



Sovellusten ohjelmointi ja käytettävyys Oppimispäiväkirja

Jaakko Kuusisto

Sisällysluettelo

[1 Viikkoharjoitukset 3](#_Toc177904951)

[1.1 Android -ympäristön asennus ja Hello World 3](#_Toc177904952)

[1.2 Jetpack Compose -tutustuminen 3](#_Toc177904953)

[1.3 Kotlin essentials – osa 1 5](#_Toc177904954)

[2 Viikkotehtävät 8](#_Toc177904955)

[2.1 Valuuttamuuntimen käyttöliittymä 8](#_Toc177904956)

[2.2 Sääsovelluksen käyttöliittymä 8](#_Toc177904957)

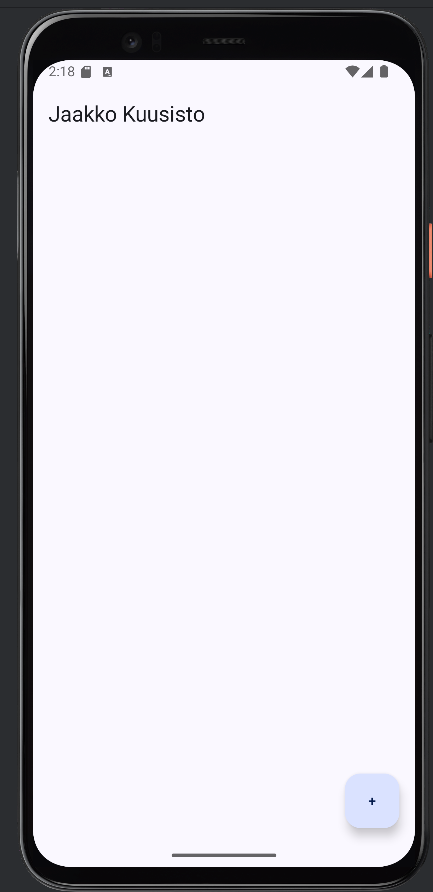
[3 Käytetyt lähteet 9](#_Toc177904958)

# Viikkoharjoitukset

## Android -ympäristön asennus ja Hello World

Android Studio asentui lataamalla Installer Wizard ja perusasetuksilla. Jossain kohtaa Installer jumittui ladatessaan palvelimelta jotain tiedostoa. Nopealla googletamisella tämä vaikutti olevan 10+ vuotta vanha ongelma, mutta ratkesi lataamalla manuaalisesti klikkaamalla tiettyä Android Studion tiedostoa.

 Projektin luonti onnistui hyvin käyttämällä Empty Activity -pohjaa. Projektia luodessa ja ohjelmaa kääntäessä Windows Defender kysyi hyväksyntää useaan otteeseen, mutta antaessa oikeudet, ei Studion käytössä ollut ongelmia.



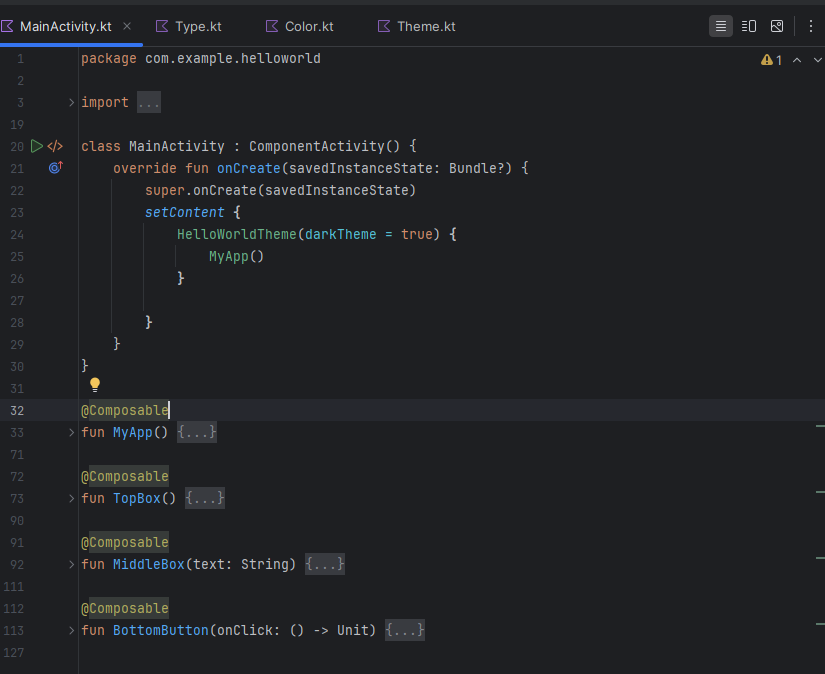
HelloWorld-projekti onnistui muokkaamalla esimerkkikoodia ja lisäämällä Scaffold()-funktioon TopBar() ja siihen teksti.

## Jetpack Compose -tutustuminen

Annotaatiot ovat Kotlinissa/Javassa tapa liittää metadataa koodin osiin, kuten luokkiin tai funktioihin jne.

@Preview on annotaatio, jota käytetään Jetpack Compose -kehityksessä esikatselun mahdollistamiseen. Tällöin IDE:llä voidaan esikatsella komponenttia ilman, että se ajetaan emulaattorissa.

@Composable on annotaatio, joka merkitsee funktiota, joka luo käyttöliittymäelementtejä Jetpack Composessa. @Composable voidaan kutsua vain toisesta @Composable-funktiosta ja näitä yhdistelemällä luodaan käyttöliittymä.

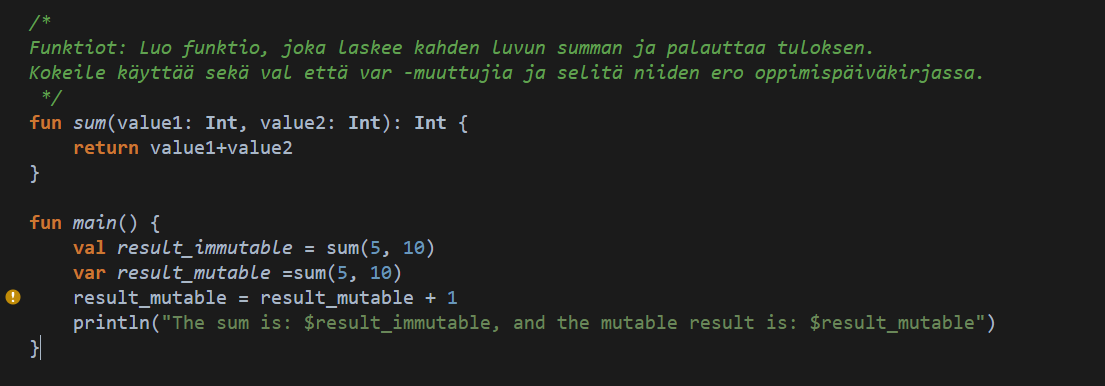


A screen shot of a cell phone

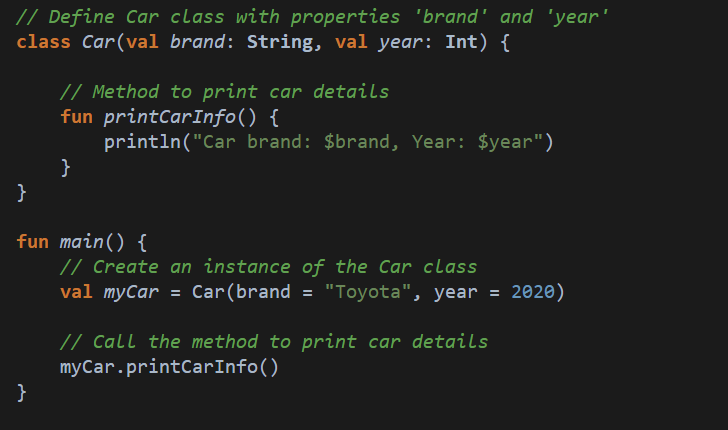
Description automatically generated

Kuvissa Android-sovellus, jossa komponenteille alifunktiot ja noudattaa Jetpack Composen väri- ja typografiateemoja.

## Kotlin essentials – osa 1



Var tarkoittaa muuttujaa ja val tarkoittaa muuttumatonta arvoa.

Null safety tarkoittaa, että Kotlinissa pyritään estämään null-viittausvirheet, jotka voivat aiheuttaa ohjelman kaatumisen. Tämä tehdään siten, että tyypit oletuksena eivät voi sisältää null-arvoa. Jos muuttuja voi olla null, sen tyyppi määritellään erikseen käyttämällä kysymysmerkkiä (?).

A computer screen shot of a program

Description automatically generated

Elvis-operaattori on lyhyt tapa käsitellä mahdollisesti null-arvoja. Se tarkistaa, onko muuttuja null, ja jos on, palauttaa sille määritetyn oletusarvon. Jos muuttuja ei ole null, se palauttaa sen arvon.

A screen shot of a computer

Description automatically generated

Laajennusfunktiot mahdollistavat olemassa olevien luokkien laajentamisen lisäämällä niihin uusia metodeja, muokkaamatta itse luokan koodia.

A screen shot of a computer

Description automatically generated

# Viikkotehtävät

## Valuuttamuuntimen käyttöliittymä

Tein UI:n jossa pudotusvalikot valuutoille, laatikko tulostukselle, summalle tekstikenttä ja nappi. Käyttöliittymä modulisoitu jakamalla se composable funktioihin, jotka halutessa voidaan jakaa eri tiedostoihin. Näin pienessä projektissa, en nähnyt tarvetta tähän.

Lähdekoodi: https://github.com/MikeDanton/5G00EV05-3004\_Sovellusten\_ohjelmointi\_ja\_kaytettavyys/tree/main/week2/CurrencyConverter  
A screenshot of a phone

Description automatically generated

## Sääsovelluksen käyttöliittymä

# Käytetyt lähteet

https://chatgpt.com  
https://kotlinlang.org/docs/home.html