Mobilfejlesztés: Az Android az első a Kotlin

A világ leggyakrabban használt mobil operációs rendszere, az Android hivatalosan is elkezdte támogatni a Kotlint, mint az alkalmazások készítésének nyelvét 2017-ben. Csak 2 évvel később, 2019-ben, a fejlesztők sok pozitív visszajelzése után, az Android lett a Kotlinfirst, így ez lett az új alkalmazások alapértelmezett választása. Azóta a Google fejlesztőeszközei, Jetpack könyvtárai (https://developer.android.com/jetpack), minták, dokumentációk és képzési tartalmak elsősorban a Kotlinra összpontosítanak. A Kotlin nyelvi funkciói az Android fejlesztését sokkal produktívabb és kellemesebb élménnyé teszik. A gyakori fejlesztési feladatok sokkal kevesebb kóddal is elvégezhetők. Az Android csapat által épített Android KTX könyvtár(https://developer.android.com/kotlin/ktx) még tovább javítja a felhasználói élményt azáltal, hogy Kotlin-barát adaptereket ad hozzá sok szabványos Android API-hoz.A Google Jetpack Compose eszközkészlete(https://developer.android.com/jetpack/compose) natív felhasználói felületek készítéséhez Androidra szintén az alapoktól kezdve a Kotlin számára készült. Felöleli a Kotlin nyelvi funkcióit, és lehetővé teszi, hogy kevesebb, egyszerűbb és könnyebben karbantartható kódot írjon a mobilalkalmazások felhasználói felületének felépítésekor. Íme egy példa a Jetpack Compose-ra, csak hogy ízelítőt adjunk abból, milyen érzésAndroid fejlesztés a Kotlinnel. A következő kód megjeleníti az üzenetet, és kattintásra kibontja vagy elrejti a részleteket:

```
@Composable
 fun MessageCard(modifier: Modifier, message: Message) {
 var isExpanded by remember { mutableStateOf(false) }

 Column(modifier.clickable { isExpanded = !isExpanded }
 ) {

 Text(message.body)

 if (isExpanded) {

 MessageDetails(message)
        }
    }
 }
 @Composable
 fun MessageDetails(message: Message) { /* ... */ }
```

A Kotlin JavaScriptre fordítható, lehetővé téve a Kotlin kód futtatását a böngészőben és a futtatókörnyezetekben, például a Node.js.A Kotlin/Native segítségével a Kotlin kódot natív binárisokra fordíthatja, lehetővé téve az iOS és más platformok önálló programokkal történő megcélzását. A Kotlin/Wasm, egy olyan célpont, amely az írás idején még fejlesztés alatt áll, lehetővé teszi a Kotlin kód lefordítását a WebAssembly bináris utasításformátumra, lehetővé téve a kód futtatását a WebAssembly virtuális gépeken, amelyeket modernböngészőkben és más futtatókörnyezetekben szállítanak. A Kotlin azt is lehetővé teszi, hogy megadja, hogy a szoftver mely részeit kell megosztani a különböző célok között, és mely részeknek van platformspecifikus implementációja, nagyon finom hangsúlyon. Mivel ez a vezérlő nagyon finom, a gyakori és a platformspecifikus kód legjobb kombinációját keverheti és illesztheti. Ez a mechanizmus, amelyet

a Kotlin várhatónak/aktuálisnak nevez, lehetővé teszihasználja ki a Kotlin kód platformspecifikus funkcióit. Ez hatékonyan enyhíti a "legkisebb közös nevező megcélzott" klasszikus problémáját, amellyel a platformok közötti eszközkészletek általában szembesülnek, és amelyben az összes megcélzott platformon elérhető műveletek egy részhalmazára korlátozódik.

A Kotlin esetében az IntelliJ IDEA beépülő modult a fordítóval együtt fejlesztik, és a nyelvi funkciókat mindig az eszközök szem előtt tartásával tervezzük. Az IDE támogatás is nagy szerepet játszik a Kotlin funkcióinak felfedezésében. Sok esetben az eszközök automatikusan felismerik a gyakori kódmintákat, amelyek tömörebb konstrukciókkal helyettesíthetők, és felajánlják a kód javítását. Az automatikus javítások által használt nyelvi funkciók tanulmányozásával megtanulhatja alkalmazni ezeket a funkciókat a saját kódjában is.

Kotlin tömör Köztudott, hogy a fejlesztők több időt töltenek a meglévő kód olvasásával, mint új kód írásával. Képzelje el, hogy egy nagy projektet fejlesztő csapat tagja vagy, és új funkciót kell hozzáadnod, vagy ki kell javítanod egy hibát. Mik az első lépéseid? Megkeresi a kód pontos részét, amelyet módosítania kell, és csak ezután hajtja végre a javítást. Sok kódot olvasol, hogy megtudd, mit kell tenned. Lehet, hogy ezt a kódot nemrégiben írták a kollégái vagy valaki, aki már nem dolgozik a projekten – vagy Ön már régen. Csak a környező kód megértése után végezheti el a szükséges módosításokat. Minél egyszerűbb és tömörebb a kód, annál gyorsabban megérti, mi történik. Természetesen a jó tervezés itt jelentős szerepet játszik, és akifejező nevek kiválasztása, biztosítva, hogy változóit, függvényeit és osztályait pontosan leírják a nevükkel. De a nyelvválasztás és tömörsége is fontos. A nyelv tömör, ha a szintaxisa egyértelműen kifejezi az elolvasott kód szándékát, és nem takarja el sablonnal, amely a szándék megvalósításának módjának meghatározásához szükséges.

Az objektumorientált nyelvek sok szabványos sablonja, mint például a getterek, a beállítók és a konstruktor paraméterek mezőkhöz való hozzárendelésének logikája, implicit a Kotlinban, és nem zsúfolja össze a forráskódot. A pontosvesszők is elhagyhatók a Kotlinban, eltávolítva egy kis extra rendetlenséget a kódból, és hatékony típuskövetkeztetése megkíméli Önt attól, hogy explicit módon megadja azokat a típusokat, amelyekre a fordító következtethet a kontextusból. A Kotlin gazdag szabványos könyvtárral rendelkezik, amely lehetővé teszi, hogy ezeket a hosszú, ismétlődő kódszakaszokat könyvtári metódushívásokkal helyettesítse. A Kotlin lambdas és anonim függvények (kifejezésként használt függvényliterálok) támogatása megkönnyíti a kis kódblokkok átadását a könyvtárfüggvényeknek. Ez lehetővé teszi, hogy beágyazza az összes közös részt a könyvtárba, és csak az egyedi, feladatspecifikus részt tartsa meg a felhasználói kódban. Ugyanakkor a Kotlin nem próbálja meg a forráskódot a lehető legkevesebb karakterre összecsukni. A Kotlin például támogatja a rögzített operátorok túlterhelését, ami azt jelenti, hogy egyéni implementációkat biztosíthat + , - , in vagy [] . A felhasználók azonban nem határozhatják meg saját customoperátoraikat. Ez megakadályozza, hogy a fejlesztők a metódusneveket rejtélyes írásjelekre cseréljék, amelyeket nehezebb lenne olvasni és nehezebben megtalálni a dokumentációs rendszerekben, mint a kifejező nevek használata. A tömörebb kód megírása kevesebb időt vesz igénybe, és ami még fontosabb, kevesebb időt vesz igénybeolvasása és megértése. Ez javítja a termelékenységet, és lehetővé teszi, hogy gyorsabban végezze el a dolgokat.

A JVM-en való futtatás már számos biztonsági garanciát nyújt – például a memória biztonságát, a puffertúlcsordulás megakadályozását és a dinamikusan lefoglalt memória helytelen használata által okozott egyéb problémákat. A JVM statikusan gépelt nyelveként a Kotlin biztosítja az alkalmazások típusbiztonságát is. És a Kotlin tovább megy: megkönnyíti a csak olvasható változók meghatározását (a valkeyszóval keresztül), és gyorsan csoportosíthatja őket nem módosítható (adat) osztályokba, ami további biztonságot jelent a többszálú alkalmazások számára. Ezen túlmenően a Kotlin célja, hogy megakadályozza a futásidőben előforduló hibákat azáltal, hogy

ellenőrzéseket hajt végre a fordítási idő alatt. A legfontosabb, hogy Kotlin arra törekszik, hogy eltávolítsa a NullPointerException a programból. A Kotlin típusú rendszer nyomon követiolyan értékeket, amelyek null értékűek lehetnek és nem lehetnek, és tiltja azokat a műveleteket, amelyek futásidőben aNullPointerException vezethetnek. Az ehhez szükséges többletköltség minimális – egy típus nullázhatóként való megjelölése csak egyetlen karaktert igényel, egy kérdőjelet a végén:

```
fun main() {
var s: String? = null
var s2: String=""
println(s.length)
println(s2.hossz)
}
```

Lehet, hogy null Lehet, hogy nem null Nem fordít le, megmentve az összeomlástól A várt módon fog működniEnnek kiegészítéseként a Kotlin számos kényelmes módszert kínál a nullableable-adatok kezelésére. Ez nagyban segít az alkalmazások összeomlásának kiküszöbölésében. Egy másik kivételtípus, amelyet Kotlin segít elkerülni, az osztály cast kivétele, amely akkor fordul elő, amikor egy objektumot egy típusra önt anélkül, hogy először ellenőrizné, hogy rendelkezik-e a megfelelő típus. Kotlin egyetlen műveletben egyesíti az ellenőrzést és a varázslást. Ez azt jelenti, hogy miután ellenőrizted a típust, hivatkozhatsz az adott típusú tagokra további castingok, újranyilatkozások vagy ellenőrzések nélkül. Ebben a példában típusellenőrzést hajt végre az értékváltozón, amely bármilyen típusú lehet. A fordító tudja, hogy a feltételes valódi ágában az értéknek String típusúnak kell lennie, így biztonságosan engedélyezheti az adott típusú metódusok használatát (ún. smart-cast, amelyet a itt ebben szakaszban részletesebben megismerünk).

```
fun modify(value: Any) { //Any bármilyen típus lehet
   if (value is String) {
       println(value.uppercase())//nagybetűre cserél minden betüt uppercase()

   }
}
```

Kotlin fájl Fordítása:

```
kotlinc <source file or directory> -include-runtime
d <jar name>
 java -jar <jar name>
```

Összefoglalás:

A Kotlin statikusan típusos, és támogatja a típuskövetkeztetést, lehetővé téve a helyesség és a teljesítmény fenntartását, miközben a forráskód tömör marad. A Kotlin támogatja mind az objektumorientált, mind a funkcionális programozási stílusokat, lehetővé téve a magasabb szintű absztrakciókat az első osztályú függvények révén, és egyszerűsítve a tesztelést és a többszálú fejlesztést a megváltoztathatatlan értékek támogatásával. A korutinok a szálak könnyű alternatívái, és segítenek az aszinkron kód természetessé tételében azáltal, hogy lehetővé teszi a szekvenciális kódhoz hasonló logika írását, és a szülő-gyermek kapcsolatokban lévő egyidejű kód strukturálását. A Kotlin jól működik a szerveroldali alkalmazásokhoz, a Kotlin-firstkeretrendszerekkel, mint például a Ktor és a http4k, valamint teljes mértékben támogatja az

összes létezőtJava keretrendszer, például a Spring Boot.Az Android a Kotlin az első, fejlesztői eszközökkel, könyvtárakkal, mintákkal és dokumentációval, amelyek mind elsősorban a Kotlinra összpontosítanak.A Kotlin Multiplatform a Kotlin kódot a JVM-en túli célpontokra viszi, beleértve az iOS-t és az internetet. A Kotlin ingyenes és nyílt forráskódú, és több buildrendszert és IDE-t támogat.Az IntelliJ IDEA és az Android Studio lehetővé teszi a zökkenőmentes navigálást a Kotlin és Java nyelven írt kódon keresztül.A Kotlin játszótér (https://play.kotlinlang.org) egy gyors módja annak, hogy kipróbálja a Kotlint beállítás nélkül. Az automatizált Java-Kotlin konverter(InteliJ -ben) lehetővé teszi, hogy meglévő kódját és tudását a Kotlinba vigye.

A Kotlin pragmatikus, biztonságos, tömör és interoperábilis, ami azt jelenti, hogy a bevált megoldások használatára összpontosít a gyakori feladatokhoz, megelőzvegyakori hibákat, mint például a NullPointerException s; támogatáskompakt és könnyen olvasható kód; és korlátlan integrációt biztosít a Java-val.