#4

Android Studio

Android Studio

“ Switches: On / Off,
Let’s talk about objects too! ”

INNEHÅLLSFÖRTECKNING

01

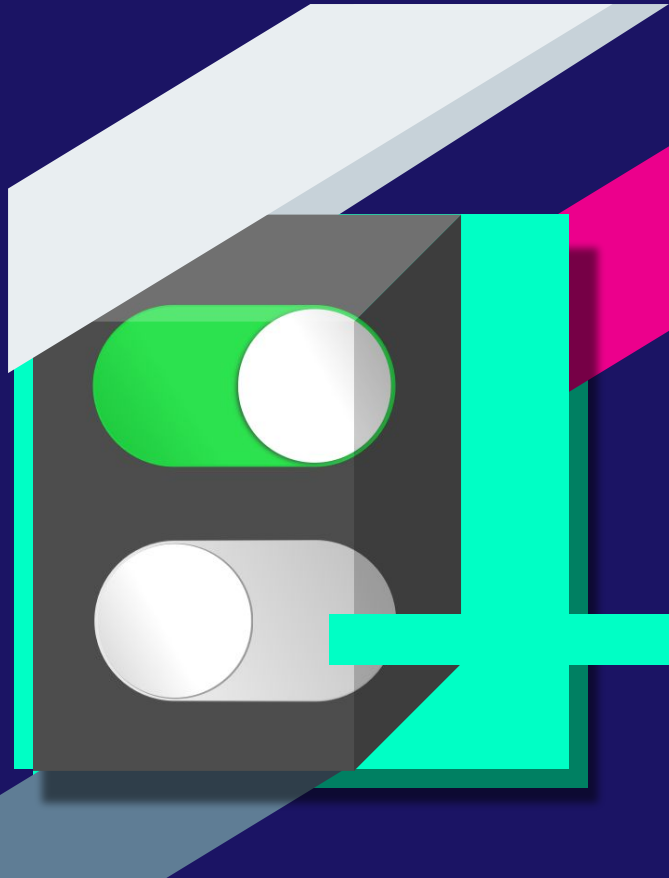
ÖVERSIKT


Switches, Backports
& Objects

Step by step

Låt oss kombinera Switch, Objekt och bakåtkompatibilitet!

- Switch
- Objects
- ‘Backport’
- Conditional Rendering - If statements



02

Switch & Backporting

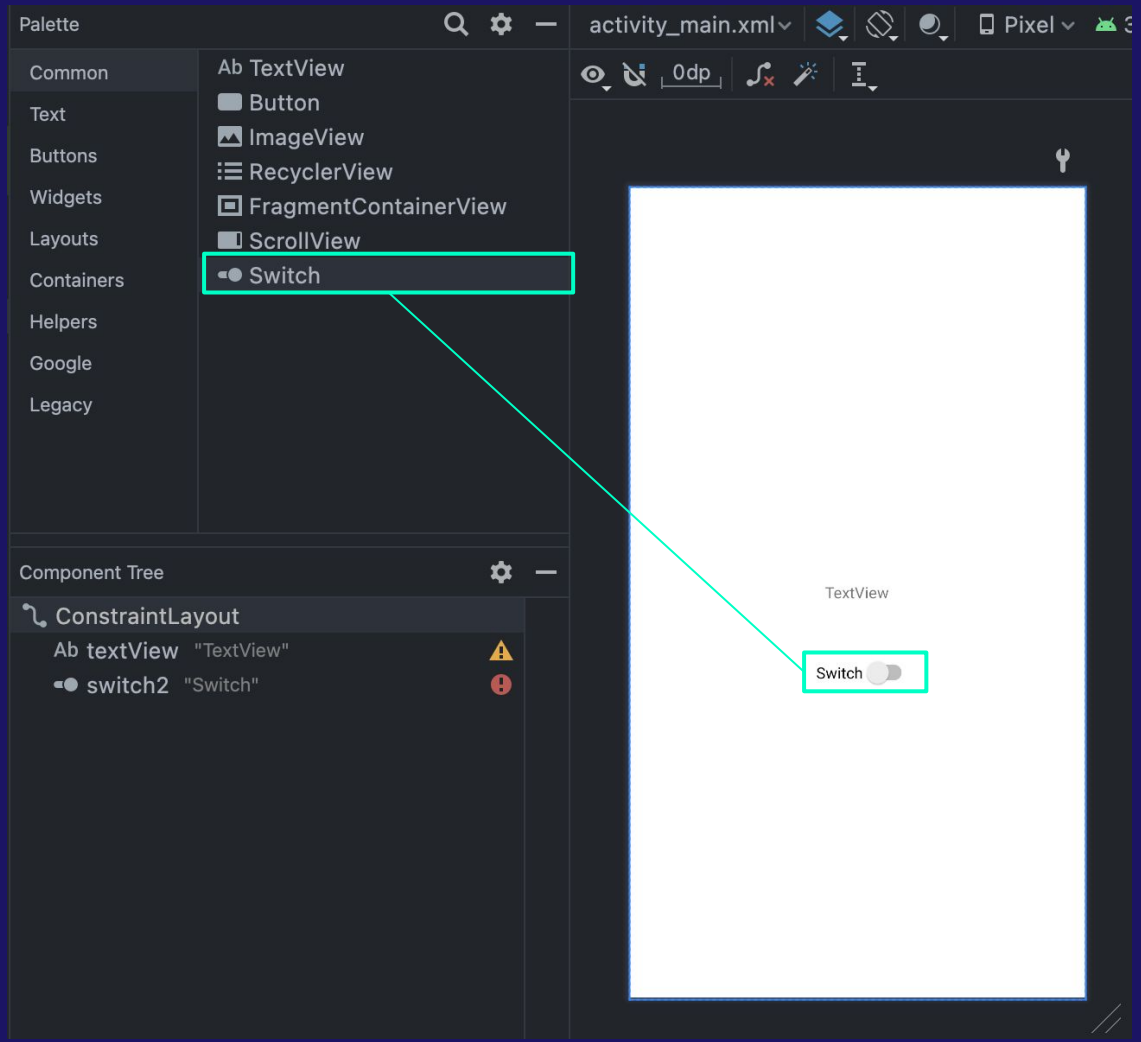
</>

“A Switch is a two-state toggle widget. Users can drag the switch "thumb" back and forth to select either of two options or simply tap the switch to toggle between options. ”

- developer.android.com


Switch Error?

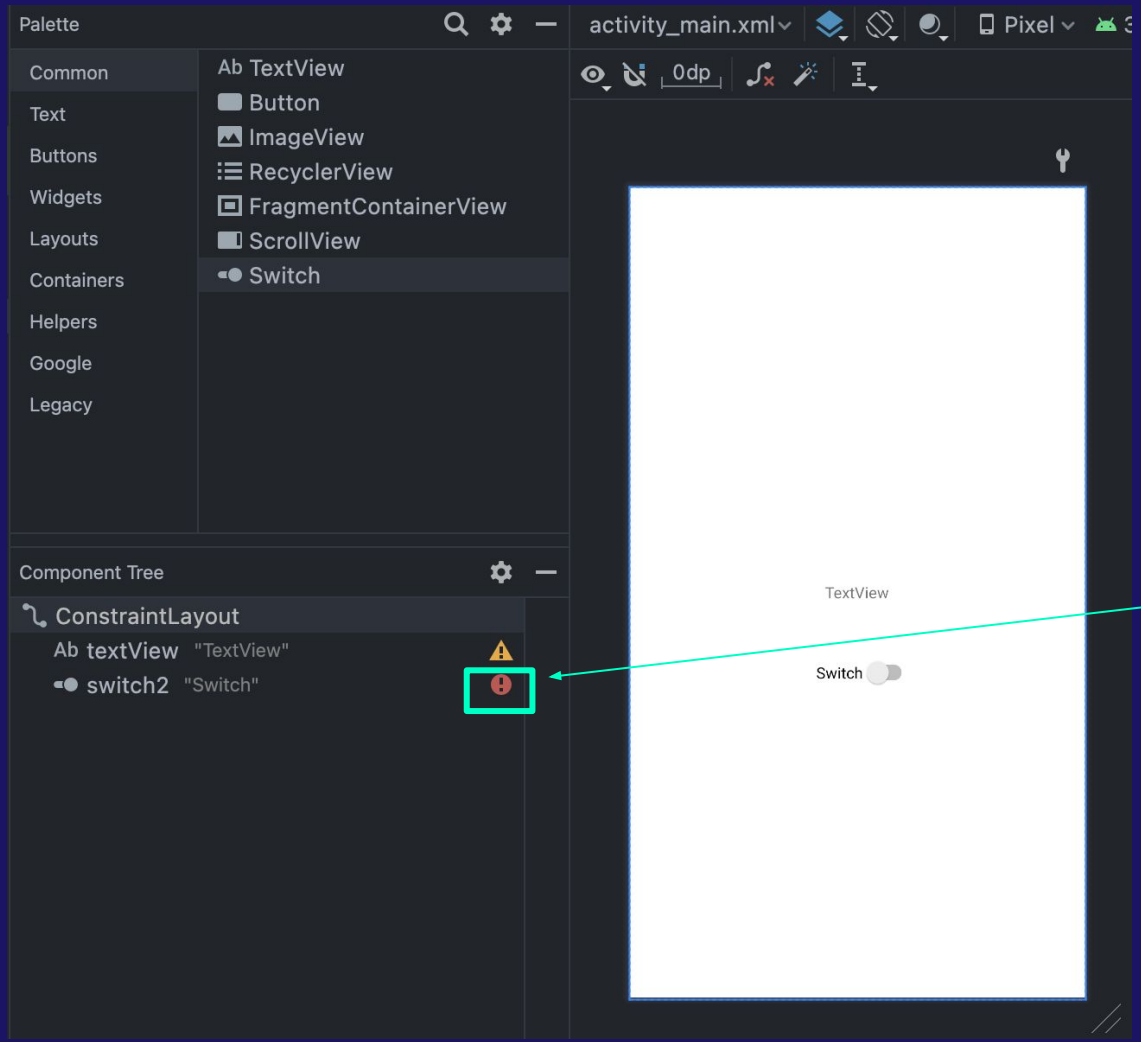
Switch Error



Vi hittar ‘Switch’ komponenten inom ‘COMMON’!

Drag och släpp den in i er vy!

Switch Error

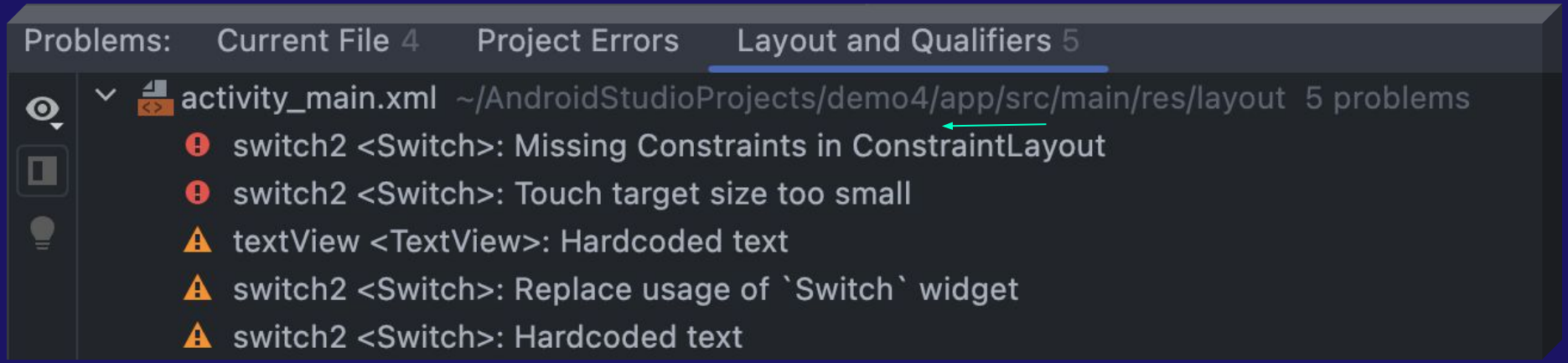


Vi ser att vi redan fått ett ‘error’.

Men vänta nu…
Är det verkligen ett error?

Tryck på den röda symbolen!

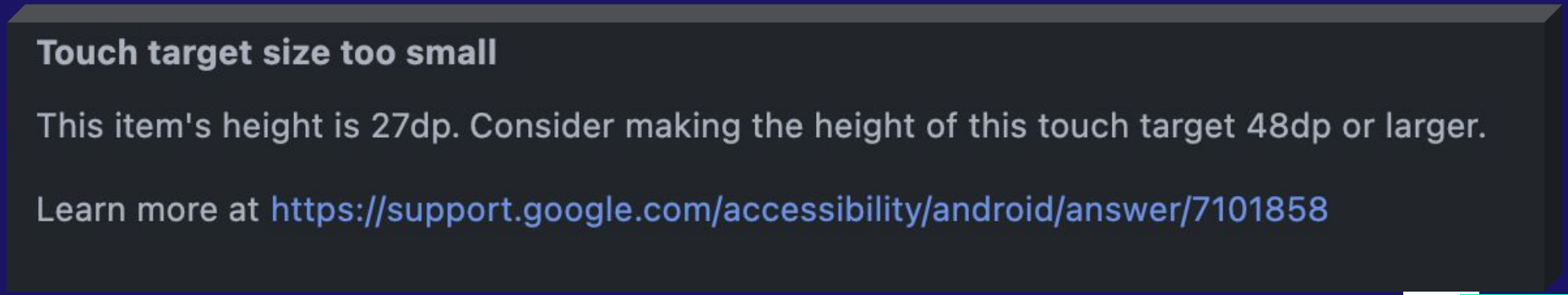
Switch Error



Nej!
Detta är inget ‘error’ utan är enbart en rekommendation.
Precis som att vi saknar en ‘Constraint’.

Låt oss åtgärda detta.

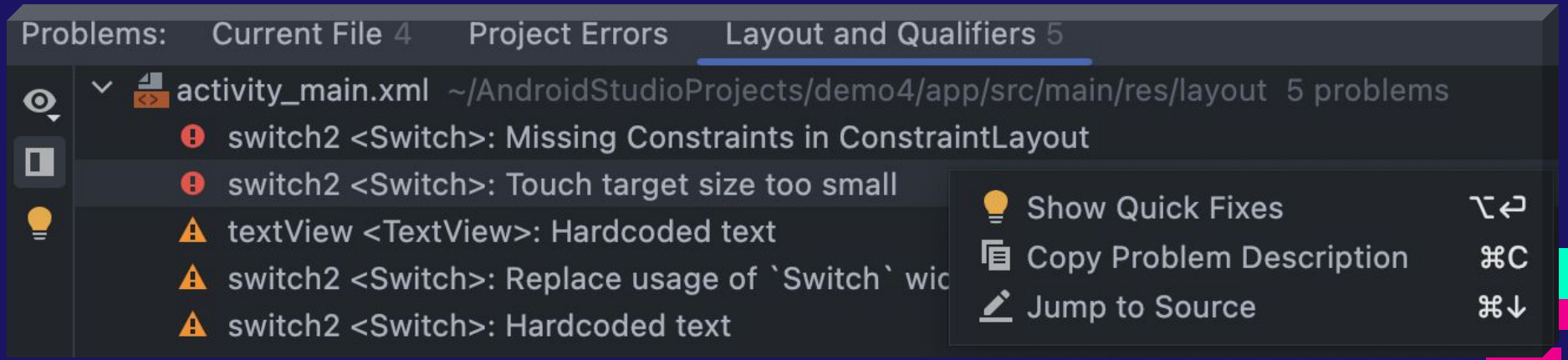
Switch Error



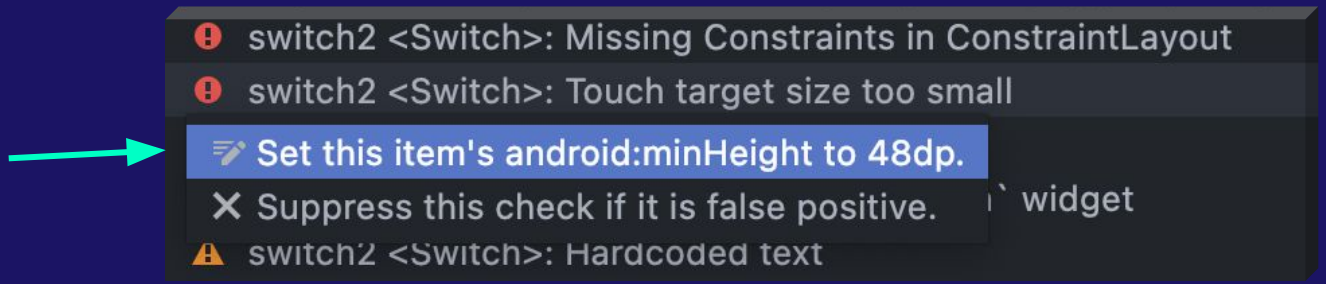
Denna komponent är för liten och bryter mot Android Studio's rekommendationer

Switch Error

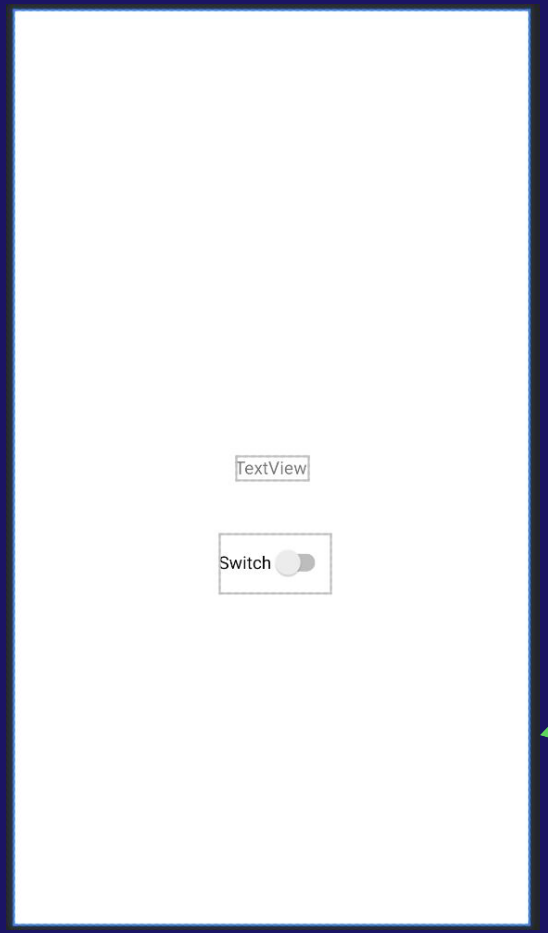
Högerklicka på problemet -> ‘show quick fixes’



Switch Error



Switch Error Solution



Resultatet blir att ‘switch’ bli lite större!

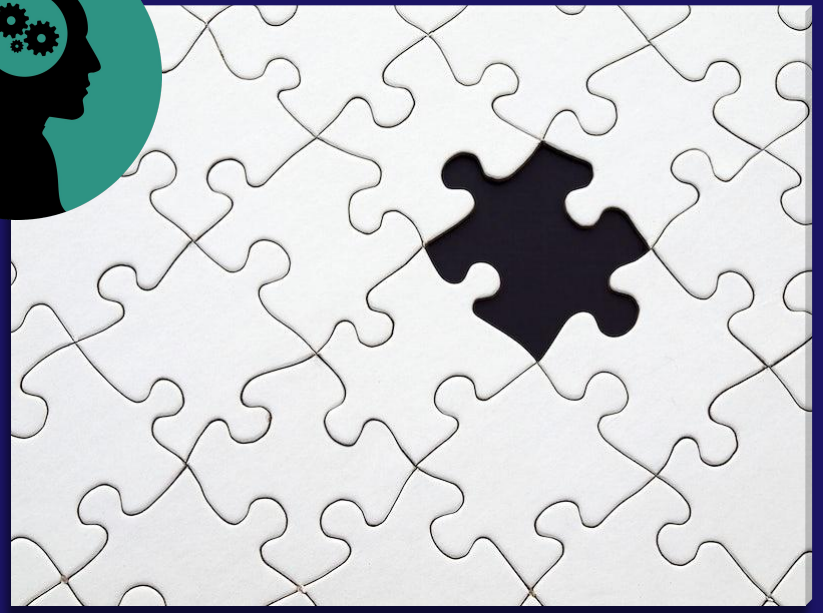
Problem

Vad är det för error jag får?
Jag har inte gjort något!

Solution

Håll över problemet, kolla vad det finns för lämpliga lösningar!

Frågor?




Backporting

Backporting

source developer.android.com

SwitchCompat is a complete backport of the core `Switch` widget that brings the visuals and functionality of the toggle widget to older versions of the Android platform. Unlike other widgets in this package, SwitchCompat is not automatically used in layouts that include the `<Switch>` element. Instead, you need to explicitly use `<androidx.appcompat.widget.SwitchCompat>` and the matching attributes in your layouts.

The thumb can be tinted with `setThumbTintList` and `setThumbTintMode` APIs as well as with the matching XML attributes. The track can be tinted with `setTrackTintList` and `setTrackTintMode` APIs as well as with the matching XML attributes.

Backporting

Ej bakåtkompatibelt

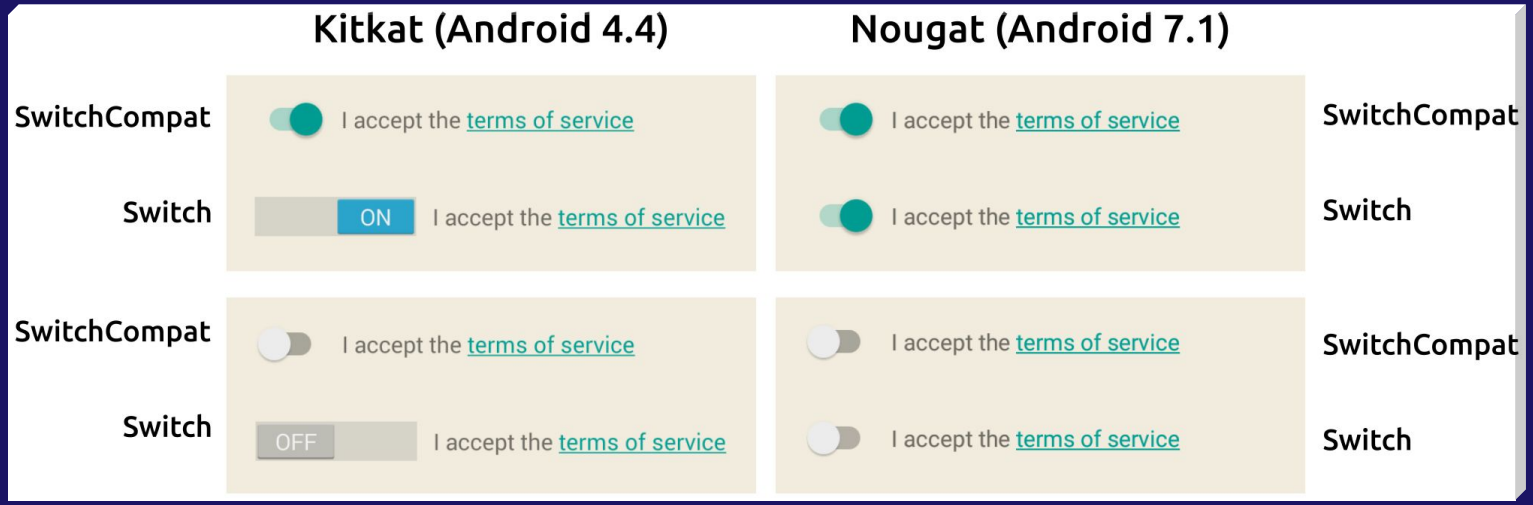
```
<Switch
    android:id="@+id/switch4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Switch"
    tools:layout_editor_absoluteX="162dp"
    tools:layout_editor_absoluteY="419dp"
    tools:ignore="MissingConstraints,UseSwitchCompatOrMaterialXml" />
```

Backporting

Bakåtkompatibelt!

```
<androidx.appcompat.widget.SwitchCompat
    android:id="@+id/switch4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Switch"
    tools:layout_editor_absoluteX="162dp"
    tools:layout_editor_absoluteY="419dp"
    tools:ignore="MissingConstraints,UseSwitchCompatOrMaterialXml" />
```

Backporting



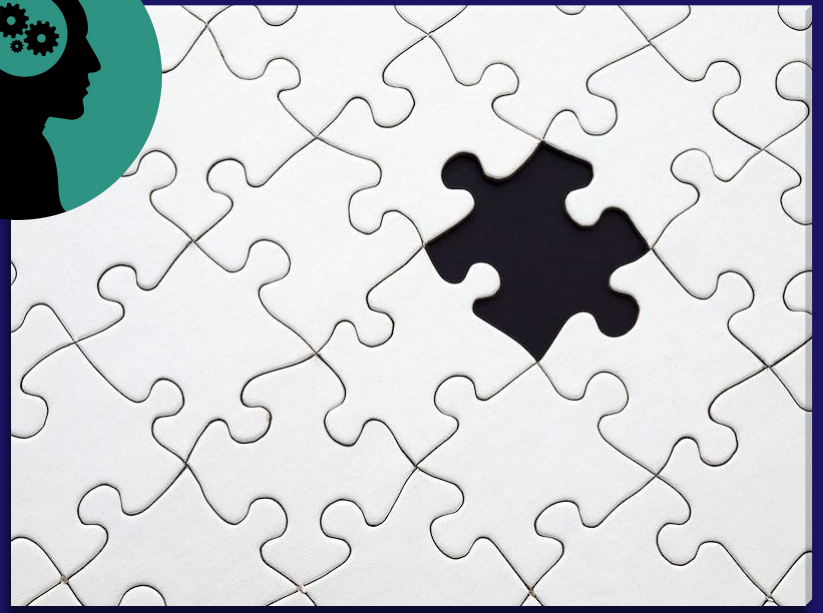
Problem

Hur gör vi komponenter bakåtkompatibelt?

Solution

Skriv appcompat + komponentnamnet
Så får ni resultatet för alla bakåtkompatibla komponenter!

Frågor?




ViewBinding

R.id.btn_submit

Ärlighet: Att arbeta på detta sätt är förvirrande och frustrerande...

```
val button: Button = findViewById(R.id.btn_navigate)
val username: EditText = findViewById(R.id.textInput_username)
```

Kotlin Android Extension lösning?

DEPRECATED

Det fanns en lösning - men detta togs bort...!

Hursomhelst... Det finns flera alternativ...!

ViewBinding!

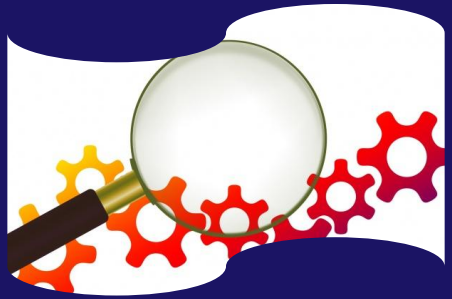
```
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding    // binding Reference

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }
}
```

Visibility modifier, init, variable, name and datatype

LateInit?

```
val age: Int
```

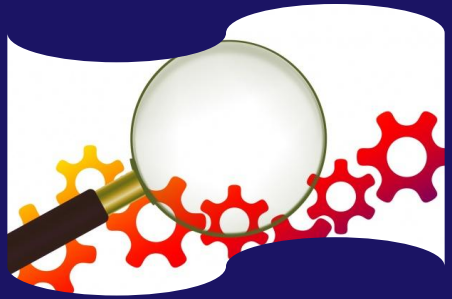
Vilket värde börjar denna variabel med?



LateInit?

```
val age: Int
```

Vilket värde börjar denna variabel med?



Vad är null?

LateInit?

```
9  class MainActivity : AppCompatActivity() {
10
11      val age2: Int
12
13      override fun onCreate(savedInstanceState: Bundle?) {
14          super.onCreate(savedInstanceState)
15
16          val age: Int
17
18
19
20      }
21  }
```

More on lateinit: source: kotlinlang

Inflate?

```
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding    // binding Reference

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Initialize ViewBinding
        binding = ActivityMainBinding.inflate(layoutInflater)

    }
}
```

Inflate är ett sätt att skapa en vy genom XML.

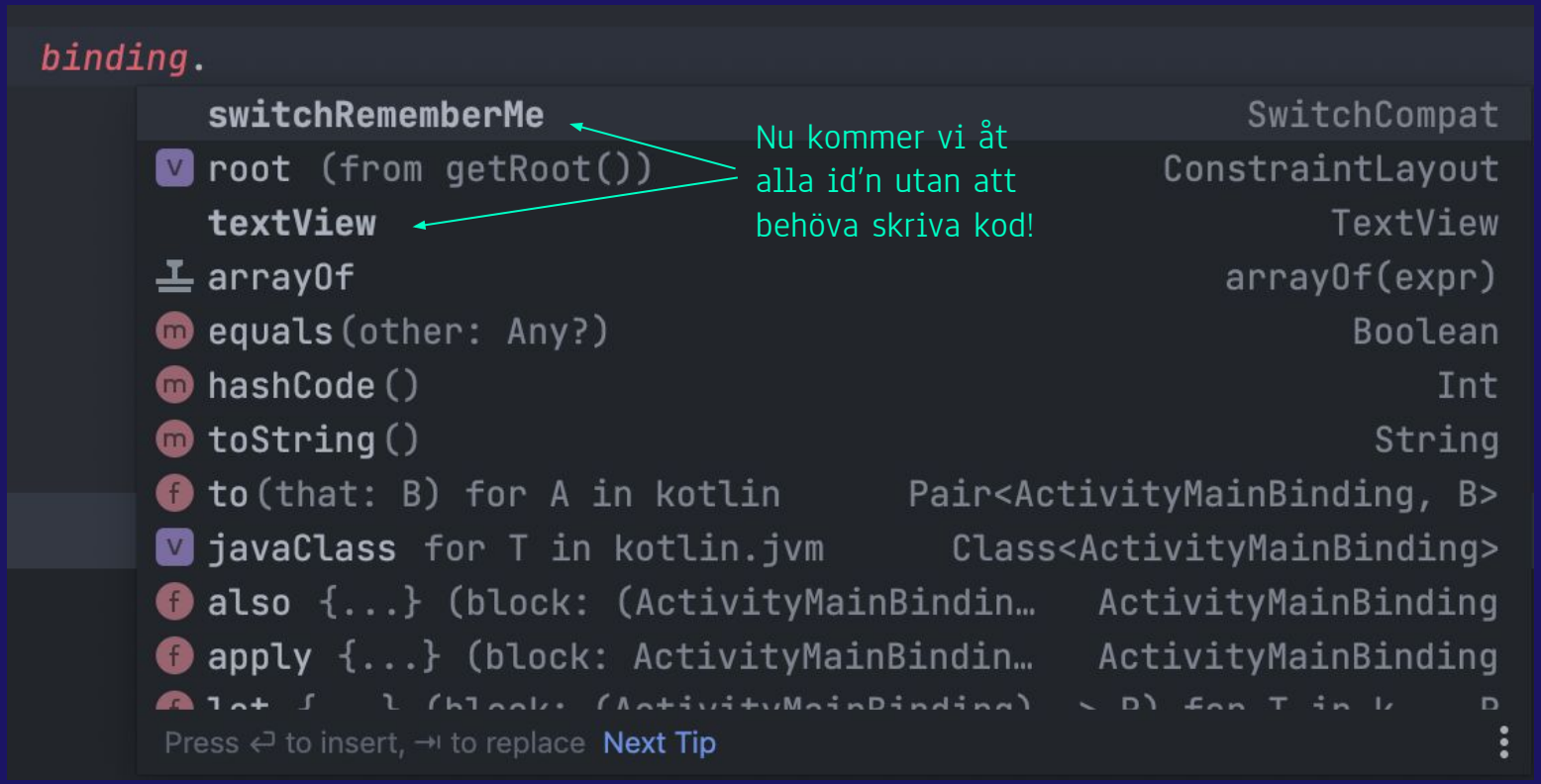
Source: developer.android

ViewBinding!

```
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding    // binding Reference

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Initialize ViewBinding
        binding = ActivityMainBinding.inflate(layoutInflater)
        val view = binding.root                 // get root View reference
        setContentView(view)                    // setContentView to our binding
    }
}
```

binding.root refererar till rot vy'n
'root view' är den yttersta vy behållaren i din layout

IN ACTION


binding.
switchRememberMe SwitchCompat
root (from getRoot()) ConstraintLayout
textView TextView
arrayOf arrayOf(expr)
equals(other: Any?) Boolean
hashCode() Int
toString() String
to(that: B) for A in kotlin Pair<ActivityMainBinding, B>
javaClass for T in kotlin.jvm Class<ActivityMainBinding>
also {...} (block: (ActivityMainBindin… ActivityMainBinding
apply {...} (block: ActivityMainBindin… ActivityMainBinding
Press ⏎ to insert, ⇥ to replace Next Tip
Nu kommer vi åt alla id'n utan att behöva skriva kod!

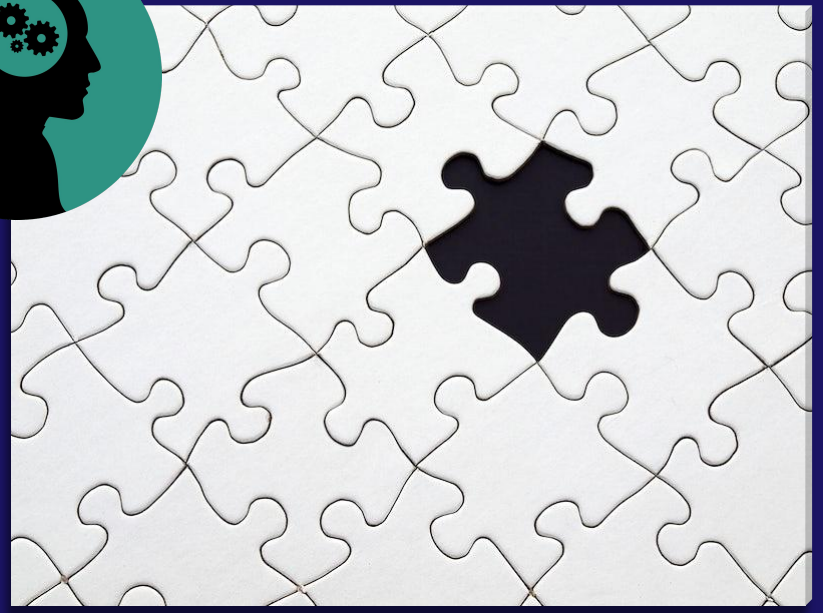
Problem

Måste vi arbeta med findViewById?

Solution

ViewBinding är ett modernare alternativ mot ‘FindViewById’

Frågor?




Switch Component

Switch Component

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    // Initialize ViewBinding
    binding = ActivityMainBinding.inflate(layoutInflater)
    val view = binding.root
    setContentView(view)
     val rememberMe = binding.switchRememberMe                    // Switch
     var rememberMeIsChecked = rememberMe.isChecked.toString()    // isChecked
     val textViewRememberMe = binding.textViewRememberMe          // TextView
```

rememberMe = komponent referens
rememberMeIsChecked = boolean on/off

Switch Component

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    // Initialize ViewBinding
    binding = ActivityMainBinding.inflate(layoutInflater)
    val view = binding.root
    setContentView(view)

    val rememberMe = binding.switchRememberMe                    // Switch
    var rememberMeIsChecked = rememberMe.isChecked.toString()    // isChecked
    val textViewRememberMe = binding.textViewRememberMe          // TextView
```

Analysera följande kod...

Switch Component

```
val rememberMe = binding.switchRememberMe                    // Switch
var rememberMeIsChecked = rememberMe.isChecked.toString()    // isChecked
val textViewRememberMe = binding.textViewRememberMe          // TextView

rememberMe.setOnClickListener { it: View!


    rememberMeIsChecked = rememberMe.isChecked.toString()
}
```

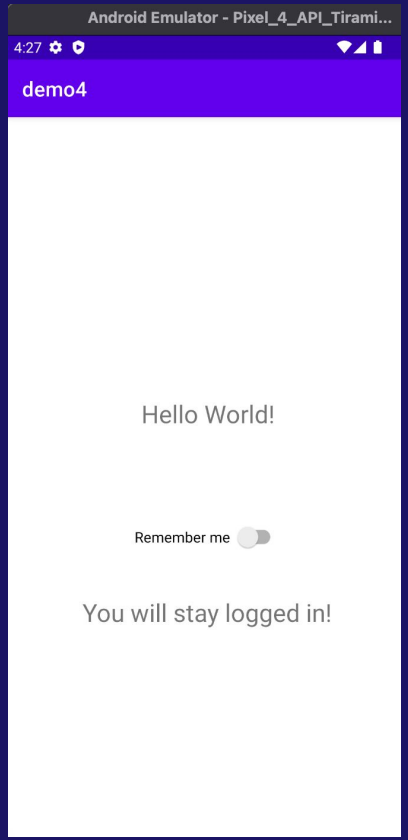
Fungerar det här?

Switch Component

Lösning:

```
val rememberMe = binding.switchRememberMe                    // Switch
var rememberMeIsChecked = rememberMe.isChecked.toString()    // isChecked
val textViewRememberMe = binding.textViewRememberMe          // TextView

rememberMe.setOnClickListener { it: View!
    println("Current State: $rememberMeIsChecked")


    rememberMeIsChecked = rememberMe.isChecked.toString()
    rememberMe.text = rememberMeIsChecked
}
```

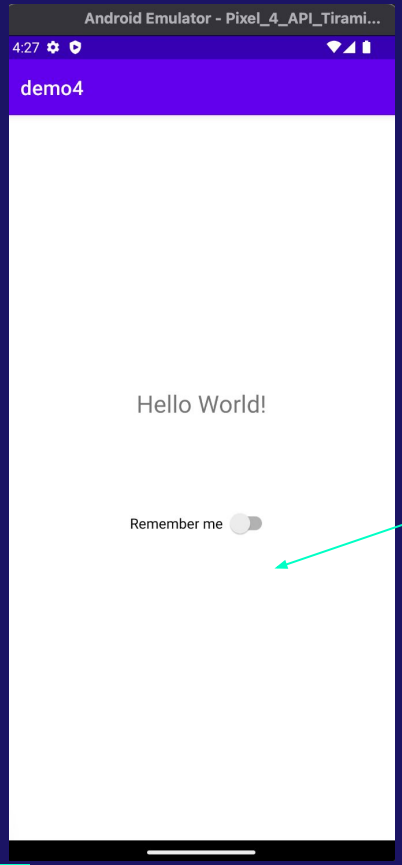
Switch Component



Om ni vill att den ska börja som ‘gömd’ så går det att ställa in via XML.

Eller programmatistk!

Switch Component




```
val rememberMe = binding.switchRememberMe                    // Switch
var rememberMeIsChecked = rememberMe.isChecked.toString()    // isChecked
val textViewRememberMe = binding.textViewRememberMe          // TextView
textViewRememberMe.isVisible = false
```

Är den synlig? Nej

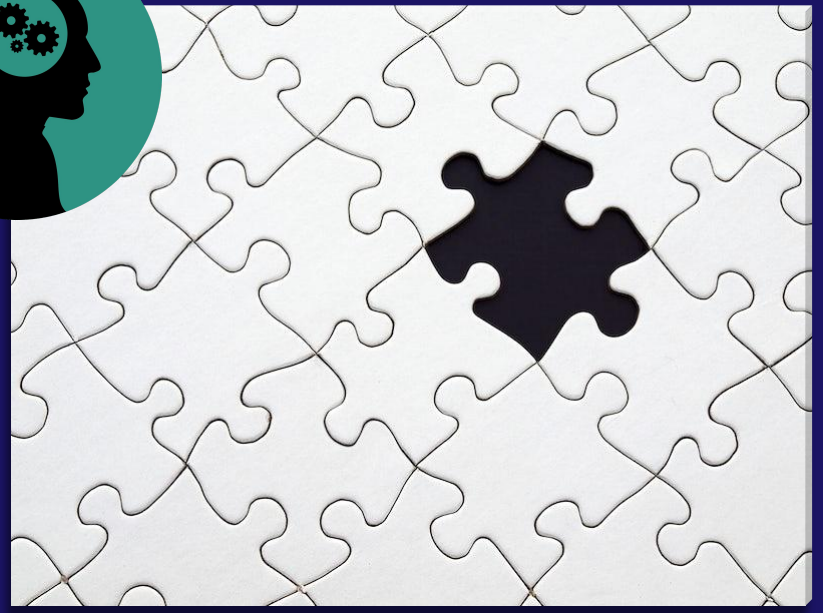
Problem

Hur tillämpar vi en Switch Komponent?

Solution

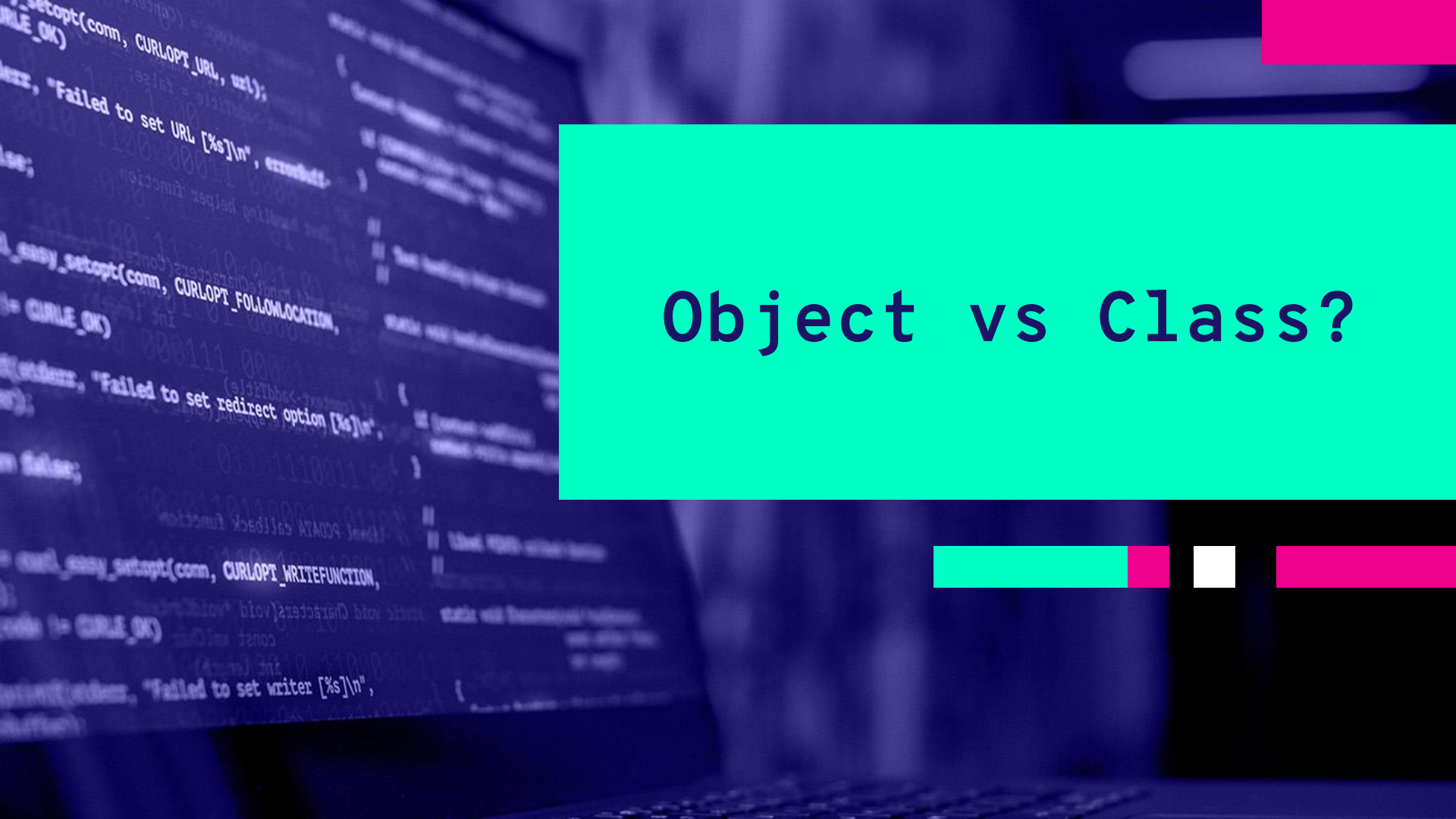
Vi kan få ut nuvarande värdet på en ‘Switch’ via metoden .isChecked. Vill vi gömma en komponent så skriver vi .isVisible = true/false

Frågor?

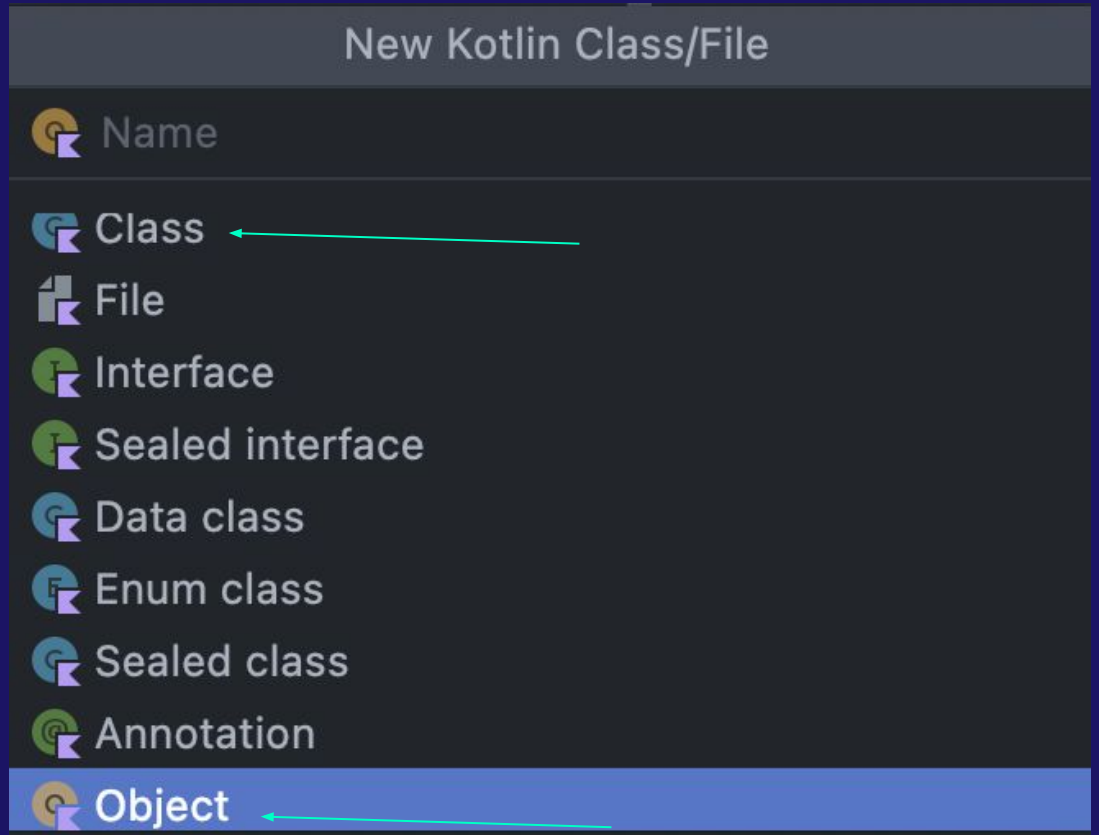


03

Objects & Classes


Object vs Class?

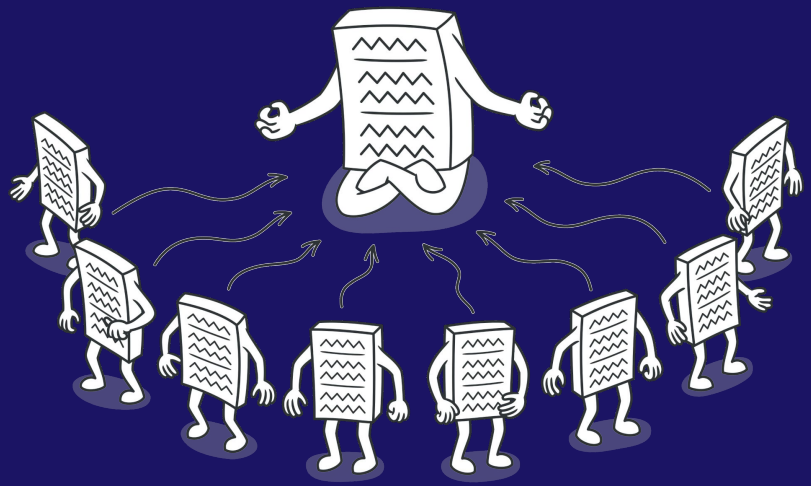
Object VS Class?



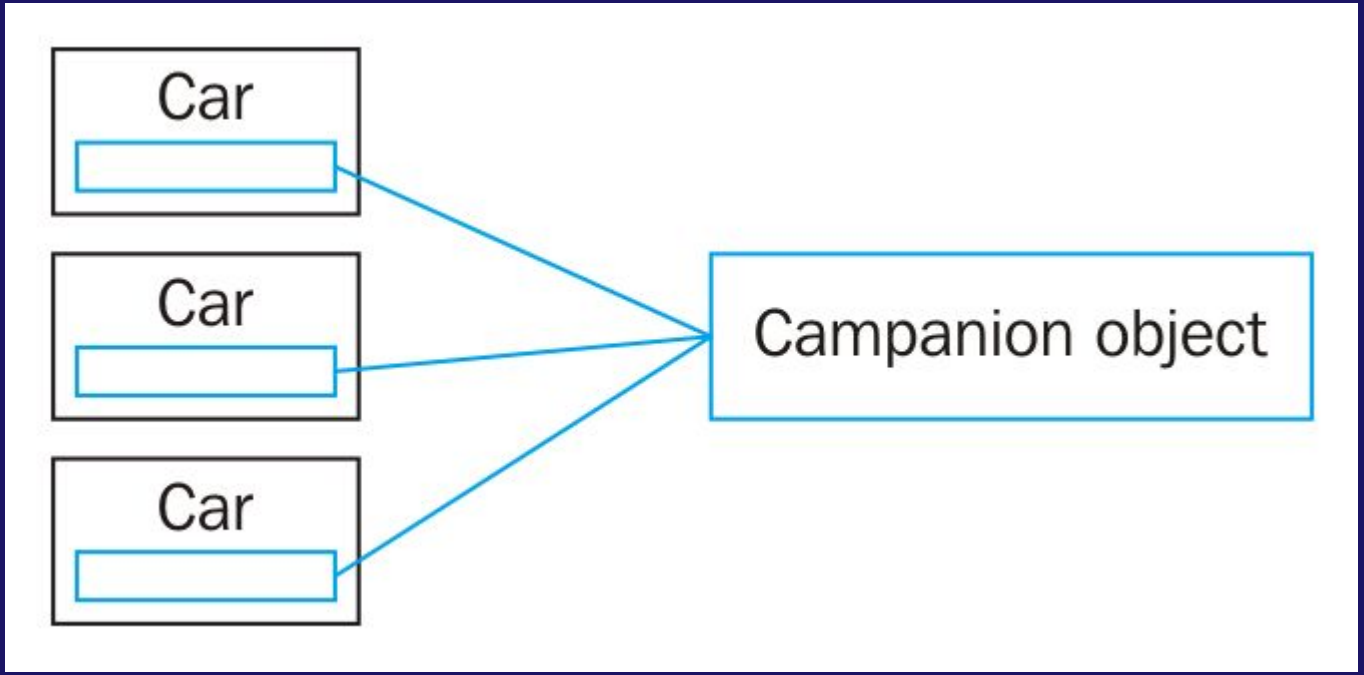
Object VS Class?

Object klasser - kan bara ha en instant (singleton pattern)

Kotlin klasser- kan ha multipla instanser



Companion Object?



Companion Object?

Ta en kalkylator klass t.ex.
I java skriver vi:

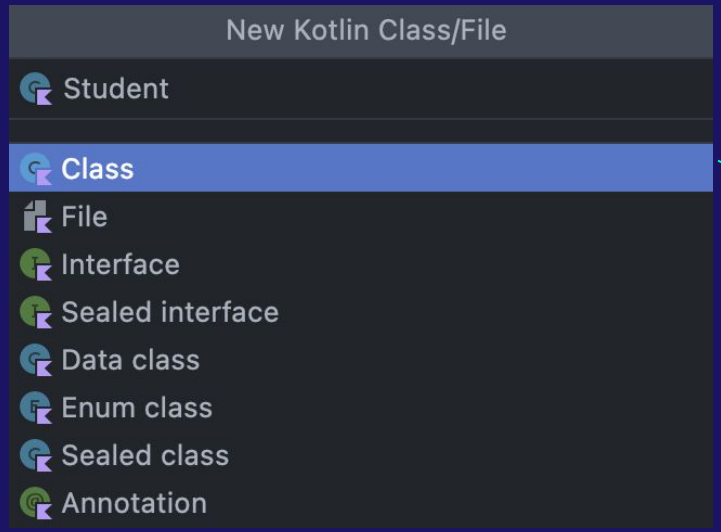
```
Calculator calculator = new Calculator()

calculator.subtract(x, y)
```

Men tänk om vi bara kunde skriva såhär:

```
Calculator.subtract(x, y)
```

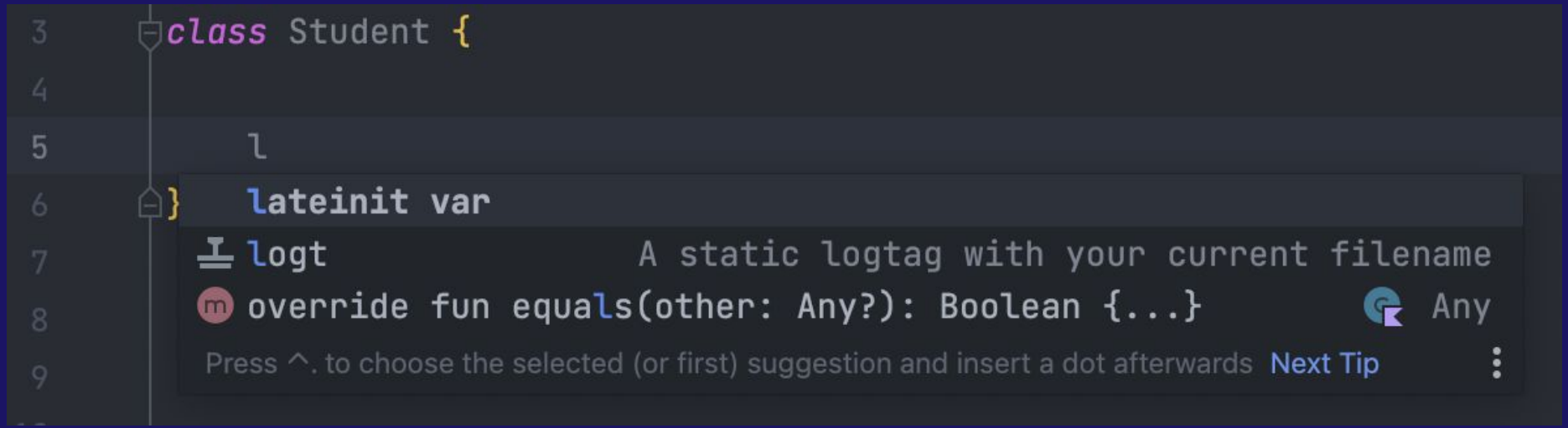
Detta är vad Kompanjon objekt tillåter oss att göra!

Skapa En helt ny klass!




```
1  package com.example.demo4
2
3  class Student {
4  }
5
```

Skapa En helt ny klass!



Sätt värde senare…

```
3    class Student {
4
5        lateinit var name: String
```

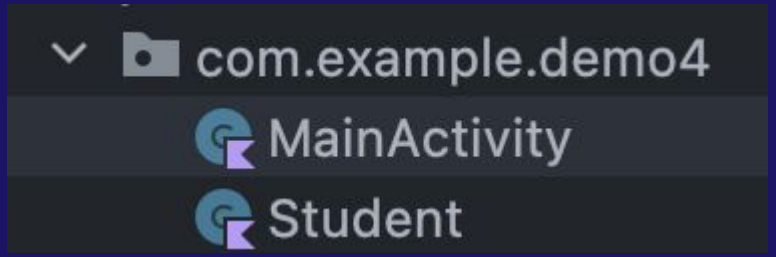
Men...

```
3   class Student {
4
5       lateinit var name: String
6       lateinit var age: Int
7       lateinit var state: Boolean
8       lateinit var size: Byte
9
10  }
11
```

Enkel lösning

```
3    class Student {
4
5        var name: String = ""
6        var age: Int = 0
7        var examined: Boolean = false
8
9    }
```

Skapa en instans!



Instansiering

```
val benny = Student()
```

Java ekvivalent mot
Student benny = new Student()

Constructor…?

```
val benny = Student("name")
```

Constructor

Primary Constructor

Om vi vill passa in argument ‘namn’ som en Sträng parameter, så måste vi först tala om för klassen att vi vill passa in ett direkt värde!

```
val benny = Student("name")
```

Constructor

Primary Constructor

Märk av att istället för att ge dessa ett värde på 0, så har vi nu en konstruktor med dessa värden!

```
package com.example.demo4

class Student(
        var name: String,
        var age: Int,
        var examined: Boolean
    ) {

    // Methods...

}
```

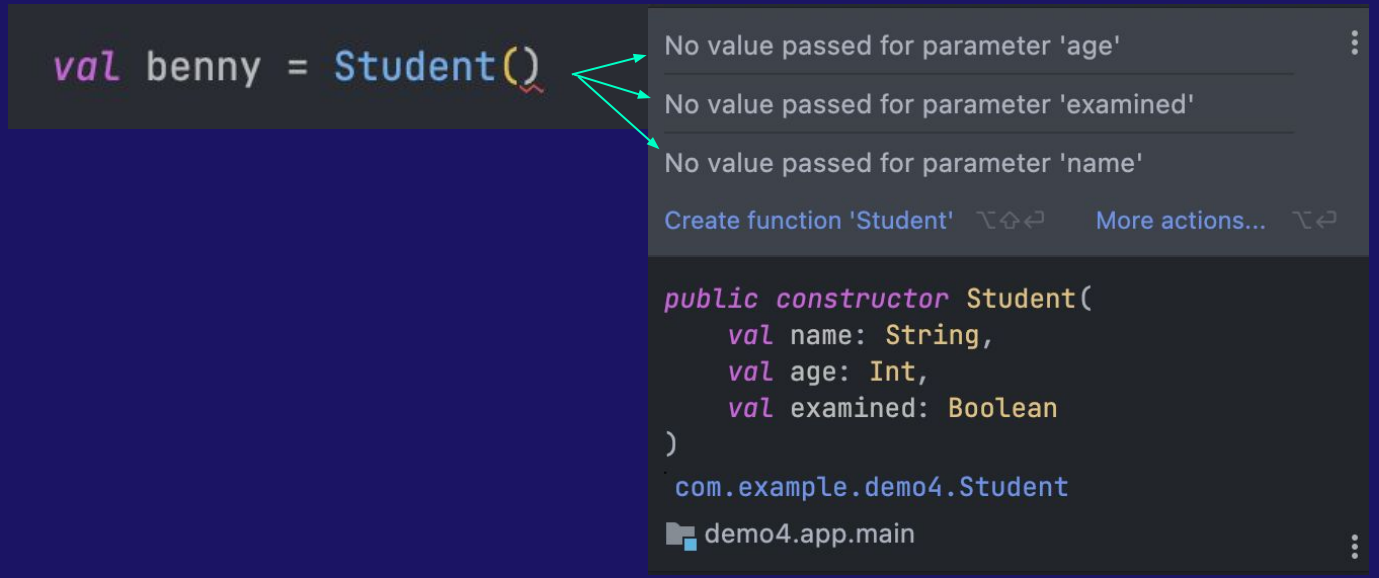
Primary Constructor?

Vi diskuterar en primär konstruktor här...
Namnet verkar då antyda på att det finns andra alternativ?

Låt oss diskutera och se skillnader och likheter!

Primary Constructor?

Med en primär konstruktor så får vi ett error - vi MÅSTE passa in värden!



Optional Constructor?

Tänk om vi kunde ha med ‘default värden från scratch men också få lov att stoppa in ett nytt värde via konstruktorn..

Detta kallas ‘Optional Constructor’

```
Benny = Student()
```

```
Benny = Student(age: 25)
```

Optional Constructor

Optional Constructor

Detta är i princip samma kod!

Fast här definierar vi ett ‘default’ värde!

```
package com.example.demo4

class Student(
        var name: String = "",
        var age: Int = 0,
        var examined: Boolean = false
    ) {

    // Methods...

}
```

Optional Constructor?

IN ACTION

```
val benny = Student()
val benny2 = Student( name: "Test",  age: 0,  examined: true)
```

List of objects?

Icke dynamisk array - kan ej växa i storlek

```
var studentList = arrayOf(
    Student( name: "Benny",  age: 15,  examined: false),
    Student( name: "Anton",  age: 14,  examined: false),
    Student( name: "Nina",  age: 18,  examined: true),
)
```

List of objects?

Dynamisk Array - kan växa i storlek (det går att exkludera <student> här

```
var studentList = ArrayList<Student>(
    listOf(
        Student( name: "Benny",  age: 15,  examined: false),
        Student( name: "Anton",  age: 14,  examined: false),
        Student( name: "Nina",  age: 18,  examined: true),
    )
)
```

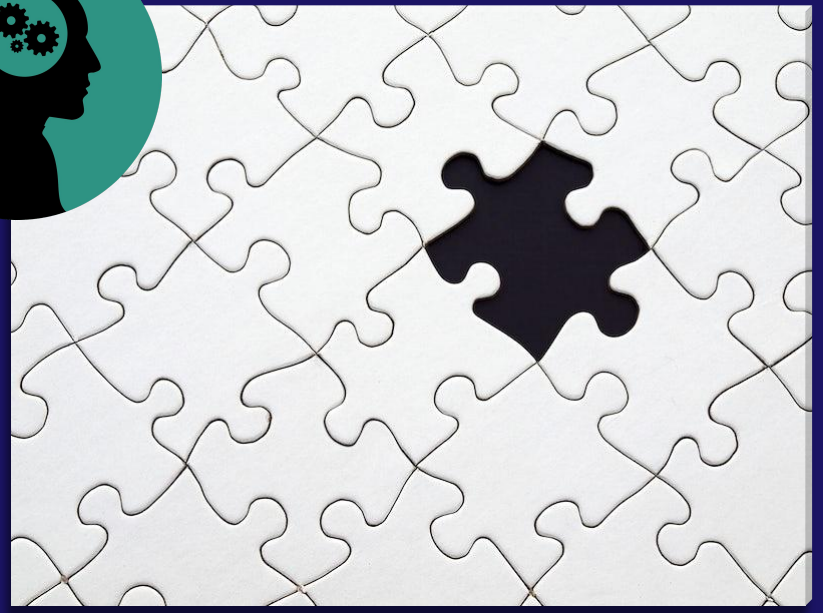
Problem

Konstruktor och klasser, hur tillämpar vi dessa?

Solution

Skapa ny kotlin klass, för att få en konstruktor så kan du lägga till en parentes på klassnivå och definiera variabler som parameter!

Frågor?



04

Uppgifter & Eget Arbete

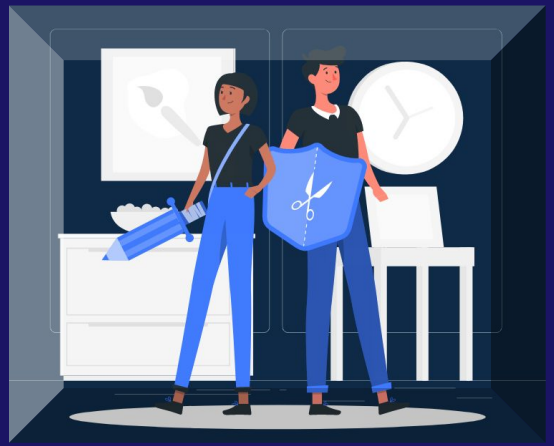
Uppgifter

Välkommen till första uppgiften!

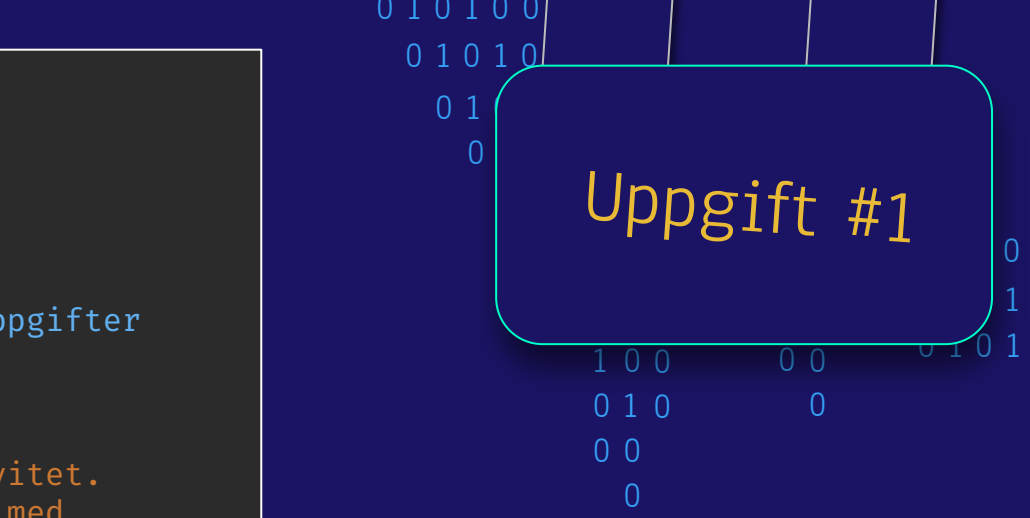
Uppgifterna är till för att testa dina färdigheter och kunskaper för att både öva och repetera på det vi har arbetat med under föreläsningarna.

Dessa är **INTE** obligatoriska.

Men är starkt rekommenderat att arbeta med.



```
1   // –Uppgift #1– //
2
3   /* INSTRUCTIONS
4
5       Skapa ett nytt projekt!
6
7       Döp projektet till: Lektion_4_uppgifter
8
9
10      Navigera till din main.xml aktivitet.
11      Lägg till en 'Switch' komponent med
12      'Constraints'.
13
14      Lägg till ett relevant ID till din
15      Textvy + Switch
16
17  */
18
19  // HINT & Examples
20  hint("Infer constraints kan hjälpa dig komma
21  igång snabbt med design, men förlita dig inte för
22  mycket på detta verktyg - ibland kan den stöka
23  till din design!")
```



Kom igång enkelt med uppgift #1

```
1   // –Uppgift #2– //
2
3   /* INSTRUCTIONS
4
5       Upprätta felmeddelandet för din ‘Switch’
6       Komponent! (minimum height)
7
8       Gör om din ‘Switch’ komponent så att den
9       förblir bakåtkompatibel med äldre Android
10      versioner!
11
12      Om man går in till .xml filen där ni har er
13      komponent liggandes, ta bort namnet och
14      skriv bara: appcompatswitch
15
16  */
17
18  // HINT & Examples
19  hint(”Bakåtkompatibilitet utökar UX, speciellt
20  för äldre enheter.
21
22  Slide #19 “)
23
```

Ett enkelt men intressant steg inom automatisk bakåtkompatibilitet!

```
1   // –Uppgift #3– //
2
3   /* INSTRUCTIONS
4
5       Skapa en ‘ViewBinding’
6
7       Ovanför ‘onCreate’, initialisera en lazyinit
8       Variabel med datatypen : ActivityMainBinding
9
10      Sedan kan du:
11
12  binding=ActivityMainBinding.inflate(layoutInflater)
13  val view = binding.root
14  setContentView(view)
15
16  */
17
18  // HINT & Examples
19  hint(”Dessa steg kan kännas otympliga och
20  förvirrande.
21
22  För kodförklaring: kolla på slide #33  “)
23
```

Inflate, binding och layoutInflater är koncept som är nytt.

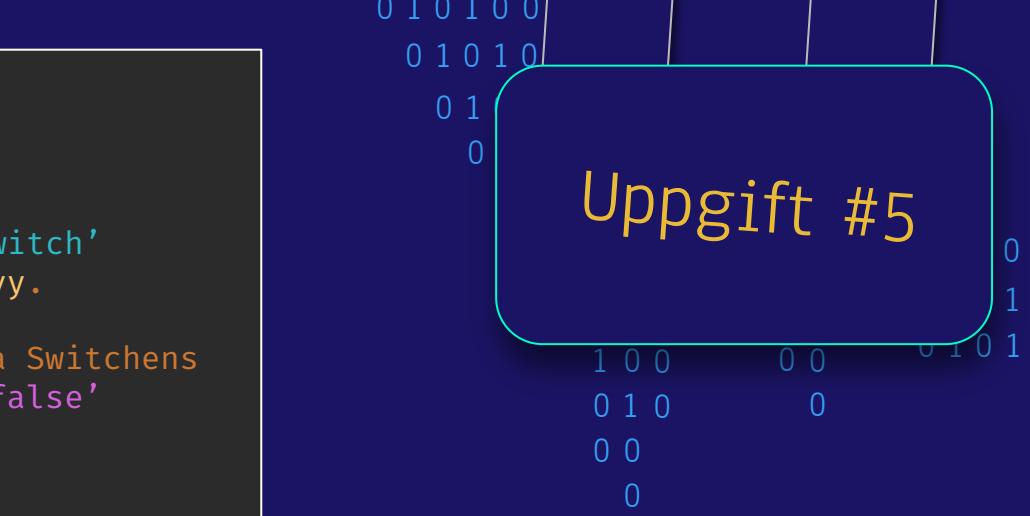
I korta drag, tillåter detta oss att exkludera intialisering av id’n med findViewById()

```
1   // –Uppgift #4– //
2
3   /* INSTRUCTIONS
4
5        Med ‘binder’ på plats, åkalla nu via
6        ‘bindern’, vår ‘switch’!
7
8        Skapa alltså en setOnClickListener()
9
10  */
11
12  // HINT & Examples
13  hint(”binding.switchID.setonclicklistener
14  Dessa blir snabbt väldigt långa rader kod.
15
16  Skapa istället en variabel t.ex:
17
18  val rememberMe = binding.switchRememberMe
19
20   “)
21
22
23
```

Istället för att behöva definiera multipla id’n så har vi gjort det på ett lite mer sofistikerat sätt.

Vi undkommer dock inte långa variabelnamn och deklarationer…

```
1   // -Uppgift #5- //
2
3   /* INSTRUCTIONS
4
5       Varje gång vi trycker på vår 'Switch'
6       så skall vi nu ändra på en textvy.
7
8       Textvy'n skall alltså reflektera Switchens
9       nuvarande 'state' t.ex 'true / false'
10
11
12  */
13
14  // HINT & Examples
15  hint(" component.text = "new Text" sätter en ny
16  text på en kopmonent!
17
18  .isChecked() ← hämtar 'switchen's nuvarande
19  status!
20   ")
21
22
23
```



Nu övar vi på setOnClickListener!

Glöm inte att detta skall hända varje gång vi TRYCKER!

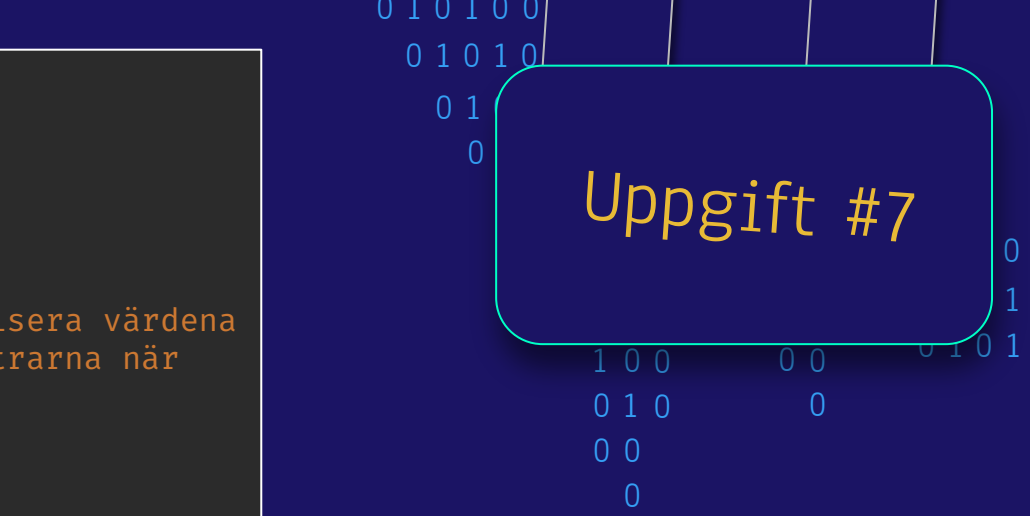
Annars finns risken att den ej uppdateras korrekt!

```
1   // -Uppgift #6- //
2
3   /* INSTRUCTIONS
4
5       Skapa en ny klass som har följande variabler
6
7         +   name : String
8         +   age : Int
9         +   isTired : Boolean
10        +   grade : Char
11
12      Klassen kan heta 'Student'
13  */
14
15  // HINT & Examples
16  hint(" Var säker på att du EXKLUDERAR
17  konstruktorn på denna uppgift!
18  ")
19
20
21
22
23
```

Skapandet av en ny klass är ett fräscht antågande.
Detta blir en förberedelse inför nästa uppgift!

```
1   // –Uppgift #6.5– //
2
3   /* INSTRUCTIONS
4
5        Skapa en loop av något slag som:
6          +   Itererar 5 gånger
7          +   Skapar ett nytt Student() objekt
8          +   Sätter ett värde för varje variabel
9              (värdet behöver ej vara unikt)
10         +   Lägg in det nya objektet inom en Array
11
12  */
13
14  // HINT & Examples
15  hint(" Var säker på att du EXKLUDERAR
16  konstruktorn på denna uppgift!
17  ")
18
19
20
21
22
23
```

```
1           // –Uppgift #7– //
2
3   /* INSTRUCTIONS
4
5        Inkludera nu en konstruktor!
6
7        Gör om din for loop och initialisera värdena
8        till dina variabler inom parametrarna när
9        du instansierar objektet.
10
11  */
12
13  // HINT & Examples
14  hint(" Istället för att skriva
15
16       Object.name = new value
17       Object.age = new value
18
19       Så kan du göra det inom parametrarna
20       som argument!
21
22       Exempelvist val benny = Student("name", 15 )
23
24  ")
25
```



Nu ser vi hur mycket konstruktorn underlättar åt oss!

```
1   // –Uppgift #8– //
2     THE TOUGH NUT
3   /* INSTRUCTIONS
4
5       Skapa en ‘Register’ aktivitet som sparar ner
6       användare inom en array!
7
8       Aktiviteten inkluderar alltså input för:
9         +   name       - String Input
10        +   age        - Numerisk Input
11        +   rememberMe - Switch / Checkbox
12
13      Här är datatyper väldigt viktigt!
14
15      Denna arrayen skall sedan passas vidare
16      In till ‘login’ page där man sedan skall
17      kunna logga in!
18
19  */
20
21  // HINT & Examples
22  hint(”Utgå från tidigare uppgifter när det kommer
23  till input och sparandet av en användare.“)
```

En kombination av kunskaper krävs för att lyckas med denna uppgift!

Klasser, Intents, Id’n, komponenter, inputs, object & arrayer såväl som felhantering logik för att kolla OM användaren skrivit in rätt!

THANKS !

Do you have any questions?
kristoffer.johansson@sti.se

sti.learning.nu/

You can also contact me VIA Teams (quicker response)
Du kan också kontakta mig VIA Teams (Snabbare svar)