Kotlin

Case study: Non-mainstream programski jezici

Luka ČULINA

OBJEKTNO ORIJENTIRANO PROGRAMIRANJE

Contents

[Uvod 2](#_Toc187530191)

[Kotlin 3](#_Toc187530192)

[Glavne karakteristike jezika 4](#_Toc187530193)

[Sintaksa 4](#_Toc187530194)

[Ključne značajke sintakse 4](#_Toc187530195)

[Paradigme 6](#_Toc187530196)

[Objektno-orijentirano programiranje (OOP) 6](#_Toc187530197)

[Funkcionalno programiranje (FP) 7](#_Toc187530198)

[Proceduralno programiranje 8](#_Toc187530199)

[Kombinacija paradigmi 9](#_Toc187530200)

[Specifične značajke Kotlina 9](#_Toc187530201)

[1. Interoperabilnost s Javom 9](#_Toc187530202)

[2. Sigurnost u odnosu na null vrijednosti (Null Safety) 9](#_Toc187530203)

[3. Produktivnost kroz proširive funkcije 10](#_Toc187530204)

[4. Podrška za multiplatformski razvoj 10](#_Toc187530205)

[5. Jednostavno rukovanje konkurentnošću 10](#_Toc187530206)

# Uvod

Razvoj programskih jezika prolazio je kroz brojne faze od svojih početaka, a danas postoji na stotine jezika koji zadovoljavaju različite potrebe programera. Dok su neki jezici, poput Jave, Pythona i C++, stekli status "mainstream" jezika zahvaljujući širokoj primjeni i dugogodišnjoj dominaciji u industriji, mnogi drugi jezici, često nazivani "non-mainstream" jezicima, ostaju u sjeni unatoč inovativnim značajkama koje nude.

"Non-mainstream" programski jezici obuhvaćaju jezike koji možda nisu toliko široko korišteni, ali su značajni zbog specifičnih problema koje rješavaju, svoje tehničke inovativnosti ili utjecaja na razvoj drugih jezika. Takvi jezici često ciljaju specifične industrije, paradigme programiranja ili zahtjeve modernih sustava. Na primjer, Erlang se koristi za sustave visoke dostupnosti poput telekomunikacija, dok Rust naglašava sigurnost i visoke performanse, čineći ga popularnim u razvoju sustava i alata za upravljanje memorijom.

Važnost "non-mainstream" jezika ogleda se u nekoliko ključnih aspekata:

1. **Inovacija** – Ovi jezici često uvode nove koncepte koji kasnije postaju standard u mainstream jezicima. Primjerice, Haskell je popularizirao čisto funkcionalno programiranje, dok je Scala pomogla integraciji funkcionalnih značajki u Javu.
2. **Specijalizacija** – Jezici poput Julia dizajnirani su za visoke performanse u znanstvenim izračunima, dok Lua nudi jednostavnost i fleksibilnost za ugrađene sustave i igre.
3. **Raznolikost ekosustava** – Različiti jezici prilagođeni su specifičnim potrebama zajednica programera, čime se potiče razvoj specijaliziranih alata i biblioteka.

Unatoč njihovoj relativnoj nepopularnosti u masovnoj upotrebi, "non-mainstream" jezici često igraju ključnu ulogu u inovacijama, učinkovitosti i evoluciji softverskog razvoja. Njihova vrijednost ne leži samo u neposrednoj primjeni, već i u širem doprinosu računalnim znanostima i razvoju paradigmi programiranja.

Cilj ovog rada je istražiti jedan od tih jezika, Kotlin, koji je postao sve popularniji posljednjih godina. Kotlin, iako relativno mlad jezik, pruža snažnu podršku objektno-orijentiranom i funkcionalnom programiranju te je postao jedan od glavnih jezika za razvoj Android aplikacija. Analiza ovog jezika pružit će dublji uvid u njegova svojstva, primjenu i značajke koje ga izdvajaju u konkurenciji.

# Kotlin

Kotlin je moderan open-source programski jezik koji je statički tipiziran i podržava objektno-orijentirano i funkcionalno programiranje. Razvijen je od strane JetBrainsa, kompanije poznate po razvoju alata za programere, Kotlin je brzo stekao popularnost zahvaljujući svojoj interoperabilnosti s Javom i pojednostavljenju razvoja softvera. Jezgra jezika temelji se na dugogodišnjoj evoluciji programskih jezika, zbog čega mnogi programeri prepoznaju poznate sintaktičke elemente inspirirane jezicima kao što su Java, Scala i C#.

Primarna prednost Kotlina leži u njegovoj fleksibilnosti i prilagodljivosti različitim platformama. Postoje tri glavne varijante jezika:

* **Kotlin/JVM**: koji se izvodi na Java Virtual Machine (JVM), čineći ga savršenim izborom za razvoj Android aplikacija.
* **Kotlin/JS**: koji omogućava razvoj aplikacija za web korištenjem JavaScript-a.
* **Kotlin/Native**: koji omogućava pisanje programa koji se mogu izvršavati izravno na uređajima bez JVM-a, uključujući iOS i ugrađene sustave.

Kotlinova popularnost naglo je porasla nakon što ga je Google službeno podržao kao preferirani jezik za razvoj Android aplikacija 2017. godine. Njegova integracija s Android ekosustavom čini ga jednostavnim za učenje i primjenu, dok njegov dizajn omogućuje korištenje postojećih Java biblioteka i alata. Ova interoperabilnost omogućuje timovima postupnu migraciju iz Jave u Kotlin, što dodatno povećava njegovu privlačnost u industriji.

Osim što poboljšava produktivnost programera, Kotlin pruža napredne značajke kao što su sigurnost u odnosu na null vrijednosti (*null safety*), proširive funkcije i lakše rukovanje konkurentnošću. Svojim dizajnom smanjuje potrebu za šablonskim kodom (*boilerplate*), što vodi do čitljivijeg i održivijeg koda.

Kotlin je pronašao svoju primjenu i izvan razvoja Android aplikacija, uključujući backend sustave putem integracije s popularnim alatima kao što su Spring i Ktor. Njegova sposobnost dijeljenja poslovne logike između platformi pomoću Kotlin/Multiplatform čini ga odličnim izborom za razvoj aplikacija koje se koriste na Androidu, iOS-u i webu. Sve više kompanija, od startupa do velikih tehnoloških divova, prepoznaje vrijednost Kotlina u razvoju skalabilnih i održivih softverskih rješenja.

# Glavne karakteristike jezika

Kotlin se ističe brojnim značajkama koje ga čine moćnim, fleksibilnim i pogodnim za moderni razvoj softvera. Dizajniran s ciljem da unaprijedi produktivnost programera, Kotlin nudi intuitivnu sintaksu, robusne alate za osiguravanje sigurnosti koda te značajke koje pojednostavljuju rad s kompleksnim sustavima. Njegova interoperabilnost s Javinim ekosustavom, podrška za multiplatformski razvoj i usmjerenost na eliminaciju uobičajenih programskih grešaka čine ga idealnim alatom za širok spektar primjena. U ovoj sekciji detaljnije ćemo opisati njegovu sintaksu, paradigme koje podržava i specifične značajke koje ga izdvajaju od drugih jezika, naglašavajući kako doprinosi ubrzanju razvoja softverskih rješenja i smanjenju složenosti koda.

## Sintaksa

Sintaksa Kotlina osmišljena je kako bi bila intuitivna, čitljiva i jednostavna za korištenje, čime se značajno smanjuje složenost koda i povećava produktivnost programera. Inspirirana jezicima poput Jave, Scale i C#, Kotlin prepoznaje i unapređuje najbolje prakse iz postojećih jezika, istovremeno eliminirajući njihove nedostatke. Sintaktička jednostavnost Kotlina omogućava brzo usvajanje jezika, čak i za početnike, dok iskusni programeri cijene moćne alate koje pruža.

### Ključne značajke sintakse

* **Statičko tipiziranje**:  
  Kotlin je statički tipiziran jezik, što znači da se tipovi varijabli provjeravaju u vrijeme prevođenja (*compile-time*), smanjujući mogućnost grešaka u vrijeme izvođenja (*runtime*). Programeri mogu eksplicitno definirati tipove varijabli, ali Kotlin također nudi automatsku dedukciju tipa pomoću ključnih riječi val (za nepromjenjive varijable) i var (za promjenjive varijable). Na primjer:

A black screen with white text

Description automatically generated

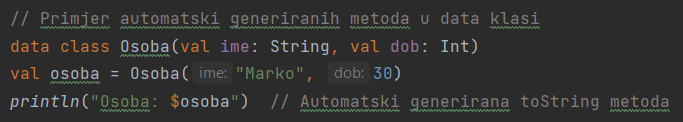
* **Jednostavne lambda funkcije**:   
  Lambda izrazi u Kotlinu omogućavaju funkcionalni stil programiranja, omogućujući programerima da pišu koncizan i čitljiv kod. Na primjer:

A black background with white text

Description automatically generated

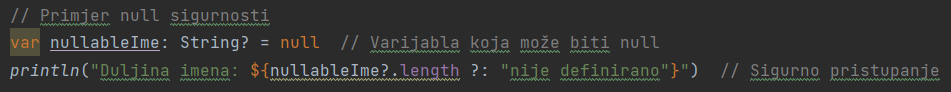
Ovaj pristup značajno pojednostavljuje rukovanje funkcijama višeg reda (*higher-order functions*) i rad s kolekcijama.

* **Minimalni šablonski kod (*boilerplate*)**:  
  Kotlin automatski generira često korištene elemente koda, čime se štedi vrijeme i smanjuje mogućnost grešaka. Na primjer, korištenjem ključne riječi data, moguće je definirati klase s unaprijed generiranim metodama kao što su toString(), equals() i hashCode():



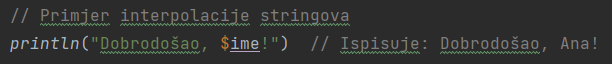
Ovaj pristup omogućava da se programeri fokusiraju na poslovnu logiku, umjesto na tehničke detalje implementacije.

* **Null sigurnost (*Null Safety*)**:  
  Kotlin uvodi strogu sigurnost u radu s null vrijednostima, koja je jedan od najčešćih uzroka pogrešaka u mnogim programskim jezicima. U Kotlinu, varijable koje mogu sadržavati null vrijednosti moraju biti eksplicitno deklarirane pomoću operatora ?:



Ova značajka eliminira mogućnost pojave *NullPointerException* grešaka, čime se povećava pouzdanost koda.

* **Dodatne značajke sintakse**
  + **Izrazito čitljive petlje i kontrolne strukture**: Kotlin uvodi moderne konstrukcije kao što su petlje preko kolekcija (for (item in collection)), što olakšava rad s podacima.
  + **Proširive funkcije (*Extension Functions*)**: Programeri mogu dodavati funkcionalnosti postojećim klasama bez potrebe za njihovom izmjenom, čime se povećava fleksibilnost jezika.
  + **String interpolacija**: Kotlin podržava interpolaciju stringova, omogućujući ugrađivanje izraza unutar stringova pomoću ${}:



Svojom pažljivo osmišljenom sintaksom, Kotlin omogućava pisanje čitljivog, sigurnog i održivog koda, koji zadovoljava potrebe kako početnika, tako i iskusnih programera.

## Paradigme

Kotlin je univerzalan jezik koji podržava različite paradigme programiranja, omogućujući fleksibilan i prilagodljiv stil pisanja koda. Kombinacija paradigmi omogućuje programerima da odaberu pristup koji najbolje odgovara specifičnom problemu, što Kotlin čini snažnim i svestranim alatom za razvoj softvera.

### Objektno-orijentirano programiranje (OOP)

Kotlin potpuno podržava objektno-orijentirano programiranje, koristeći klase, objekte i sučelja kao temeljne građevne blokove. OOP omogućuje organizaciju koda u module i ponovnu upotrebu putem nasljeđivanja i polimorfizma.

* **Klase i objekti**:  
  Klase u Kotlinu koriste se za definiranje objekata, dok ključna riječ object omogućuje definiranje singletona ili statičkih instanci objekata.

A screen shot of a computer program

Description automatically generated

* **Sučelja (Interfaces)**:  
  Kotlin podržava višestruko nasljeđivanje kroz sučelja, koja mogu sadržavati metode s implementacijom ili bez nje.

A computer screen with white and green text

Description automatically generated

### Funkcionalno programiranje (FP)

Kotlin nudi snažnu podršku za funkcionalno programiranje, omogućujući koncizno i izrazito pisanje koda.

* **Lambda izrazi i funkcije višeg reda**:  
  Lambda izrazi koriste se za definiranje funkcija "u letu", dok funkcije višeg reda prihvaćaju druge funkcije kao parametre ili ih vraćaju kao rezultat.

A screen shot of a computer

Description automatically generated

* **Nepromjenjive kolekcije**:  
  Kotlin nudi nepromjenjive kolekcije, što je ključno za funkcionalni stil programiranja, gdje se podaci ne mijenjaju već se stvaraju novi.

A black screen with white text

Description automatically generated

### Proceduralno programiranje

Kotlin podržava proceduralni stil, gdje se programi strukturiraju kao niz funkcija i kontrolnih blokova.

* **Funkcije i kontrolne strukture**:  
  Proceduralni pristup omogućuje definiranje funkcija za rješavanje specifičnih zadataka i njihovo pozivanje unutar programa.

A screen shot of a computer code

Description automatically generated

* **Kontrolni blokovi**:  
  Kotlin koristi poznate kontrolne strukture poput if, when, for, i while, čineći proceduralni kod intuitivnim i čitljivim.

A screen shot of a computer program

Description automatically generated

### Kombinacija paradigmi

Jedna od glavnih snaga Kotlina je njegova sposobnost da kombinira paradigme. Na primjer, programer može koristiti objektno-orijentirani pristup za organizaciju koda, dok istovremeno koristi lambda funkcije i nepromjenjive kolekcije za specifične zadatke. Ova fleksibilnost omogućuje razvoj softverskih rješenja koja su čitljiva, učinkovita i jednostavna za održavanje.

## Specifične značajke Kotlina

Kotlin se ističe nizom jedinstvenih značajki koje ga čine pogodnim za razvoj modernih aplikacija. Ove značajke omogućuju bolju produktivnost, veću sigurnost i lakšu integraciju s postojećim projektima. U nastavku ćemo detaljno opisati nekoliko ključnih karakteristika koje Kotlin čine posebnim:

### 1. Interoperabilnost s Javom

Jedna od najvećih prednosti Kotlina je njegova besprijekorna interoperabilnost s Java kodom. Kotlin je dizajniran da se bez problema integriše s postojećim Java projektima, omogućujući programerima da koriste Java knjižnice i frameworke u Kotlin projektima. Na primjer, možete koristiti Kotlin za novu funkcionalnost u postojećem Java projektu, ili obrnuto, bez potrebe za potpunim refaktoriranjem koda. Ovo omogućuje tvrtkama koje već koriste Javu da postupno migriraju na Kotlin, a istovremeno održavaju postojeći Java kod.

### 2. Sigurnost u odnosu na null vrijednosti (Null Safety)

Jedna od najpoznatijih značajki Kotlina je njegova sigurnost u odnosu na null vrijednosti. U mnogim programskim jezicima, rad s null vrijednostima može biti izvor mnogih grešaka pri izvođenju programa (runtime errors). Kotlin uvodi koncept null sigurnosti putem svog tipnog sustava. Varijable koje mogu biti null označavaju se s oznakom ?, dok su svi ostali tipovi ne-nullable po defaultu. To znači da se ne možete slučajno pozvati na null vrijednost, jer Kotlin zahtijeva eksplicitno rukovanje mogućim null vrijednostima. Na taj način, Kotlin značajno smanjuje broj grešaka koje mogu nastati zbog nepravilnog korištenja null referenci.

A screen shot of a computer

Description automatically generated

### 3. Produktivnost kroz proširive funkcije

Kotlin omogućava programerima da proširuju postojeće klase bez potrebe za mijenjanjem njihovog izvornog koda. Ova značajka je poznata kao proširive funkcije (extension functions). Proširive funkcije omogućuju dodavanje novih metoda postojećim klasama koje ne možete modificirati, kao što su klase iz vanjskih knjižnica. To omogućuje vrlo fleksibilan i čist način za proširenje funkcionalnosti aplikacija bez potrebe za promjenom izvornog koda.

A screen shot of a computer program

Description automatically generated

### 4. Podrška za multiplatformski razvoj

Kotlin nudi podršku za multiplatformski razvoj putem Kotlin/Multiplatform alata. Korištenjem ovog alata, programeri mogu dijeliti poslovnu logiku između različitih platformi poput Androida, iOS-a, weba, pa čak i backend-a. Ova značajka omogućava razvoj aplikacija koje dijele zajednički kod, smanjujući vrijeme razvoja i održavanje aplikacija na više platformi. Programeri mogu pisati dijelove aplikacije koji su zajednički svim platformama u zajedničkom kodu, dok specifični dijelovi mogu biti implementirani za svaku platformu posebno.

### 5. Jednostavno rukovanje konkurentnošću

Kotlin uvodi *coroutines* kao način za rukovanje asinhronim i konkurentnim kodom. Coroutines su lagani, visoko učinkoviti i omogućuju jednostavniji rad s višezadaćnim procesima bez potrebe za korištenjem tradicionalnih niti. Omogućuju pisanje asinkronog koda na način koji izgleda kao sinkroni kod, čineći kod čitljivijim i lakšim za razumijevanje. Coroutines olakšavaju implementaciju složenih operacija poput mrežnih poziva ili operacija na pozadinskim nitima bez blokiranja glavne niti aplikacije.

A screen shot of a computer program

Description automatically generated

Ove specifične značajke čine Kotlin snažnim i fleksibilnim jezikom koji omogućuje razvoj aplikacija visokih performansi s većom produktivnošću i sigurnošću. Kotlin se lako prilagođava potrebama modernog razvoja softverskih aplikacija i omogućuje programerima da pišu čist i održiv kod.