



Sovellusten ohjelmointi ja käytettävyys Oppimispäiväkirja

Jaakko Kuusisto

Sisällysluettelo

[1 Viikkoharjoitukset 3](#_Toc178018511)

[1.1 Android -ympäristön asennus ja Hello World 3](#_Toc178018512)

[1.2 Jetpack Compose -tutustuminen 3](#_Toc178018513)

[1.3 Kotlin essentials – osa 1 5](#_Toc178018514)

[2 Viikkotehtävät 8](#_Toc178018515)

[2.1 Valuuttamuuntimen käyttöliittymä 8](#_Toc178018516)

[2.2 Sääsovelluksen käyttöliittymä 8](#_Toc178018517)

[2.3 Scaffold ja sen käyttö käyttöliittymässä 9](#_Toc178018518)

[2.4 Kotlin -harjoituksia 10](#_Toc178018519)

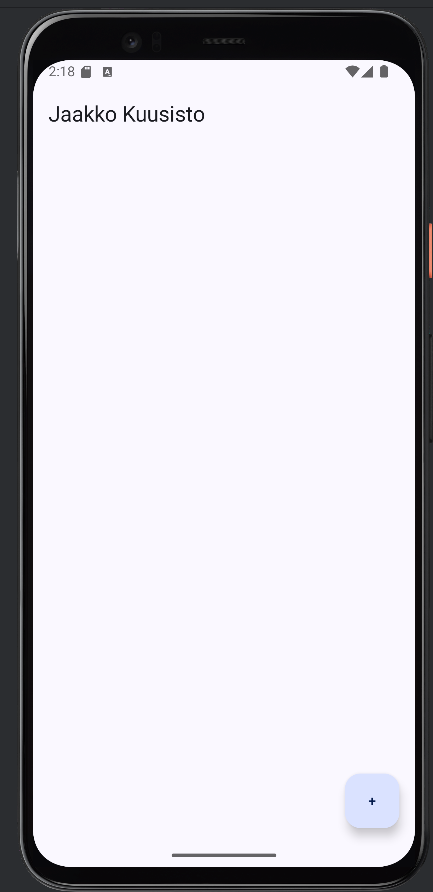
[3 Käytetyt lähteet 11](#_Toc178018520)

# 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.



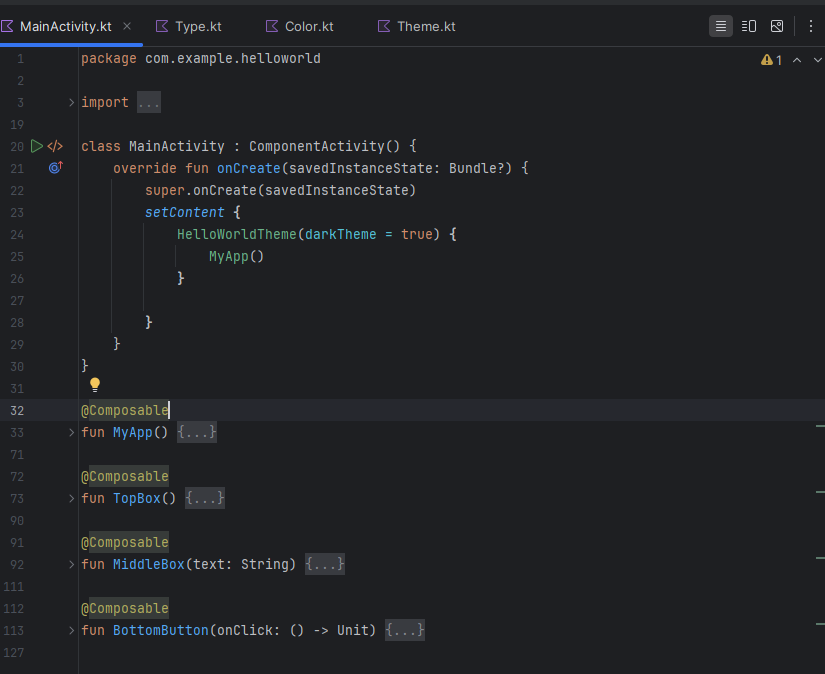
HelloWorld-projekti onnistui muokkaamalla esimerkkikoodia ja lisäämällä Scaffold()-funktioon TopBar() ja siihen teksti.

## 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.

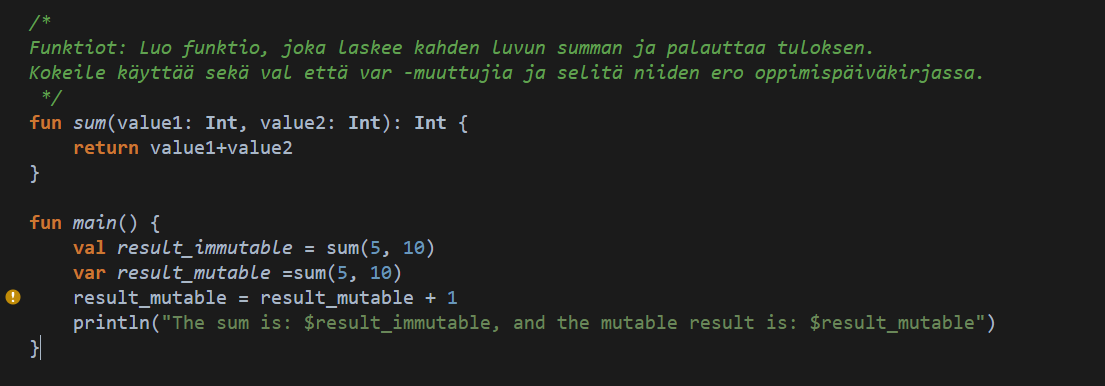


A screen shot of a cell phone

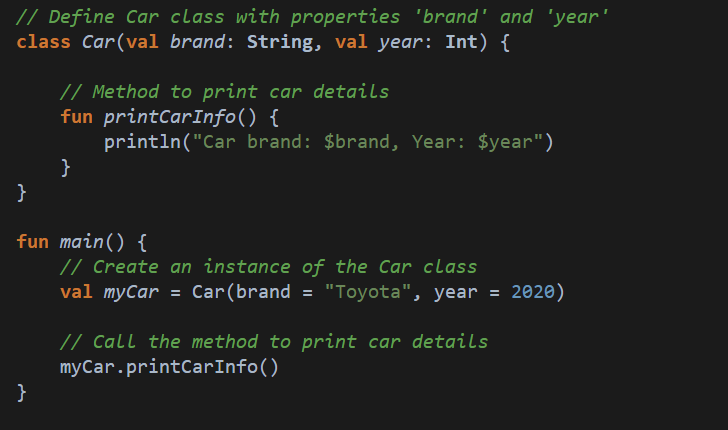
Description automatically generated

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

# Viikkoharjoitukset

## 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

Toteutettiin UI tehtävänmukaiselle sovellukselle jossa erilaisia komponentteja ja omina resursseina kaksi vektorigrafiikkatiedostoa konvertoituna androidin käyttämään tiedostomuotoon.  
Lähdekoodi: https://github.com/MikeDanton/5G00EV05-3004\_Sovellusten\_ohjelmointi\_ja\_kaytettavyys/tree/main/week2/WeatherApp

A screen shot of a phone

Description automatically generated

## Scaffold ja sen käyttö käyttöliittymässä

Tämä tehtävä oli yksinkertainen, koska olen käyttänyt Scaffoldia jo kaikissa aikaisemmikin projekteissa.

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

Description automatically generated

## Kotlin -harjoituksia

A screenshot of a computer program

Description automatically generated

IntelliJ IDEA:lla demottu ListOf() ja MutableListOf(). Kuten nimi kuvailee, ensimmäinen on muuttumaton, sen arvoja ei voi muuttaa tai lisätä siihen. Muuttuvaan listaan voidaan näin tehdä.

Lähdekoodi: <https://github.com/MikeDanton/5G00EV05-3004_Sovellusten_ohjelmointi_ja_kaytettavyys/tree/main/week2/kotlinHarjoituksia/src/main/kotlin>

When -avainsana on kotlinin vastine switch-case:lle, mutta tukee myös erilaisia tietotyyppejä ja sen takia joustava. ChatGPT kertoo suorituskyvyn vastaavan C++ switch-case:a perustietotyypeillä, joten se on myös tehokas.

# Viikkoharjoitukset

## Lokalisointi

Lokalisointi on hyvä hoitaa resurssitiedostoissa koodin sijaan useista eri syistä. Tavallisiin ohjelmointiperiaatteisiin kuuluu koodin tekeminen modulaariseksi, loogisten ja tekstikomponenttien erottaminen, aivan samalla kuin funktioiden erottaminen yhden tehtävän perusteella. Tämä myös johtaa helpompaan koodin ylläpitoon ja myös ohjelmointia osaamaton henkilö voi tehdä käännöksiä helposti. A screen shot of a phone

Description automatically generatedA black and white sign with a star

Description automatically generated

Kuvissa tehtävän sovellus oletus- (englanti tässä) ja suomenkieliasetuksilla. Kieli ja teema-asetuksia voi vaihtaa emulaattorin pikavalikosta.

## Teemat

Teemoilla on samat hyödyt kuin lokalisoinnilla. Modulaarisuus, loogisten ja ulkoasukomponenttien erottaminen. Koodin ylläpito jne. Teemoilla määritellään eri komponenttien värejä ja tekstin fonttiasetuksia.

A screen shot of a cell phone

Description automatically generatedA close-up of a certificate

Description automatically generated

A screen shot of a computer

Description automatically generated

Kuvissa eri kaksi moodia, satunnaisesti valitusta theme builder tiedostosta ja custom- väriteemoja. Sovelluksen alanurkan napista väriteema vaihtuu.

## Sovelluksen tila ja toiminnallisuus

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generated

Kuvissa polttoainekustannuslaskin kahdella eri lokalisaatiolla ja dark/light modella.

Käyttöliittymän tila määrittää, miltä sovelluksen käyttöliittymä näyttää ja miten se käyttäytyy tietyllä hetkellä. Esimerkiksi valittu teema tai mitkä elementit ovat näkyvillä.

Remember on Jetpack Composen funktio, jonka avulla voidaan säilyttää muuttujan arvo komposition avulla. Tätä voidaan käyttää tilojen säilyttämiseen.

RememberSaveable on samankaltainen, mutta säilyttää arvon silloinkin, kun sovellus on taustalla.

# Viikkoharjoitukset 4 Navigointi, Androidin Intent -mekanismi

## Harjoitus 1: Navigointi Jetpack Composessa

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generated

Tässä demosovelluksessa käytettiin Androidin NavHost -Composable funktiota ja NavController -luokkaa toteutuksessa siirtyä eri näkymiin. NavController vastaa navigoinnista ja Composable -funktio logiikasta.

## Harjoitus 2: Bottom Tabs -navigointi

## Tehtävä 3: Androidin Intent mekanismi

# Käytetyt lähteet

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