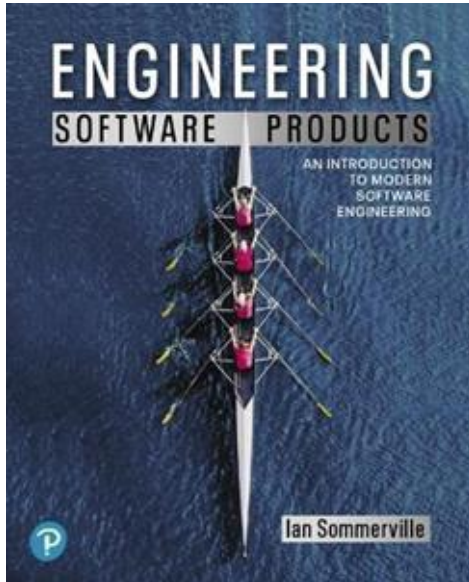
Date	Time	Activity	Room
We 21.01.26	12:15-14:00	Forelesninger Om emnet, viktig informasjon og introduksjon til teknologier/verktøy	QJD Aud Simula
Fr 23.01.26	12:15-14:00	Forelesninger Introduksjon til Android og Kotlin	QJD Aud Simula
We 28.01.26	12:15-14:00	Forelesninger Applikasjonsarkitektur i Android	QJD Aud Simula
Fr 30.01.26	12:15-14:00	Forelesninger Git, API og dataformater	QJD Aud Simula
We 04.02.26	12:15-14:00	Forelesninger Modellering og objektorienterte prinsipper	QJD Aud Simula
Fr 06.02.26	12:15-14:00	Forelesninger Funksjonell programmering med Kotlin	QJD Aud Simula
We 11.02.26	12:15-14:00	Forelesninger Teamarbeid	QJD Aud Simula

Grupper og gruppelærere

- 6 labber/grupper
- **15 gruppelærere + 2 sjefsgruppelærer**
- To til tre gruppelærere per lab
- Når prosjektarbeidet starter vil gruppelærerne bli veiledere for teamene, hvert team får hovedsakelig to veiledere
- **Kick-off 27.februar :-)**

Pensum - Bok

- **Bok:** Ian Sommerville: Software Products. An Introduction to Modern Software Engineering (enten 2019- eller 2020-utgaven)



Pensum—læringskrav skriftlig eksamen

- **Bok:** Ian Sommerville: Software Products. An Introduction to Modern Software Engineering (enten 2019- eller 2020-utgaven)
- Alle forelesninger (forelesningsslides) er pensum
- Obligatoriske oppgaver og ukesoppgaver
- Det kan også bli gitt spørsmål fra prosjektet (og prosjektrapporten) på eksamen

Forventninger til dere

- Innsats (20 studiepoeng), dvs. minst 26 timer i uken i gjennomsnitt
- Dere skal jobbe i team:
 - Bidra i å utvikle en Android app
 - Bidra med å utføre teamaktiviteter
 - Bidra med rapportskriving
 - Presentere arbeidet som er gjort
 - Lese til avsluttende eksamen

26 timer?

*26 timer i uka **gjennomsnitt**. For noen blir det mer for andre mindre.*

*Typisk **før** prosjektarbeidet:*

- 4 timer forelesning + 4 timer labber/grupper (= 8 timer)
- Arbeid med obliger
- Ukesoppgaver
- Jobbe med nytt stoff og nye verktøy alene og **sammen med andre**

± 18 timer til å jobbe med faget sammen med andre og alene utover undervisning.

Under prosjektet?

- Ingen forelesninger eller gruppetimer
- Hvis teamet = 6 personer betyr det **156 timer / uke / team**
- 10 uker = 1560 timer, nesten et årsverk
- Hvis da:
 - Teamet møter veileder 1x i uka i 1 time
 - Teamet har 2x møter i uka som varer 2 timer (4 timer), så er det 21 timer igjen (~4 timer/dag igjen)
- Vi anbefaler hyppige møter / lengre workshops

Viktige milepæler i kurset, med frister

- Oblig 1, individuell, frist tirsdag 4. februar
- Oblig 2, individuell, frist tirsdag 25. februar
- Oblig 3, teamarbeid, frist fredag 11. mars
- **Kickoff fysisk @ IFI, fredag 27. februar, obligatorisk oppmøte**
- Innlevering prosjektarbeid: fredag 15. mai
- Avsluttende skriftlig eksamen: fredag 29. mai
- Muntlige presentasjoner: Uke 23 og 24 (*1. til 12. juni*)

Kommunikasjon

Den viktigste kommunikasjonen skjer i på semestersiden, i forelesninger, laber og ikke minst mellom gruppelærere/veiledere, andre i teamet og medstudenter som tar kurset.

Etter at teamet er etablert velger dere selv verktøy for kommunikasjon, f.eks. Teams, Slack, Discord...

Vurdering—prosjekt og skriftlig eksamen

- **Prosjektarbeid teller 50 % (team)**
 - Kan i enkelte tilfeller gis individuell poengsum til én eller flere i et team.
 - Muntlig presentasjon/eksamen i prosjektet.
 - Det er mulig å oppnå 100 poeng på prosjektarbeidet.
- **Skriftlig eksamen teller 50 % (individuell)**
 - Det er mulig å oppnå 100 poeng på individuell eksamen.

Prosjektarbeid / teamarbeid

- Jobbe i team/grupper
 - Som regel: 6 studenter i hvert team
 - Sammensetning av team – hovedregel:
 - Mulighet til å velge én medstudent som du ønsker å jobbe sammen med
 - Kursledelsen legger til resterende teammedlemmer
- Jobbe etter smidig metodikk
 - Scrum, Kanban, Scrumban, sprinter, planning, backlog, standup, review, retrospektive
- Prosjektet vurderes i tre deler: 1) Rapport, 2) App med prosjektfiler og 3) Teamaktiviteter (mer info kommer senere).

Vurdering—karaktersetting

Eksempel

- Prosjektarbeid (for hele teamet): 84 poeng (av 100 mulige)
- Individuell skriftlig eksamen 72 poeng (av 100 mulige)
- Totalt: $(84 + 72) / 2$, til sammen 78 av 100 poeng, karakteren avhenger av hvor karaktergrensene går.

Prosjektarbeid og case



Meteorologisk
institutt

- Samarbeid med Meteorologisk Institutt
- Eksempler på tidligere case:
 - Båtvarsel Oslofjorden 
 - Badetemperaturer 
 - Klimaendringer (historiske data og framskrivninger) 
 - Luftkvalitet og luftforurensing 
 - Flom og rasvarsel 
 - Beregning av ballistisk bane 
 - Eget case 

Frem til kickoff

- Forelesninger
- Labber/grupper første 6-7 uker:
 - Oppgaveløsning, hjelp med verktøy, obliger mm.
 - Minst to gruppelærere i hver lab/gruppe
 - **Samarbeid** (også med tanke på å finne teammedlemmer)
- Egenarbeid
- ...

Veiledning til teamarbeid

- Veiledning under teamarbeid
 - Ditt team får oppfølging av en eller to veiledere(gruppelærere)
 - Ekstra tiltak om nødvendig; hvis teamet ikke fungerer, problemer med møter etc.
 - Forventningsavklaring til samarbeidet er viktig!
 - Teamavtale som del av oblig 3
- Ved svært ujevn arbeidsfordeling/konflikter i teamet, **gi beskjed!**

Introduksjon i teknologier

Om emnet

- **Teamarbeid** i fokus
- IN2000 er annerledes enn emnene dere har hatt før
- Du kommer **ikke** til å lære alt du trenger til prosjektet i forelesning
 - men vi skal gi dere forutsetningene for å kunne lære selv

IN2000 - Læringsmål

Etter å ha fullført IN2000:

- har du kunnskap om de viktigste systemutviklingsmetodene, inkludert deres styrker og svakheter
- har du kunnskap om sentrale prosesser og aktører i prosjekt- og teamarbeid som anvender smidige prinsipper
- har du kunnskap om følgende aktiviteter innenfor systemutvikling:
kravinnstilling og -analyse, design, programmering, testing, samt vedlikehold og videreutvikling
(...)

IN2000 - Læringsmål

Etter å ha fullført IN2000:

- har du kunnskap om de viktigste systemutviklingsmetodene, inkludert deres styrker og svakheter
- har du kunnskap om sentrale prosesser og aktører i prosjekt- og teamarbeid som anvender smidige prinsipper
- har du kunnskap om følgende aktiviteter innenfor systemutvikling:
kravinnsamling og -analyse, design, programmering, testing, samt vedlikehold og videreutvikling
(...)



IN2000 - Læringsmål (forts.)

Etter å ha fullført IN2000:

(...)

- kan du anvende profesjonelle systemutviklingsmetoder, -teknikker og -verktøy
- har du kompetanse for å arbeide i team og evne til å reflektere over eget og teamets arbeid i systemutviklingsprosjekter
- kjenner du til metoder og prinsipper for innebygget sikkerhet og universell utforming

Forventningsstyring

- Dette er **ikke** et rent teknisk kurs
- Teknologier er verktøy som hjelper å løse prosjektoppgaven!
- Alle må likevel kunne litt om teknologiene og verktøyene
 - Oblig 1 og 2
 - Eksamen
 - **Bidra i prosjektarbeidet**
- **Kunsten å lære nye ting på egenhånd!**

Teknologier og verktøy

“I IN2000 skal dere i team utvikle en **Android app**. Appen skal skrives med programmeringsspråket **Kotlin**, ved bruk av **Android Studio**. Dere skal benytte **git** til versjonshåndtering. Appen skal hente data fra **API-ene** til Meteorologisk institutt.”





...men først...hva blir output?

```
val numbers = listOf(1, 2, 3, 4, 5)
```

```
val result = numbers
```

```
    .filter { it % 2 == 0 }
```

```
    .sumOf { it * 2 }
```

```
println(result)
```

...men først...hva blir output?

```
val numbers = listOf(1, 2, 3, 4, 5)

val result = numbers
    .filter { it % 2 == 0 } // [2, 4]
    .sumOf { it * 2 }

println(result)
```

...men først...hva blir output?

```
val numbers = listOf(1, 2, 3, 4, 5)

val result = numbers
    .filter { it % 2 == 0 } // [2, 4]
    .sumOf { it * 2 } // Summen av [4, 8]

println(result)
```

...men først...hva blir output?

```
val numbers = listOf(1, 2, 3, 4, 5)

val result = numbers
    .filter { it % 2 == 0 } // [2, 4]
    .sumOf { it * 2 } // Summen av [4, 8]

println(result)
```

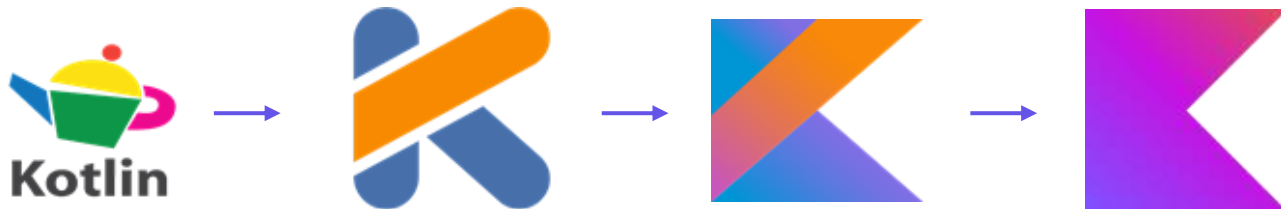
>> 12

La oss sette i gang!



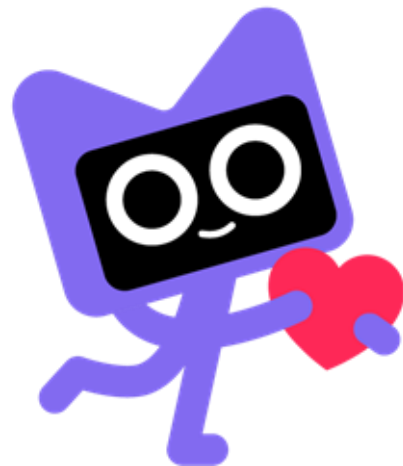
Kort om Kotlin

- Utviklet av [JetBrains](#)
- Første utgivelse: 2011
 - Versjon 1.0 i 2016
 - Versjon 2.0 i 2024
- Open Source
- Multiparadigme
- .kt



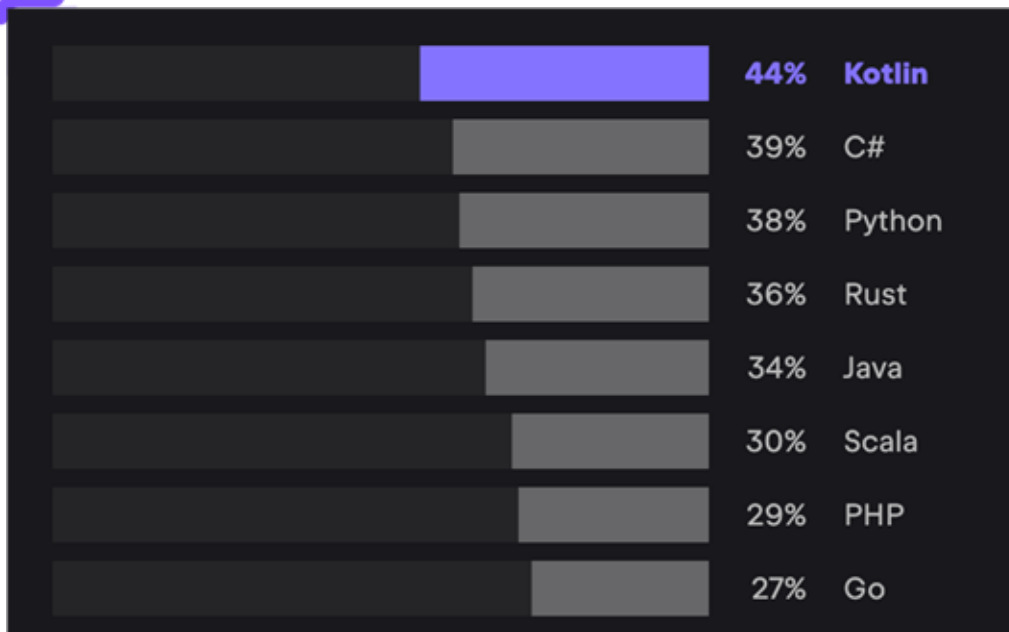
Kort om Kotlin

- Utviklet av [JetBrains](#)
- Første utgivelse: 2011
 - Versjon 1.0 i 2016
 - Versjon 2.0 i 2024
- Open Source
- Multiparadigme
- .kt
- Egen [maskot](#) →



Navn: “Kodee”

Kotlin - popularitet blant utviklere



<https://www.jetbrains.com/lp/devecosystem-2022/>

Kotlin - “Hello World” vs. Java

Tidligere versjoner av Java

```
public class Hey {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

Kotlin

```
fun main() {  
    println("Hello World!")  
}
```

Kotlin - “Hello World” vs. Java

Nyere Java

```
void main() {  
    System.out.println("Hello, World!");  
}
```

Kotlin

```
fun main() {  
    println("Hello World!")  
}
```

Kotlin - Variabler val

val name : data type = initial value

```
val x: Int = 5  
val y: String = "Foo"
```

```
val x = 5  
val y = "Foo"
```

Immutable / Mutable

val



“Immutable”

var



“Mutable”

Immutable / Mutable

val

“Immutable”

val x = 5

x = 6 ❌

val cannot be reassigned

Eksempel val

var

“Mutable”

var x = 5

x = 6 ✅

Eksempel var

String templates

Uten String templates

```
val i = 10  
println("i = " + i)  
  
>> i = 10
```

Med String templates

```
val i = 10  
println("i = $i")  
  
>> i = 10
```

String templates (forts.)

Uten String templates

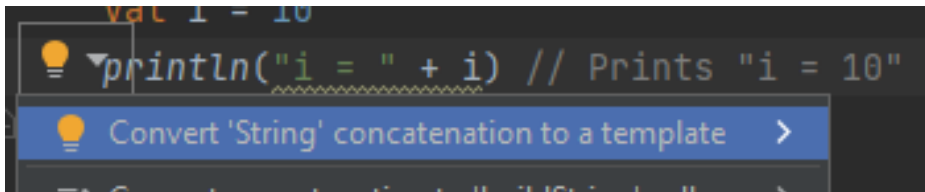
```
val i = 10
println("i = " + i)

// Prints "i = 10"
```

Med String templates

```
val i = 10
println("i = $i")

// Prints "i = 10"
```



String templates (forts.)

Kan også template uttrykk!

```
val s = "abc"  
println("$s.length is ${s.length}")  
  
// Prints "abc.length is 3"
```

Funksjoner - Grunnleggende oppbygging

```
fun name () {  
    body  
}
```

```
fun greeting() {  
    println("Hello World")  
}
```

Funksjoner - Med returverdi(er)

```
fun name () : return type {  
    body  
    return statement  
}
```

```
fun greeting(): String {  
    val hello = "Hello World"  
    return hello  
}
```

Funksjoner - Med parameter(e)

```
fun name ( parameters ) {  
    body  
}
```

```
fun greeting(x: String, y: String) {  
    println("$x $y")  
}
```

```
fun main() {  
    greeting("Hello", "World")  
}
```

Funksjoner - Single-Expression function

fun **name** (**parameters**) : **return type** = **body**

```
fun greeting(first: String, second: String): String = "$first $second"
```

```
fun main() {  
    println(greeting("Hello", "World"))  
}
```

Kotlin - Interoperabilitet med Java

Java (Car.java)

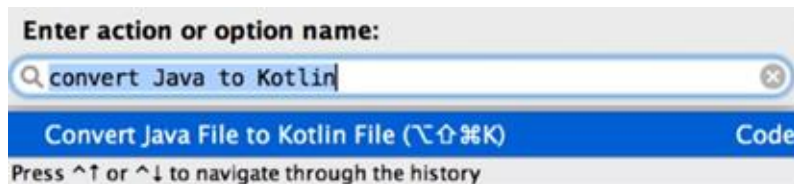
```
public class Car {  
    String number;  
    boolean isElectric;  
  
    public Car(String number, boolean isElectric) {  
        this.number = number;  
        this.isElectric = isElectric;  
    }  
}
```

Kotlin

```
val myElectricCar = Car("EL12345", true)  
val myPetrolCar = Car("AS12345", false)
```

Konvertering fra Java til Kotlin

- Android Studio (IntelliJ) har innebygd konvertering fra **Java** til **Kotlin**



**android
studio**



Android - Om

- Operativsystem som Google eier og utvikler
- Siste versjon: [Android 16](#)
- Andel smarttelefoner desember 2025...
 - ...i verden: ~72%
 - ...i Norge: ~39%

> [Kilde](#)



Android Studio - Om

- Integrated Development Environment (IDE)
- Verktøy for å lage Native Apps til Android
- Open Source
- Basert på IntelliJ Jetbrains IDEA
 - Stor omgivelse
 - **Trenger kun å kjenne til deler av det!**



Android Studio - Native vs. Hybrid app-utvikling

NATIVE

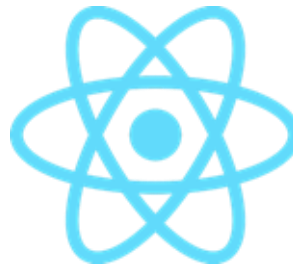


Android Studio



Xcode

HYBRID



React Native



Flutter



Multiplatform

Android Studio - Emulator

- Kjøre og teste programmer
- Simulerer en Android mobiltelefon, virtuelt
- Lett å kjøre ulike versjoner / config

> <https://developer.android.com/studio/run/emulator>



Android Studio - Specs



Minner om: [Laptop kjøpsråd ved IFI](#)

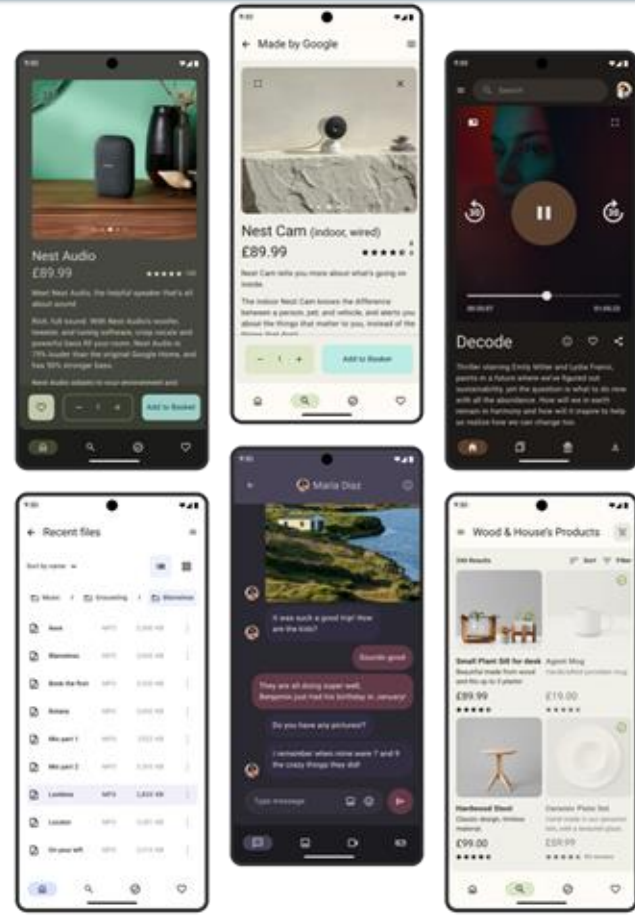
Android Studio - Brukergrensesnitt

- **Jetpack Compose**

- Det vi bruker i IN2000
- Skriver UI-logikk med Kotlin og lager såkalte “Composables”

- **Views (XML-basert)**

- Den eldre måten å definere brukergrensesnitt ble laget på før



Hva er Jetpack Compose?

- Anbefalte rammeverk for brukergrensesnitt i Android
- Lansert i 2019, første stabile versjon i 2021
- **Deklarativt UI-rammeverk**
 - Regenererer hele skjermen fra “scratch”
 - Compose velger hva som blir oppdatert til enhver tid for å ikke måtte generere hele skjermen på nytt



Jetpack Compose

- En komposisjon (Composition) beskriver utseende til appen
- En komposisjon er bygget opp av flere Composable-funksjoner (tenk på musikk)
- Composables er Immutable, dvs.
 - At den ikke kan endres etter at den er opprettet.
 - UI må oppdateres, dette kalles rekomponering (Recomposition)

Composable funksjoner

- Annoteres med `@Composable`
- PascalCase
- Har ingen returverdi
- Bør være idempotent (*ikke ha sideeffekter*)
- Kan bruke Kotlin som vanlig i funksjonen, f.eks. bruke if/else for kontrollflyt og velge hva som skal vises basert på logikk

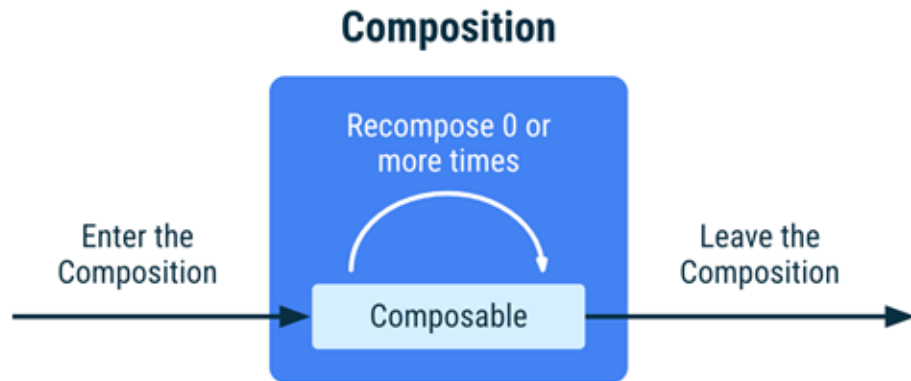
```
@Composable
fun Greeting(name: String) {
    Text("Hello $name")
}
```

Compose - Noen begreper

Komposisjonen er beskrivelsen av brukergrensesnittet som blir bygd av Jetpack Compose når den kjører composables.

Initiell komposisjon er den første opprettelsen av komposisjonen

Rekomposisjon er det å kjøre en composable på nytt når informasjonen/input til funksjonen endres



Kort oppsummert

Android Studio → verktøyet vi bruker for å utvikle
(i stedet for f.eks. VS Code)



Jetpack Compose → rammeverk for å definere brukergrensesnitt



Kotlin → programmeringsspråket vi skal benytte i dette kurset



Ressurser i IN2000

Ressurser i IN2000

Interne ressurser

- Semestersiden
- Forelesninger
- Lab / Gruppetimer
- Ukesoppgaver
- **Teamet**

Eksterne ressurser

- [Kotlin dokumentasjon](#)
- [Android dokumentasjon](#)
- [Jetpack Compose dokumentasjon](#)
- Google m/StackOverflow
- Codelabs fra Google (se *ukesoppgaver*)

Obligatorisk innlevering 1 legges ut i dag!