Jaakko Palm

Kotlinin ja Javan vertaus androidin kehityksessä

Tietotekniikan kandidaatintutkielma

14. kesäkuuta 2022

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Jaakko Palm

Yhteystiedot: jaakkop1998@gmail.com

Ohjaaja: Jonne Itkonen

Työn nimi: Kotlinin ja Javan vertaus androidin kehityksessä

Title in English: Kotlin versus Java in Android Applications

Työ: Kandidaatintutkielma

Opintosuunta: Informaatioteknologian tiedekunta

Sivumäärä: 22+0

Tiivistelmä: Vuonna 2017 Google otti Kotlinin viralliseksi kieleksi Androidin kehitykseen. Kotlinin suosio on kasvanut viimevuosina tästä johtuen. Perinteisen Javan ja Kotlinin vertauksesta on tehty useita tutkimuksia. Tämän tutkielman tarkoitus on ollut kartoittaa ja esitellä kielten erot toisistaan, että kyettäisiin päättelemään onko Kotlin vakavasti otettava vaihtoehto Android-kehityksessä. Lopputuloksena Kotlin antaa itsestään vahvan vaikutelman, Kotlinilla on enemmän ominaisuuksia ja on usein myös raportoitu tiiviimmäksi ja luettavammaksi kieleksi kuin Java.

Avainsanat: Android, Java, Kotlin

Abstract: In the year 2017 Google took Kotlin as their official language for Android-development. Because of this, Kotlin's popularity has been risen in the recent years. There has been many studies done between traditional Java and Kotlin. This research's purpose has been to map and show differences between two languages, so that we could deduce is Kotlin to be taken seriously as an alternative for Java. In conclusion Kotlin gives strong impression, Kotlin has more functions and is reported to be more readable and concise language than Java.

Keywords: Android, Java, Kotlin

Esipuhe

Viime kuukausina pääsin tutustumaan tarkemmin Android-sovellusten kehitykseen omassa työelämässä, jossa sitten kohtasin Kotlin ohjelmointikielen. Omaksi yllätykseksi Kotlin on kieli, jota voidaan ajaa samassa tiedostossa Javan kanssa. Tästä heräsi oma mielenkiintoni näiden kahden kielen vertailuihin, kielet kääntyvät samaksi tavukoodiksi ohjelmiston takana, mutta miten ne sitten eroavat käytännössä. Tästä sitten kehityi aihe tutkielmalle kun lähdin tutustumaan perinteisen Javan ja uudemman Kotlinin eroihin Androidin kehityksessä.

Jyväskylässä 14. kesäkuuta 2022

Jaakko Palm

Kuviot

Sisällys

1 Johdanto

Vuonna 2017 Google julisti Kotlin-ohjelmointikielen viralliseksi kieleksi Android-sovellusten kehityksessä. Kotlin on kieli josta alunperin julkaistiin vuonna 2016 versio 1.0. Googlen tuen myötä Kotlin on kasvattanut viimeaikoina suosiota nopeasti Android-kehityksessä. Suosion kasvua on myös helpottanut, että Kotlin kääntyy Java Virtual Machine-ympäristössä Javan tavukoodiksi, mikä on mahdollistanut, että Kotlinin koodia voidaan ajaa Javassa ja Javan koodia Kotlinissa. Tämä on mahdollistanut Kotlinin integroinnin jo olemassa oleviin sovelluksiin vaivattomasti, joten projektin uudelleen kirjoittaminen ei ole ollut tarpeen sovelluksissa.

Tämän paperin tähtäys on verratta näitä kahta kieltä keskenään ja määritellä syyt Kotlinin suosion kasvulle, mitkä asiat ovat johtaneet Googlen suosimaan Kotlinia perinteisen Javan sijaan Android-kehityksessä ja onko Kotlin hyvä vaihtoehto Javalle Android-sovelluksissa. Kotlinin ja Javan eroja on esimerkiksi toiminnallisuuksissa ja virhetilanteiden käsittelyissä, kuten *NullPointerException* eli nolla-arvon viittauksessa. Tietysti syntaksieroja löytyy kielistä, mutta niistä käydään läpi vain suurimmat.

Seuraavassa kappaleessa käsitellään taustaa Javalle ja Kotlinille, avataan koodien historiaa ja niiden yhteensopivuutta. Kappaleessa käydään läpi myös Kotlinin suosion nousu ja adoptoinin ilmiöt liittyen Android-sovelluksiin, sekä Kotlinin ja Javan väliset tutkimukset. Kolmannessa kappaleessa avataan kielien eroihin käytännössä ja käydään läpi mitä lopputuloksia eri tutkimukset ovat saaneet kielien käytöstä toisiinsa verrattuina.

2 Taustaa

Java on vuonna 1995 julkaistu, luokkiin perustuva oliopohjainen koodikieli. Javalla on kolme tärkeää osaa sen ekosysteemissä (Alankus ym. 2019), Java Runtime Enviroment (JRE), Java Development Kit (JDK) ja tämän tutkielman kannalta tärkein: Java Virtual Machine (JVM). JVM on virtuaalikone joka lukee Javan tavukoodia, joksi Java kääntyy. Tämä on tärkeä tieto sillä kieli Kotlin kääntyy myös Javan tavukoodiksi mikä mahdollistaa, että Javan koodia voidaan kutsua Kotlinissa ja toisinpäin. On olemassa muitakin kieliä jotka toimivat JVM:n kautta, kuten Scala ja Clojure, mutta tässä tutkielmassa keskitytään vain Javaan ja Kotliniin. Suurin osa tutkimuksista näiden kahden kielen välillä keskittyy Android.alustalle. Tälle syynä voi olla että Kotlin on saanut Googlen virallisen tuen Androidin alustalle. Kotlin sai kaikki työkalut, jotka olivat jo Javalla, lokakuussa 2017 Android Studion versiossa 3.0 [1] käyttöön. Kotlin on nykyään oletukieli Googlen omassa Androidin kehitysalustassa Android Studiossa.

2.1 Kotlinin suosio

Martinez ja Gois Mateus (2021) mukaan lähiaikoina on ollut selkeä ilmiö, jossa Android-kehittäjät ovat alkaneet käyttämään Kotlin ohjelmoitikieltä Javan kanssa tai kokonaan korvaamaan Javan. Hecht ja Bergel (2021) tekivät tarkemman tutkimuksen adoptoinnin määrään liittyen Android-sovelluksissa ja löysivät että Kotlinin määrä Android-sovelluksissa on tasaisesti kasvanut vuodesta 2017 lähtien. Myös eri tietolähteet mainostavat Kotlinin käyttöä sovelluksissa, esimerkiksi Googlen blogin [2] mukaan 1000 suosituimmasta Android-sovelluksesta noin 60 prosentia sisältävät ainakin jonkin verran Kotlinia.

Hecht ja Bergel (2021) toteuttivat tavan erottaa Javan koodia Kotlinista tavukoodin kautta, jotta he voisivat analysoida Kotlinin käyttöä sovelluksissa, joista ei ole lähdekoodia saatavilla. Hecht ja Bergel (2021) kävivät läpi AndroZoo-nimisestä tietojoukosta (Allix ym. 2016) läpi satunnaisesti valittuna 200 tuhatta sovellusta. Hechthin ja Bergelin (Hecht ja Bergel

1. https://android-developers.googleblog.com/2017/10/android-studio-30.html, käytiin 30.4.2022
2. https://android-developers.googleblog.com/2019/12/androids-commitment-to-kotlin.html,käytiin 29.4.2022

2021) löytöjen mukaan suurin osa sovelluksista käyttää edelleen Javaa ja Kotlinin käyttöön otto on ollut yleisempää isoimmilla ja suosituimilla sovelluksilla. Tätä tukee myös AppBrain-tietojoukko [3], jossa on luokiteltu, että 164 tuhannesta sovelluksesta noin 23 prosenttia käyttää Kotlinia, mutta kun katsotaan USA:n 500 suosituimman sovelluksen tilastoa, Kotlinia löytyy jo 68 prosenttia sovelluksista. Huomautuksena on kuitenkin sanottavana, että tilastoista näkee vain, että Kotlinia on käytetty, mutta ei nähdä Kotlinin koodin käytön määrää sovelluksissa.

2.2 Kotlinin adoptointi

Martinez ja Gois Mateus (2021) mukaan Kotlinin impelmentaatio projekteihin on ollut usein yksisuuntaista, Kotlinin koodin määrä on yleensä ylittänyt Javan Android-sovelluksissa, joissa se on otettu mukaan sovelluksiin. Martinez ja Gois Mateus (2021) määrittivät kyselyillä migraation syiksi esimerkiksi uuden kielen oppimisen, Javalle ominaisten virheiden välttäminen, kuten *NullPointerException*-virhe, mahdollisuus käyttää modernia ohjelmointikieltä. Myös Android-alustan jakautuminen Javan eri versioihin on raportoitu ongelmalliseksi. Android API-versioista vain 24 (Android 7.0) uudemmat versiot voivat käyttää Javan versiota 8. Vanhemmissa versioissa on käytettävä Javan versiota 6, josta puuttuu joitain modernimpia ominaisuuksia kuten lamdalausekkeet.

Oliveira, Teixeira ja Ebert (2020) tekivät tutkimuksen Kotlinin implemennoinin vaikutuksista olemassa oleviin, Javalla kirjoitettuihin, Android-projekteihin. Heidän tekemän kyselyn mukaan suurin osa kehittäjistä totesi kielen olevan luettavampi ja tiiviimpi kuin Java, mutta Kotlinin luettavuus voi myös kärsiä jos liikakäytetään Kotlinin tiettyjä ominaisuuksia, kuten korkean kertaluokan funktioita ja lambdalausekkeita. Myös Kotlinin implemoinnissa tuli vastaan jotain ongelmia liittyen Javan ja Kotlinin yhteensopivuuteen, mutta kehittäjät kuitenkin löysivät yhteensopivuuden hyödyllisemmäksi kuin ei. Myös Android Studioon liittyviä ongelmia syntyi adoptoinnissa liittyen Android Studion käyttämään rakennustyökaluun, Gradleen ja kirjastojen päivittämiseen.

Vaikka Kotlin antaa itsestään positiivisen tilastollisen näkökulman, aiheesta löytyi yllättävän

3. https://www.appbrain.com/stats/libraries/details/kotlin/kotlin, käytiin 30.4.

vähän tutkimuksia. Tämä voi olla koska Kotlin on vielä suhteellisen uusi ilmiö Android-kehityksessä, mutta Hecht ja Bergel (2021) nostivat myös esiin Kotlinin saaman vähäisen huomion. He kävivät läpi 4 Androidiin keskittyvää konfrenssi-julkaisua (ICSE, MSR, SANER ja MOBILESoft) vuosilta 2018-2020 ja Kotlin oltiin mainittu kuudessa artikkelissa ja yksi artikkeli keskittyi Kotlinin adoptointiin (kyseinen artikkeli: (Oliveira, Teixeira ja Ebert 2020)). Seuraavassa kappaleessa käydään läpi mitä eri tutkimuksia Kotlinin ja Javan vertailusta on löytynyt, jotta saataisiin hyvä kuva miten aihetta on tutkittu tieteellisessä näkökulmasta.

2.3 Tutkimuksia Kotlinista ja Javasta

Aiemmin kappaleissa 2.1 ja 2.2 käytiin läpi Hecht ja Bergel (2021) tekemä tutkimus liittyen Kotlinin adoptoinnin määrään ja Martinez ja Gois Mateus (2021) tekemä tutkimus liittyen Kotlinin adoptoinnin syihin. Myös Kotlinin ja Javan vertauksesta on tehty useita eri tutkimuksia. Coppola, Ardito ja Torchiano (2019) ja Oliveira, Teixeira ja Ebert (2020) tutkivat Kotlinin adoptoinin vaikutuksia Android-sovelluksissa. Ardito ym. (2020) ja Bose ym. (2018) tekivät tutkielmat kielten tehokkuuksista Androidin sovellusten kehityksessä. Mateus ja Martinez (2019) vertasivat Kotlinin koodin hajuja *(code smells)* Javan koodin hajuihin, sekä vertasivat kielten koodien laatua keskenään. Gupta ja Chauhan (2021) tutkivat työkaluja koodin hajujen löytämiseen Kotlinissa ja Javassa. Mazuera-Rozo ym. (2022) kartoittivat Android-sovellusten turvallisuusriskejä ja vertasivat riskien esiintymisiä Kotlinilla kirjoitettujen ja Javalla kirjoitetujen sovellusten välillä. Peters, Scoccia ja Malavolta (2021) tutkivat Kotlinin vaikutusta sovellusten suorituskykyyn.

3 Vertaus käytännössä

Jos kehittää sovellusta vanhemmalle Androidin versiolle täytyy käyttää Javan versiota 6. Tämä aiheuttaa sen, että kehityksessä ei voida käyttää modernimpia ominaisuuksia kuten laajennusfunktioita tai lambdalausekkeita. Ardito ym. (2020) kirjoittaman artikkelin mukaan javalla on neljä yleisintä ongelmaa; *NullPointerException* (NPE), pakollinen tyyppimääritys, pitkät argumenttijonot ja dataluokkien toteutus. Bose ym. (2018) mukaan Kotlinilla on ratkaisu jokaiselle javan tunnetuimmalle ongelmalle. Saman mainitsee Kotlinin oma dokumentaatio (JetBrains 2022), jossa kerrotaan juuri Javan ongelmista jotka on otettu huomioon kielen kehityksessä. Kotlinilla on myös useita eri ominaisuuksia joita Javalla ei ole käytettävissä versiossa 6, kuten luokkien laajennusfunktiot *(extension functions)* ja vuorottelurutiinit *(coroutines)*. Näitä ominaisuuksia voidaan käyttää Android Studiossa välittämättä mille Androidin versiolle ollaan kehittämässä sovellusta. Kotlin myös tukee funktio- ja oliopohjaistaohjelmointia verrattuna Javan oliopohjaiseen ohjelmointiin.

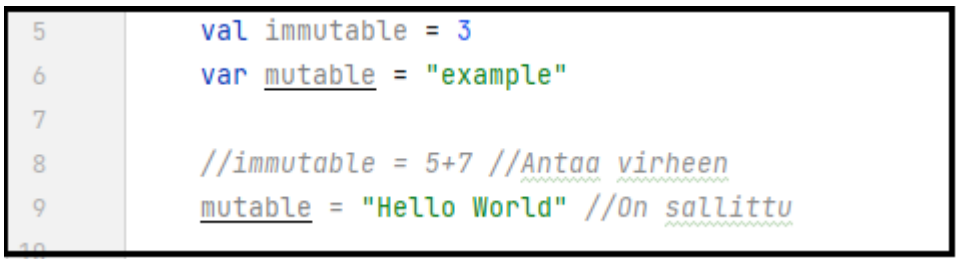
3.1 Oliopohjainen ja funktionaalinen kieli

Oliopohjaisesta kielestä puhuttuessa tarkoitetaan kieltä, joka toimii olioiden ja niiden välisen kommunikaation kautta (Poo, Kiong ja Ashok 2007). Olioiden kommunikaatio hoidetaan olioiden metodeilla, jotka ovat olioiden toiminnallisuuksia. Java on eräs tunnetuimmista oliopohjaisista kielistä. Tästä vertauksena Kotlinilla voidaan käyttää sekä oliopohjaista suunnittelua ja funktiopohjaista suunnittelua. Toisin kuin oliopohjainen kieli, funktionaalinen ohjelmointi toimii funktioiden kautta.

Kotlinin funktiot ovat ensiluokkaisia, mikä tarkoittaa, että niitä voidaan tallentaa muuttujiin tai muihin datarakennelmiin. Kotlinissa funktiot voivat olla myös korkean kertaluvun funktioita *(high-order functions)* (JetBrains 2022), eli ne ottavat parametreinä funktioita tai palauttavat funktion. Lisäksi Kotlinissa voidaan käyttää anonyymeja funktioita (JetBrains 2022), mikä on funktio jolle ei tarvitse määritellä palautuksen tyyppiä.

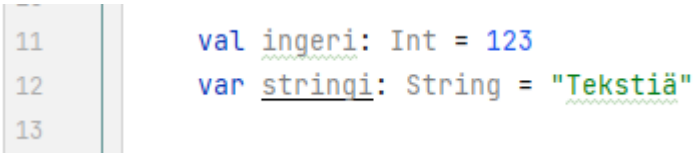
3.2 Muuttujat

Kotlinissa muuttujat luodaan kahdella avainsanalla (kuvio 1.), *val* ja *var*. Avainsanalla *val* luodut muuttujat ovat muuttumattomia *(immutable)*, eli niiden arvoa ei voida uudelleen sijoittaa koodissa. Toisin kuin *var*, joka on muuttuva *(mutable)* muuttuja, jonka arvo voidaan uudelleen sijoittaa.



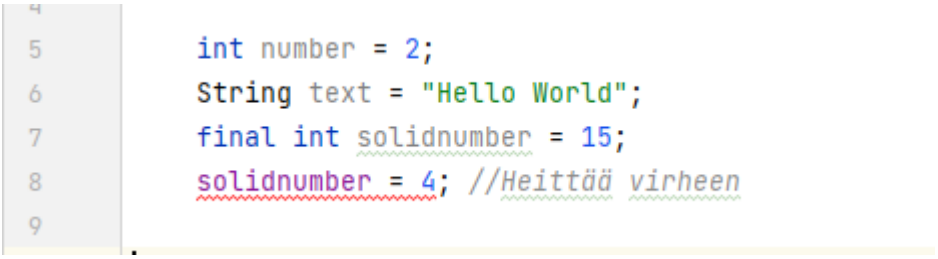
Kuvio 1. Esimerkki val ja var muuttujista

Toinen ominaisuus Kotlinin muuttujissa verrattuna Javaan on myös näkyvillä kuviossa 1. Kyseessä on automaattinen tyypin anto muuttujan arvolle. Javassa muuttujan arvon tyyppi annetaan muuttujan yhteydessä, mutta Kotlinissa arvon tyyppi määritellään automaattisesti. Kotlinissa tyyppi voidaan kuitenkin manuaalisesti määritellä seuraavin tavoin (kuvio 2).



Kuvio 2. Esimerkki Kotlinin muuttujien arvojen tyyppin määrittelystä

Vertauksena Javassa muuttujan luonnin yhteydessä määritellään muuttujalle aina tyyppi, esimerkiksi Int, Double tai String (kuvio 3). Javassa voidaan määritellä myös muuttumaton muuttuja avainsanalla *final* (kuvio 3).



Kuvio 3. Esimerkki Javan muuttujien luonnista

3.3 Eroja ominaisuuksissa

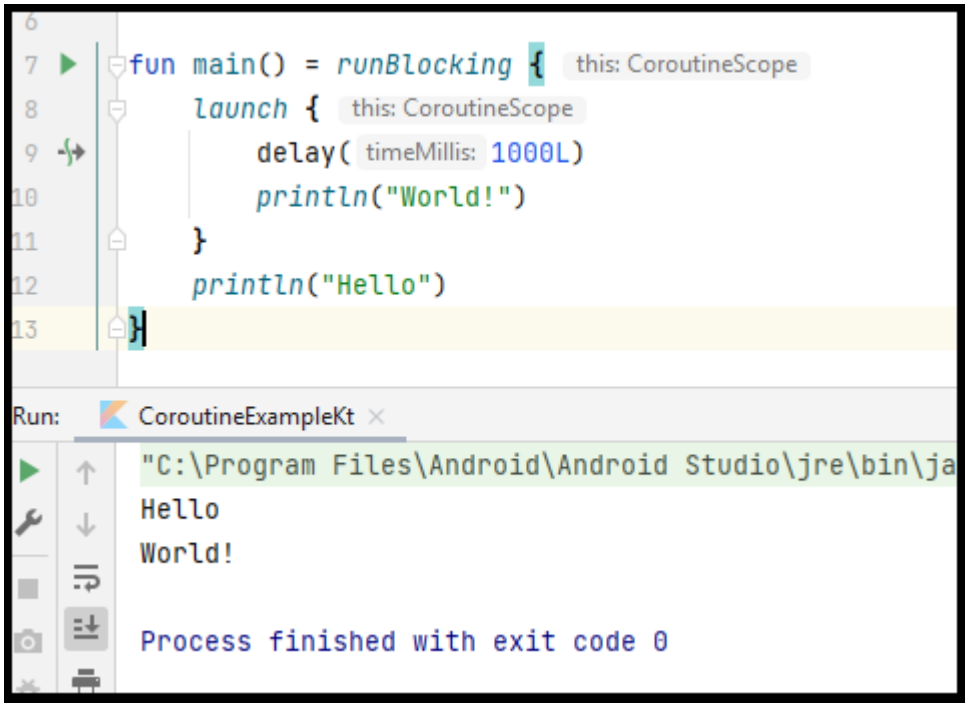
Kotlinilla on eräitä ominaisuuksia joita ei ole saatavilla suoraan Javalla[1], kuten luokkien laajennusfunktio (*extension functions*) ja vuorottelurutiinit (*coroutines*). Usein ominaisuuksia kuitenkin lisätään kehityksessä kielessä kuin kielessä ja Javalle on lisättävissä ominaisuuksia kolmannen osapuolten kirjastojen kautta. Tämä vertaus keskittyy enemmän virallisiin kirjastoihin ja pyritty huomioimaan Javan versioiden jakautuminen eri Android-käyttöjärjestelmien välillä. Vanhemmat Androidin versiot joutuvat käyttämään Javan versiota 6.

Kotlinissa voidaan käyttää lambdalausekkeita. Lambdalauseke on kuin metodi tai funktio, joka ottaa vastaan parametrit ja palauttaa tuloksen, mutta ei tarvitse nimeä ja voidaan vapaasti implemoida esimerkiksi metodin sisällä. Lambdalausekkeet ovat myös käytössä Javalla versioista 8 eteenpäin, mikä ei ole mahdollista käyttää vanhemmille Androidin versioille.

Luokkien laajennus voidaan Kotlinilla toteuttaa suoraan laajennusfunktiolla *(extension functions)*. Tällä voidaan luokan ominaisuuksia lisätä suoraan luokkaan itsessään. Vertauksena Javassa luokkaa laajennetaan perimällä luokan ominaisuudet lapsiluokalle, jolle sitten lisätään uudet ominaisuudet.

Dataluokat eli luokat joiden tehtävänä on vain ylläpitää dataa on saatavilla Kotlinin peruskirjastossa. Dataluokat luodaan Kotlinissa kuten normaalit luokat [2]. Javalla ei voida dataluokkia toteuttaa suoraan, mutta Ardito ym. (2020) tekemän kyselyn mukaan Javalle on dataluokat kuitenkin helposti toteuttettavissa Javalla, mutta eivät ole yhtä yksinkertaisia kuin Kotlinissa.

1. https://kotlinlang.org/docs/comparison-to-java.html#some-java-issues-addressed-in-kotlin, käytiin 26.3.2022
2. https://kotlinlang.org/docs/data-classes.html, käytiin 5.4.2022



Kuvio 4. Esimerkki vuorottelurutiinin käytöstä Kotlinilla Android Studiossa

Kotlinissa on myös toteutettu vuorottelurutiinit *(coroutines)*, asynkronisten operaatioiden *(asynchronous tasks)* sijaan. Vuorottelurutiini on tapa toteuttaa asynkronista koodia, toisin sanoen koodipätkän suoritus ei pysyätä ohjelmaa vaan suorittaa koodin samaan aikaan muun koodin kanssa(kuvio 4). Martinez ja Gois Mateus (2021) kyselyn mukaan kehittäjät ovat todenneet vuorottelurutiinien olevan asynkronisten operaatioihin verrattuna yksinkertaisempi käyttää. Vuorottelurutiineilla voidaan hoitaa esimerkiksi HTTP pyyntöjä.

Kotlinilla on myös *Null Safety* ja *Smart Cast* ominaisuudet *NullPointerException* errorien välttämiseksi, joista käydään seuraavassa kappaleessa läpi.

3.4 NullPointerException eli nolla-arvoisen muuttujan viittaus

Javan *NullPointerException* on tapaus jossa nolla-arvoiselle *(null)* objektille yritetään kutsua metodia, mikä johtaa ohjelman kaatuumiseen. Coelho ym. (2017) kävivät läpi yhteensä 639 Android sovellusta GitHubista ja tutkimuksen mukaan *NullPointerException* aiheutti melkein 30 prosenttia ohjelmien kaatumisista.

Vertauksena Kotlinissa on nullability-ominaisuus muuttujilla, mikä tarkoittaa että muuttuja voi olla viittaus nolla-arvoon, mutta esimerkiksi Kotlin estää metodien kutsumisen kyseisellä objektilla. Nolla-arvoiseen objektiin viitataan ?-merkillä tyypin yhteydessä (kuvio 5). Tämä on smart-cast ominaisuus Kotlinissa, jossa kääntäjä tekee automaattisesti tarkistuksen nolla-arvolle (Samuel ja Bocutiu 2017) ennen ohjelman kääntämistä. Kotlinin dokumentaation mukaan (JetBrains 2022) yksi kielen tehtävistä on ollut vähentää NPE-virheiden esiintymisten määrää.

```
15    var notsafe: String = "This is not nullable variable"
16    var safe: String? = "This is nullable variable"
17    notsafe = null //Antaa virheen
18    safe = null //On sallittu
19
```

Kuvio 5. Esimerkki Null-muuttujan luonnista Kotlinilla

3.5 Kirjoitetun koodin määrä

Kotlinin oma dokumentaatio (JetBrains 2022) mainostaa, että Kotlin tuottaa vähemmän koodia kuin Java. Tästä löytyy myös takeita eri tutkimuksissa: Martinez ja Gois Mateus (2021) kysyivät avoimen lähteen Android-sovellusten 98 kehittäjältä kokemuksia koodin laadusta ja vastaajista 34 vastasi, että Kotlinissa tarvitaan vähemmän kirjoitettua koodia saman toiminnallisuuden tekemisessä kuin Javassa. Lisäksi 17 vastaajaa mainitsi, että Kotlin on myös syntaksiltaan luettavampi kieli kuin Java. Myös Ardito ym. (2020) päätyivät samaan lopputulokseen tutkimuksellaan jossa he järjestivät mobiilikehitys-kurssin, johon osallistui 108 osallistujaa. Kurssilaisilla annetun Android-sovelluksen tekemiseen kirjoitettiin Javalla 583-4745 riviä koodia, missä taas Kotlinilla kirjoitettiin 246-1568 riviä koodia saman sovelluksen tekemiseen. Bose ym. (2018) pääsivät samaan lopputulokseen tarvitun kirjoitetun koodin määrästä kahta kieltä vertaessaan.

3.6 Koodin Hajut

Kent Beck kirjassa ”Refactoring: improving the design of existing code” (Fowler 2018), esitti termin koodin haju (code smell). Koodin haju on tapaus, joka ei ole bugi tai estä ohjelman toimivuutta, mutta viittaa huonoon koodin implemointiin tai muuten huonoon ohjelmoinnin käytöntöön. Esimerkiksi yksi koodin haju voi olla *String Literal Duplication* (SLD), tapaus, jossa sama String-muuttuja on kirjoitettu useaan otteeseen koodissa. Melkein jokainen ohjelma kärsii koodi hajuista, mikäli niitä ei oteta huomioon heti kehityksen alussa (Habchi, Moha ja Rouvoy 2019). Hecht, Moha ja Rouvoy (2016) tutkivat koodin hajuja Android-sovelluksissa ja totesivat hajuilla olevan negatiivinen vaikutus muistinkäyttöön sovelluksissa. Mateus ja Martinez (2019) tekivät tutkimuksen liittyen koodin hajujen vertailuun Javan ja Kotlinin välillä.

Mateus ja Martinez (2019) kävivät läpi 2167 avoimen lähdekoodin (open source) Android-sovelluksen tietokantaa läpi. Tietokanta koostui sovelluksista, joista 11,26 prosenttia oli kirjoitettu osaksi tai kokonaan Kotlinilla, loput Javalla. Tutkimuksessa testattiin koodeja oliopohjaisista hajuista ja Android-alustan liityvistä hajuista. Oliopohjainen haju viittaa koodin hajuun, joka esiintyy oliopohjaisissa ohjelmointikielissä. Tästä esimerkkinä on Swis Army Knife (SAK) tapaus, missä luokalla on valtava määrä eri metodeja, jotka kyettäisiin jakaa erillisiin luokkiin tai aliluokkiin (Hecht 2016). Androidin alustalle ominaisissa hajuissa puhutaan taas hajuista, joita esiintyy Androidin sovelluksissa, esimerkiksi Heavy ASynctask (HAS), mikä tapahtuu, kun vaativia operaatioita kutsutaan eri komponenteista samaan aikaan (Hecht 2016). Mateus ja Martinez (2019) löytöjen mukaan Kotlin kärsi Javaa enemmän kolmesta neljästä tutkitusta oliopohjaisesta hajusta, mutta Javalla todettiin koodin hajuja keskivertoa yleisemmin verrattuna Kotliniin.

Lisäksi mainittava on myös Guptan ja Chauhanin (Gupta ja Chauhan 2021) tekemä tutkimus, jossa käytiin läpi 30 avoimen lähteen Android-ohjelmaa (joista 15 oli kirjoitettu Javalla ja 15 Kotlinilla) hamottaakseen tarkkuuden koodien hajujen löytämiseen sekä vakavuuksien luokitteluun Kotlinissa ja Javassa. Gupta ja Chauhan 2021 pääsivät *JRiP* nimisellä koneoppimis-algoritmilla 96 prosentin tarkkuuteen hajujen löytämisessä Kotlinilla. Kyseinen metodi saatiin myös toimimään yli 90 prosentin tarkkuudella Javalla kirjoitetuille Android-sovelluksille.

3.7 Turvallisuus

Turvallisuusvertauksia Javan ja Kotlinin välillä ei vielä olla paljoa tutkittu, mutta Mazuera-Rozo ym. (2022) kartoittivat ja vertasivat Kotlin ja Javan turvallisuusriskejä keskenään. Tutkimuksessa kartoitettiin 28 tunnettua ohjelmoinin virhettä, jotka johtavat turvallisuusriskeihin Android-sovelluksissa. Nämä riskit jaettiin 4 pääkategoriaan: **1)** Epäkäytännöllinen resurssien kontrollointi, esimerkiksi henkilökohtaisen informaation paljastaminen tilanteissa, joissa ei ollut tarkoitus. **2)** Huomioimaton sitoutuminen ohjelmoinnin standarteihin, eli kehittäjä ei kiinnitä huomiota parhaimmaksi todettuihin ohjelmoinnin standarteihin, esimerkiksi *kuolleen koodin* eli käyttömättömän koodin jättäminen sovelluksen lähdekoodiin. **3)** Huono tarkistus poikkeustilanteista, mikä voi johtaa sovelluksen arvaamattomaan käyttäytymiseen ja **4)** Suojamekanismin vika jolla tarkoitetaan luvatonta pääsyä resurssiin kohteesta, jota ei oltu hyväksytty.

Tutkimuksen lopputuloksessa kategoria 1 vaivasi Kotlinia vähän enemmän. Kategorioissa 2 ja 4 ei ollut nähtäviä eroja. Kategoria 3 esiintyi Javassa Kotlinia enemmän. Huomiona täytyy kuitenkin pitää että vaikka viat ovat pääsääntöisesti kiinni kehittäjästä on hyvä tietää kuinka hyvin kielten työkalut voivat vaikuttaa turvallisuusriskien ja minkä tyyppisten syntymiselle.

3.8 Sovellusten suorituskyky

Peters, Scoccia ja Malavolta (2021) tekivät tutkimuksen Javan ja Kotlinin välillä, jossa he keskittyivät Kotlinin vaikutuksesta sovellusten käynnissä olemisen aikaiseen suorituskykyyn. Suorituskykyä mitattiin ad-hoc ohjelmien avulla, jotka emuloivat käyttäjiä ja käyvät läpi sovelluksen ominaisuuksia (esimerkiksi asetusten muokkaus, tekstin läpi rullaaminen ja ruudulle klikkailun). Tutkimuksessa käytiin läpi 10 sovellusta, joista oli Java ja Kotlin versiot käytettävissä. Kotlinin vaikutusta sovellusten suorituskykyyn mitattiin CPU:N käytössä, muistin käytössä, kutsuista roskien kerääjään, kuvien vaihtumisen renderöintiaikana *(frame time)*, sovellusten koossa ja energian kulutuksessa.

Petersin, Scoccian ja Malavoltan (Peters, Scoccia ja Malavolta 2021) löytöjen mukaan Kotlin käytti enemmän CPU:N tehoja, vei enemmän muistia ja myös mahdollisesti vaikutti ruudun vaihtumisen renderöintiaikaan. Kotlinilla ei taas ollut selkeää vaikutusta roskien keruuseen,

sovellusten kokoon ja energian kulutukseen. Peters, Scoccia ja Malavolta (2021) mukaan Kotlinin negatiivinen vaikutus CPU:n ja muistin käyttössä olivat liian pienet aiheuttakseen selkeää haittaa Kotlinin implementointiin Android-projekteihin. Myös vaikutus seuraavan ruudun renderöintiaikaan antoi liian pienen vaikutuskoon, jotta siitä voitaisiin tehdä selkeitä johtopäätöksiä.

4 Yhteenveto

Kotlin antaa useamman hyödyn Javaan verrattuna, *NullPointerException* virheiden harvinaisuus, pienempi koodin määrä ja mahdollisuus valita funktiopohjaisen ja oliopohjaisen toteutuksen välillä on auttanut Kotlinin kielen suosion kasvua viimevuosina. Suurimpana syynä Kotlinin suosion kasvuun Android-kehityksessä todennäköisesti on kuitenkin Googlen virallinen tuki Android Studiolle, Googlen virallinen *Integrated Development Enviroment (IDE)* Androidille. Myös mahdollisuus integroida Kotlin jo olemassa oleviin Java-projekteihin on antanut Kotlinille mahdollisuuden kasvattaa suosiotaan nopeasti Android-kehittäjien kesken. Javalla on kuitenkin etuja Kotliniin, kielen yleisyys ja pitkä ikä on takanut hyvän tuen yhteisön puolelta Javalle. Mahdollisesti Java on myös hieman paremmin optimoitu, mitä tulee CPU:n ja muistin käyttöön (Peters, Scoccia ja Malavolta 2021), mutta tämä vaatii mahdollisia lisätutkimuksia.

Kotlinin integraatio ei ole kuitenkaan ollut vaivatonta olemassa oleviin projekteihin. Oliveiran, Teixeiran ja Ebertin (Oliveira, Teixeira ja Ebert 2020) tekemän tutkimuksen mukaan osalla Android-kehittäjistä on ollut hankaluuksia integroida Kotlinia olemassa oleviin Java-projekteihin, ongelmiksi on noussut esimerkiksi nollaturva *(nullsafety)* siirtyessä Javan koodista Kotliniin. Oliveira, Teixeira ja Ebert (2020) kuitenkin myös totesivat, että kehittäjät kohtasivat Javan ja Kotlinin yhteensopivuuden enemmän positiiviseksi asiaksi vaikka se ei aina ihan vaivatonta ollutkaan. Lisäksi vaikka Kotlinia ollaat raportoitu luettavammaksi kieleksi (Oliveira, Teixeira ja Ebert 2020; Bose ym. 2018), mutta Oliveiran, Teixeiran ja Elbertin (Oliveira, Teixeira ja Ebert 2020) mukaan Kotlinin luettavuus saatta kärsiä jos sovelluksen koodissa liikakäytetään Kotlinin funktionaalista joustavuutta *(functional flexibility)*, jolla tarkoitetaan esimerkiksi korkean luokan funktioiden ja lambdalausekkeiden käyttöä.

Suurin osa tutkimuksista koostui avoimen lähteen Android-sovelluksista, jatkossa olisi hyödyllistä kuulla Kotlinin implemoinnista virallisissa kaupallisissa projekteissa. Tällä hetkellä ei ole myöskään tiedossa tutkimuksia kehittäjistä jotka ovat kokeilleet Kotlinia, mutta ovat todenneet sen olevan vähemmän käytännöllisempi kuin Java. Tämän ongelman myös nosti esiin tutkielma Martinez ja Gois Mateus (2021), jossa he totesivat että on hankalampaa löytää projekteja, joissa kehittäjät olisivat kokeilleet Kotlinia, mutta eivät olleet kokeneet kieltä

toimivaksi.

Tällä hetkellä Kotlin antaa itsestään positiivisen näkökuvan perinteiseen Javaan verrattuna kun puhutaan Android-sovellusten kehityksestä. Kotlinilla on käytössä enemmän ominaisuuksia, kuten vuorotteluruutiinit ja luokkien laajennusfunktiot. Vaikka Java voittaa edelleen suosiollaan Kotlinin suosio on kuitenkin kasvanut jatkuvasti viimevuosina eri resurssien mukaan[1] [2] (Hecht ja Bergel 2021). Tämä tarkoittaa, että Kotlinin yhteisö, vaikkei Javan yhteisön kokoinen, voi taata että Kotliniin tulee löydettyä apua mahdollisiin ohjelmoinnin ongelmiin.

1. https://insights.stackoverflow.com/trends?tags=kotlin, käytiin 22.3.2022
2. https://madnight.github.io/githut/#/pull_requests/2021/4, käytiin 22.3.2022

Lähteet

Alankus, Gazihan, Basheer Ahamed Fazal, Vinicius Isola, Miles Obare ja Rogerio Theodoro de Brito. 2019. *Java Fundamentals: A Fast-paced and Pragmatic Introduction to One of the World's Most Popular Programming Languages.* Packt Publishing.

Allix, Kevin, Tegawendé F Bissyandé, Jacques Klein ja Yves Le Traon. 2016. "Androzoo: Collecting millions of android apps for the research community". Teoksessa *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR),* 468–471. IEEE.

Ardito, Luca, Riccardo Coppola, Giovanni Malnati ja Marco Torchiano. 2020. "Effectiveness of Kotlin vs. Java in android app development tasks". *Information and Software Technology* 127:106374. ISSN: 0950-5849. https://doi.org/https://doi.org/10.1016/j.infsof.2020.106374.

Bose, Subham, ym. 2018. "A comparative study: java vs kotlin programming in android application development". *International Journal of Advanced Research in Computer Science* 9 (3): 41–45. https://doi.org/http://dx.doi.org/10.26483/ijarcs.v9i3.5978.

Coelho, Roberta, Lucas Almeida, Georgios Gousios, Arie van Deursen ja Christoph Treude. 2017. "Exception handling bug hazards in Android". *Empirical Software Engineering* 22 (3): 1264–1304. https://doi.org/https://doi.org/10.1007/s10664-016-9443-7.

Coppola, Riccardo, Luca Ardito ja Marco Torchiano. 2019. "Characterizing the transition to Kotlin of Android apps: a study on F-Droid, Play Store, and GitHub". Teoksessa *Proceedings of the 3rd ACM SIGSOFT International Workshop on App Market Analytics,* 8–14. https://doi.org/https://doi.org/10.1145/3340496.3342759.

Fowler, Martin. 2018. *Refactoring: improving the design of existing code.* Addison-Wesley Professional.

Gupta, Aakanshi, ja Nidhi Kumari Chauhan. 2021. "A Severity-Based Classification Assessment of Code Smells in Kotlin and Java Application". *Arabian Journal for Science and Engineering,* 1–18. https://doi.org/https://doi.org/10.1007/s13369-021-06077-6.

Habchi, Sarra, Naouel Moha ja Romain Rouvoy. 2019. “The rise of android code smells: Who is to blame?” Teoksessa *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR),* 445–456. IEEE. https://hal.inria.fr/hal-02054788.

Hecht, Geoffrey. 2016. “Detection and analysis of impact of code smells in mobile applications”. *Theses, Université Lille* 1. https://tel.archives-ouvertes.fr/tel-01418158.

Hecht, Geoffrey, ja Alexandre Bergel. 2021. “Quantifying the adoption of Kotlin on Android stores: Insight from the bytecode”. Teoksessa *2021 IEEE/ACM 8th International Conference on Mobile Software Engineering and Systems (MobileSoft),* 94–98. IEEE. https://doi.org/10.1109/MobileSoft52590.2021.00019.

Hecht, Geoffrey, Naouel Moha ja Romain Rouvoy. 2016. “An empirical study of the performance impacts of android code smells”. Teoksessa *Proceedings of the international conference on mobile software engineering and systems,* 59–69. https://hal.inria.fr/hal-01276904.

JetBrains. 2022. “https://kotlinlang.org/docs/home.html”. Viitattu 7. maaliskuuta 2022. https://kotlinlang.org/docs/home.html.

Martinez, Matias, ja Bruno Gois Mateus. 2021. “Why did developers migrate Android Applications from Java to Kotlin?” *IEEE Transactions on Software Engineering,* 1–1. https://doi.org/10.1109/TSE.2021.3120367.

Mateus, Bruno Góis, ja Matias Martinez. 2019. “An empirical study on quality of Android applications written in Kotlin language”. *Empirical Software Engineering* 24 (6): 3356–3393. https://doi.org/https://doi.org/10.1007/s10664-019-09727-4.

Mazuera-Rozo, Alejandro, Camilo Escobar-Velásquez, Juan Espitia-Acero, David Vega-Guzmán, Catia Trubiani, Mario Linares-Vásquez ja Gabriele Bavota. 2022. “Taxonomy of security weaknesses in Java and Kotlin Android apps”. *Journal of Systems and Software* 187:111233. https://doi.org/https://doi.org/10.1016/j.jss.2022.111233.

Oliveira, Victor, Leopoldo Teixeira ja Felipe Ebert. 2020. “On the adoption of kotlin on android development: A triangulation study”. Teoksessa *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER),* 206–216. IEEE. https://doi.org/10.1109/SANER48275.2020.9054859.

Peters, Michael, Gian Luca Scoccia ja Ivano Malavolta. 2021. “How Does Migrating to Kotlin Impact the Run-time Efficiency of Android Apps?” Teoksessa *2021 IEEE 21st International Working Conference on Source Code Analysis and Manipulation (SCAM),* 36–46. IEEE. https://doi.org/https://dx.doi.org/10.1109/SCAM52516.2021.00014.

Poo, Danny, Derek Kiong ja Swarnalatha Ashok. 2007. *Object-oriented programming and Java.* Springer Science & Business Media.

Samuel, Stephen, ja Stefan Bocutiu. 2017. *Programming kotlin.* Packt Publishing Ltd.