УДК 004.5 DOI: 10.15330/itee.2026.4.04

А.І. КСЕНИЧ, канд. техн. наук

ПОРІВНЯЛЬНИЙ АНАЛІЗ МОВ JAVA ТА KOTLIN У КОНТЕКСТІ ПОЧАТКОВОГО НАВЧАННЯ ДЛЯ СТУДЕНТІВ

Вступ

Стрімке зростання попиту на програмних розробників спонукає більш змістовніше підходити до питання вибору першої мови програмування для навчання. Так наприклад, одним із найперспективніших напрямів професійного старту є Android-розробка, оскільки за даними StatCounter у 2024 році частка Android на світовому ринку мобільних операційних систем становила близько 70,9% [1]. Водночас Kotlin, основна мова нативної Android розробки, уже не обмежується тільки мобільною розробкою, оскільки вона активно використовується для створення серверних застосунків. Так Spring Framework, починаючи з версії 5, забезпечує повноцінну підтримку Kotlin, а фреймворк Ktor орієнтований на розробку мікросервісів і REST API у JVM-екосистемі [2, 3]. За результатами JetBrains Developer Ecosystem Survey 2024, вже понад 60% Kotlin-розробників застосовують її також у серверній розробці [4]. Це свідчить про суттєве розширення сфери використання Kotlin і зумовлює необхідність переосмислення її місця в університетських курсах програмування. У зв'язку з цим постає питання, чи доцільно й надалі розпочинати навчання з мови Java, яка тривалий час була академічним стандартом, чи все ж таки варто перейти до Kotlin, яка після оголошення Google у 2019 році є основною мовою розробки для Android і набуває все ширшого поширення [5].

Ця дилема є особливо актуальною, якщо врахувати різницю між двома мовами. Java вимагає від початківців засвоєння значно більшого об’єму синтаксичних конструкцій для написання першої змістовної програми. Kotlin же, з його виведенням типів, data класами та null безпекою, дає змогу зосередитися на логіці алгоритмів значно раніше, різниця тут досить ґрунтовна. При цьому вибір між мовами є актуальним не лише для Android стеку. Backend розробник на JVM обов’язково стикається з Kotlin кодом у реальних проєктах, а асинхронна модель Kotlin Coroutines однаково застосовується як у мобільній так і в серверній розробці. З іншого боку, Java залишається мовою з більшою кількістю вакансій і масштабною присутністю в серверній розробці. Тому наразі постає актуальним питання, чи справді лаконічніший синтаксис Kotlin означає меншу когнітивну складність для студента-початківця і яка мова краще готує до реального ринку праці? Відповіді на ці питання потребують одночасного аналізу педагогічних даних, статистики ринку праці та прикладів коду.

Аналіз останніх досліджень і публікацій

Проблематика вибору першої мови програмування розглядається дослідниками вже кілька десятиліть, однак порівняння Java і Kotlin у педагогічному контексті є відносно новою темою. Так в емпіричному дослідженні [6] вивчено вплив синтаксису мови на кількість помилок новачків і встановлено, що мови з меншою кількістю спеціальних символів та явніше вираженою семантикою суттєво знижують рівень помилок при першому знайомстві з програмуванням. Результати показали, що студенти, які вивчали мови з так званим «нотаційним синтаксисом» (C, Java, Perl), припускалися статистично значно більше помилок порівняно з тими, хто починав із більш лаконічних мов. Вибірка перевищувала 1500 учасників. Ця робота стала теоретичним підґрунтям для подальших порівняльних досліджень і безпосередньо пов’язана з контекстом Java-Kotlin, оскільки Java належить саме до мов із «нотаційним синтаксисом».

У роботі [7] порівняно Java і Python, як перші мови для студентів інженерних спеціальностей. На вибірці з 240 студентів встановлено, що група Python досягала базового рівня грамотності швидше, а рівень відсіву на першому курсі в цій групі був на 15% нижчим. Цей результат пов'язано зі зменшеним «синтаксичним шумом» Python. Хоча Python і Kotlin - різні мови, обидві поділяють властивість виведення типів і скороченого boilerplate, що робить висновки роботи релевантними для цього дослідження.

У масштабному контрольованому експерименті [8] за участю 391 студента половина навчалася на Python, половина - на C. Група Python показала кращі результати в розумінні рекурсії, обробки рядків та базових структур даних, а середній бал за курс у них був на 0,8 бала вищим (за 10-бальною шкалою). Важливо, що різниця у академічній успішності зберігалася і на другому курсі, навіть коли обидві групи перейшли до єдиного навчального плану. Переваги легшого старту мали довготривалий ефект. Зроблено висновок, що мова зі складнішим синтаксисом витрачає когнітивний ресурс студента на «синтаксичнй шум» замість розвитку алгоритмічного мислення.

У дослідженні [9], проведеному в трьох грецьких університетах, вивчено наслідки використання C-подібних та Pascal-подібних мов. Студенти, які починали з Pascal-подібного синтаксису, краще засвоювали концепції структурного програмування та рідше припускалися помилок у роботі з циклами й умовними операторами. Вибірка охоплювала понад 400 студентів протягом трьох навчальних семестрів. Результат цілком узгоджується з попередніми дослідженнями і є безпосередньо релевантним для порівняння Java (C-подібна) та Kotlin (лаконічніший синтаксис).

У роботі [10] проаналізовано тренди вибору мов у курсах CS1/CS2 у 115 американських університетах. Python остаточно витіснив Java з позиції першої мови у більшості топ-університетів. У 2021 році 69% курсів CS1 використовували Python проти 17% для Java. Kotlin у цьому аналізі ще практично не згадувався. Java, втім, залишається лідером на другому курсі (CS2), коли студенти переходять до об'єктно-орієнтованого програмування.

На конференції SIGCSE 2024 [11] порівняно когнітивне навантаження при навчанні на Python та спрощеній педагогічній мові Algot. Використовуючи метрику «ментальна ємність» на основі eye-tracking (слідкування за очима) та опитування встановлено, що студенти, які вивчали більш лаконічну мову, витрачали значно менше когнітивних ресурсів на синтаксис і більше - на логіку програми.

Особливої уваги в контексті дослідження заслуговує звіт [12] щодо освітнього використання Kotlin. Станом на 2024 рік понад 300 університетів у всьому світі включили Kotlin до навчальних програм, зокрема 25 із топ-100 університетів за рейтингом Times Higher Education - серед них Стенфорд, Університет Карнегі-Меллон та ряд європейських технічних університетів. Спостерігається чітка тенденція щорічного подвоєння кількості навчальних закладів, що переходять на Kotlin як першу мову в контексті Android розробки.

У дослідженні [13] проаналізовано задоволеність студентів та роботодавців при переході курсів Android розробки з Java на Kotlin у двох університетах Великобританії. Студенти відзначали на 23% вище задоволення від курсу, а представники компаній-партнерів зазначили, що Kotlin випускники потребували приблизно на 30% менше часу для адаптації в IT компаніях.

Водночас зазначені результати не є єдиною усталеною педагогічною позицією. У світовій університетській практиці тривалий час домінувала альтернативна концепція, згідно з якою, першою мовою програмування доцільно обирати саме формальні та синтаксично вимогливі мови, як наприклад Java, C чи C++. Цей підхід ґрунтується на тому, що строга статична типізація, явне оголошення змінних та обов'язкова об'єктно-орієнтована структура змушують студента з перших кроків формувати алгоритмічну дисципліну та розуміти внутрішню організацію програмних систем. Водночас слід враховувати, що за час, який минув від формування

цієї концепції, складність сучасних програмних систем суттєво зросла, а сама природа розробки змістилася. Більшість прикладних задач сьогодні вирішується не низькорівневою реалізацією алгоритмів та структур даних, а компонуванням готових модульних, високорівневих блоків - бібліотек, фреймворків і керованих абстракцій. Це частково послаблює аргумент про необхідність «формальної дисципліни», як обов'язкової бази для сучасного програміста-початківця. Питання доцільності такого підходу в даний час залишається відкритим, а вибір оптимальної першої мови значною мірою залежить від конкретної мети курсу.

Аналіз наведених наукових праць дає можливість констатувати, що лаконічний синтаксис, низький поріг входу і мінімальний обсяг шаблонного коду є маркерами педагогічної ефективності курсів програмування. Попри широке використання Java у вступних університетських курсах, численні дослідження фіксують, що її C-подібний синтаксис і обов'язкова об'єктно-орієнтована структура формують додаткове когнітивне навантаження для початківців. З огляду на це можна стверджувати, що Kotlin цілеспрямовано усуває ті характеристики Java, які дослідники найчастіше ідентифікують як бар'єри для новачків. Відповідно, беручи до уваги актуальні ринкові дані та освітні ініціативи, подальше дослідження доцільно спрямувати на конкретне порівняння Java і Kotlin крізь призму кар'єрних перспектив і педагогічної ефективності для студентів технічних спеціальностей.

Мета дослідження

Метою цього дослідження є визначення, яка з мов, Java чи Kotlin, є більш доцільним вибором, як першої мови програмування для студентів IT-спеціальностей в сучасних умовах. Досягнення мети реалізується шляхом вирішення таких задач:

- проаналізувати динаміку ринку праці та зарплатні тенденції для Java і Kotlin розробників на підставі даних DOU.ua за 2021-2025 роки;
- порівняти синтаксичну складність обох мов крізь призму когнітивного навантаження та кількості необхідних концепцій для виконання типових задач;
- навести порівняльні приклади коду як в контексті Android розробки (на прикладі додатка для електронної комерції), так і в контексті серверної розробки;
- сформулювати обґрунтовані рекомендації для викладачів щодо стратегії вибору мови.

Основний матеріал дослідження

Аналіз ринку праці побудовано на основі п'яти послідовних зарплатних опитувань DOU.ua -найбільшої україномовної платформи для IT-спеціалістів за червень 2021-2025 років [14]. Вибірки охоплюють від 5 829 (2021) до 14 004 (2023) респондентів-IT-фахівців. Дані відображають виключно осіб зайнятих у IT на момент опитування (full-time, part-time, фріланс, контрактори). Зарплата зазначена у доларах США «чистими» після сплати податків. Абсолютна кількість респондентів, що назвали Java або Kotlin основною мовою, наведена в таблиці 1.

Таблиця 1. Кількість респондентів, що назвали Java або Kotlin основною мовою.

Рік	Java, осіб	Kotlin, осіб	Kotlin/Java, %
2021	997	214	21,5%
2022	1 080	304	28,1%
2023	1 206	269	22,3%
2024	919	248	27,0%
2025	832	222	26,7%

Оскільки розмір вибірок суттєво змінюється між роками (від 5 829 у 2021 до 14 004 у 2023) абсолютна кількість респондентів, що назвали ту чи іншу мову основною, не придатна для прямих міжрічних порівнянь. Зміна числа Java розробників відображає лише зміну обсягу

вибірки, а не реальну ринкову динаміку. Тому аналіз проведено по відношенню частки Kotlin респондентів відносно Java респондентів в межах однієї й тієї самої вибірки, на яке розмір опитування впливає в межах статистичної похибки. У 2021 році це відношення становило 21,5%, що становить приблизно один Kotlin розробник на чотирьох-п'ятьох Java розробників.

У 2022 році відношення Kotlin/Java сягнуло максимального значення за весь досліджуваний період - 28,1%. Це збігається з періодом, коли Google перевів усі нові проєкти і документацію виключно на Kotlin. В цей же період вийшов на ринок Jetpack Compose - Kotlin-only UI-фреймворк, що кардинально змінив підхід до побудови Android інтерфейсів[5]. Медіани зарплат у 2022 році вже практично зрівнялися: Java - 3 950 USD, Kotlin - 4 000 USD.

З 2023 по 2025 рік відношення Kotlin/Java коливалося в межах 22-27%. У цей період вибірки опитування у 200 - 300 Kotlin респондентів є незначними, і зміни показника на 4-6 відсоткових пунктів між окремими роками перебувають у зоні статистичної невизначеності. Загальну картину цього періоду формує повномасштабне вторгненням в Україну. Цей фактор пригнічує весь ринок праці незалежно від вибраної мови програмування. Тому єдиним ключовим висновок може бути те, що відношення Kotlin/Java стабільно утримується в діапазоні 22-28% і свідчить про структурну присутність Kotlin у JVM-спільноті. За умови відновлення ринку обидві мови повернуться до зростання, і це відношення, найімовірніше, залишатиметься в діапазоні 25-28%.

Особливо наглядною є динаміка відношення величин заробітної плати для розробників з різним рівнем досвіду (рисунок 1).

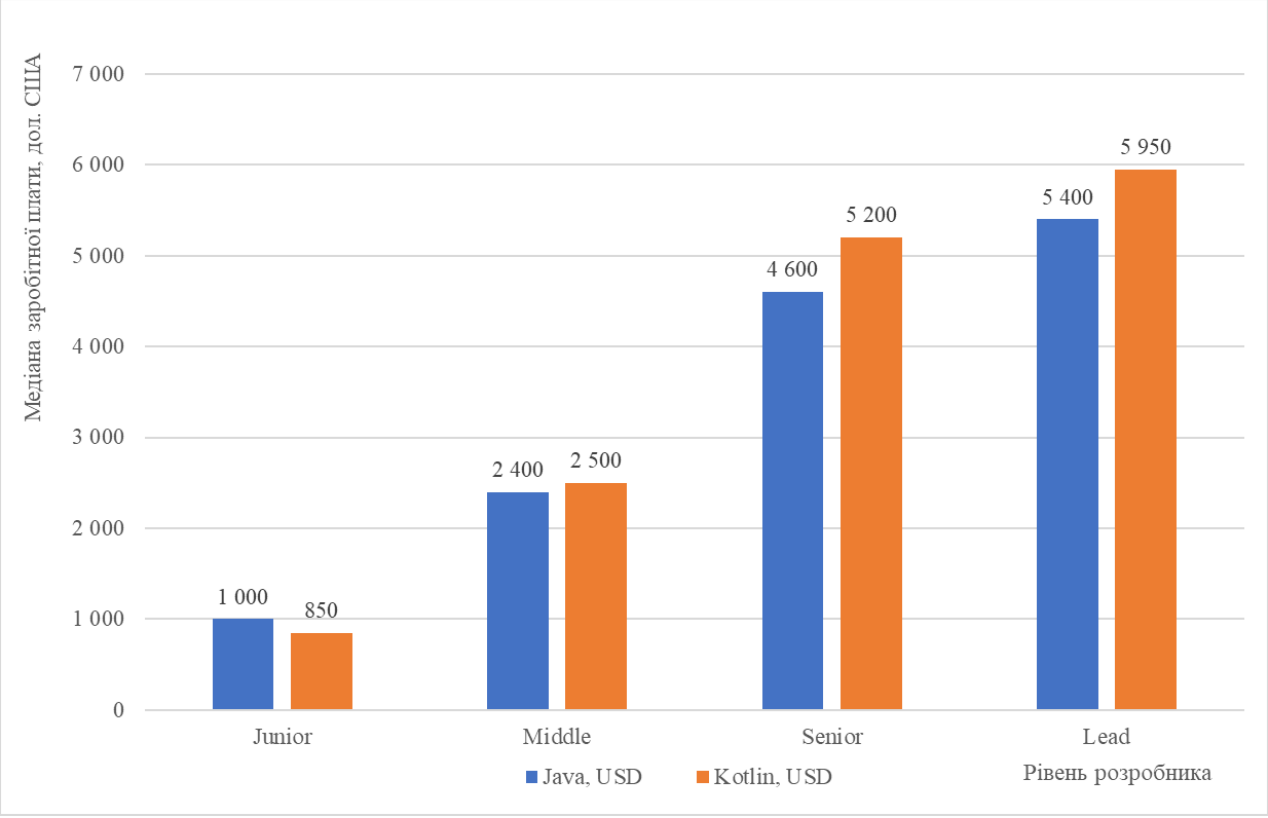


Рис. 1. Медіани заробітних плат IT-розробників в розрізі рівня досвіду, червень 2025р.

Вона демонструє чітку тенденцію до зростання для Kotlin - від -15% на рівні Junior до +13% на рівні Senior. На старті кар'єри такий розробник поступається Java колезі на 15%, однак вже на рівні Middle різниця практично зникає (+4%), а з переходом на Senior рівень Kotlin випереджає Java на 13%. Якщо говорити за Lead рівень, то перевага зберігається і становить +10%. З цього можна зробити висновок, що чим вищий досвід, тим доцільніше знати Kotlin.

Ця закономірність є не випадковою, адже досвідчена Kotlin-спільнота в основному сформувалася шляхом переходу з Java, і такі розробники мають хороші знання з обох мов програмування, що і відображається в рівні оплати праці.

Глобальний контекст підтверджує ці тенденції. За даними JetBrains Developer Ecosystem Survey 2024 [4], Kotlin використовують 16% розробників у світі проти 33% для Java. При цьому 95% із тисячі найпопулярніших Android додатків у Google Play містять код на Kotlin, а 70% написані переважно на Kotlin [5]. Популярний UI-інструментарій Jetpack Compose застосовується вже у 40% із тисячі найпопулярніших додатків (станом на травень 2024 року, проти 24% у травні 2023-го) [15]. Ці цифри наглядно свідчать про те, що в Android розробці ключовою вимогою є необхідність знати Kotlin.

Ці факти підводять до питання, чому Kotlin так стрімко розвивається і велика кількість розробників перейшли на нього, як на основну мову програмування. Для цього проведемо порівняння синтаксису Java і Kotlin для аналізу їх когнітивної складності. Найкраще це можна продемонструвати на конкретних прикладах коду. Розглянемо три типові задачі, з якими часто стикаються розробники як Android додатків, так і backend-розробники.

Перший приклад - створення так званих **POJO** (Plain Old Java Object) класів. Це прості класи з даними, які зазвичай в великих кількостях використовуються в репозиторіях. Для прикладу візьмемо модель даних товару**.** У Java для збереження даних потрібно вручну написати поля, конструктор, геттери, сетери, методи `equals()`, `hashCode()` та `toString()`:

```java
    // Java: модель товару (≈40 рядків)
public class Product {
    private final String id;
    private final String name;
    private final double price;
    private final String imageUrl;

    public Product(String id, String name, double price, String imageUrl) {
        this.id = id;
        this.name = name;
        this.price = price;
        this.imageUrl = imageUrl;
    }

    public String getId() { return id; }
    public String getName() { return name; }
    public double getPrice() { return price; }
    public String getImageUrl() { return imageUrl; }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Product)) return false;
        Product product = (Product) o;
        return Double.compare(product.price, price) == 0
                && id.equals(product.id)
                && name.equals(product.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, name, price);
    }
}
```


ISSN 3083-7618 *Information Technologies and Engineering Electronics No. 4 (2026)*
eISSN 3083-7626

```
    // Kotlin: та сама модель (1 рядок)
data class Product(val id: String, val name: String, val price: Double, val imag
eUrl: String)
```

Kotlin компілятор автоматично генерує equals(), hashCode(), toString(), copy() та деструктурувальні декларації. Студент одразу розуміє що таке «клас даних» - і може рухатись далі, не витрачаючи час на boilerplate код.

Другий приклад - null-безпека. Нехай у нас є задача калькуляції кошику замовлень. Слід врахувати можливість використання промокоду покупцем. При цьому слід врахувати можливість виникнення NullPointerException у Java:

```
    // Java: обов'язкова перевірка на NPE
public double calculateTotal(Cart cart) {
    double total = cart.getSubtotal();
    String promoCode = cart.getPromoCode(); // може повернути null
    if (promoCode != null) {
        Discount discount = discountService.find(promoCode); // теж може null
        if (discount != null) {
            total -= discount.getAmount();
        }
    }
    return total;
}

    // Kotlin: null-безпека гарантована компілятором
fun calculateTotal(cart: Cart): Double {
    val discount = cart.promoCode
        ?.let { discountService.find(it) }
        ?.amount ?: 0.0
    return cart.subtotal - discount
}
```

У Kotlin спроба передати null туди, де очікується non-null тип, спричинить помилку компіляції, а не падіння застосунку в runtime. Для студента-початківця це означає, що велика проблема з NullPointerException вирішується ще на етапі написання коду. Це, безумовно, одна з найважливіших переваг цієї мови.

Як третій приклад розглянемо асинхронне виконання операцій. У Java асинхронність традиційно реалізується через callbacks підхід. Це зачасту призводить до так званого «callback hell», коли велика кількість чи вкладеність колбеків призводить до складної реалізації і дебагу програми. Розглянемо на прикладі HTTP запиту (завантаження списку продуктів):

```
    // Java: вкладені зворотні виклики
apiService.loadProducts(new Callback<List<Product>>() {
    @Override
    public void onSuccess(List<Product> products) {
        imageLoader.preload(products, new Callback<Void>() {
            @Override
            public void onSuccess(Void v) {
                runOnUiThread(() -> adapter.update(products));
            }
            @Override
            public void onFailure(Throwable t) { showError(t); }
        });
    }
    @Override
    public void onFailure(Throwable t) { showError(t); }
});

    // Kotlin Coroutines: лінійний, зрозумілий код
lifecycleScope.launch {
```

```
    try {
        val products = apiService.loadProducts()
        imageLoader.preload(products)
        adapter.update(products)
    } catch (e: Exception) {
        showError(e)
    }
}
```

Kotlin coroutines дають змогу писати асинхронний код у тому ж лінійному стилі, що й синхронний. Для студента це вирішує одну з найскладніших концепцій, а саме вкладені асинхронні виклики. Такий підхід дозволяє більше зосередитись на бізнес-логіці. Наведений приклад показує, що правильне рішення на рівні мови знімає цілий клас архітектурних проблем.

Наведені приклади є тільки невеликою частиною з тих структурних задач, які Kotlin здатен вирішити більш лаконічніше і простіше. Їх сукупний аналіз підтверджує, що незважаючи на сферу застосування, Java вимагає приблизно в 2-4 рази більше рядків коду для реалізації аналогічної функціональності. Це напряму впливає на педагогічний процес - студент повинен спершу засвоїти концепції getters/setters, boilerplate конструкторів і callback патернів, перш ніж написати першу змістовну програму. Kotlin дозволяє зробити це набагато раніше, що підтверджується результатами досліджень [6, 7], де зменшення синтаксичного шуму напряму корелює з кращим засвоєнням матеріалу.

Відмінність між мовами також можна виразити кількісно через поняття «мінімальний концептуальний набір». Це мінімальна кількість конструкцій мови, які студент мусить засвоїти до написання першого практично значущого класу. Наприклад для класу даних, для Java цей набір включає модифікатори доступу (`public`, `private`), статичні члени (`static`), ключове слово `void`, явні типи всіх змінних, поняття `null` і методи `equals()`/`hashCode()`, конструктори та `this`. Лише після цього студент може написати правильну модель класу даних. У Kotlin той самий результат досягається через `data class, типи даних` і `val`/`var` - тобто буквально три нових поняття. Узагальнюючи, на Java-курсі кілька тижнів слід витратити на синтаксичні правила, тоді як на Kotlin-курсі ці тижні можуть бути присвячені алгоритмам, роботі з API або UI.

Слід підкреслити, що лаконічність програмного коду й когнітивне навантаження є тісно пов'язаними поняттями. Кількість рядків коду відображає лише обсяг синтаксичних конструкцій, які необхідно написати. В це й же час когнітивне навантаження визначається більш ширшим колом чинників - кількістю нових понять, які студент засвоює одночасно, рівнем їхньої абстракції, особливостями моделі програмування та психологічними аспектами сприйняття матеріалу. Головним аргументом на користь меншої когнітивної складності Kotlin є саме менша кількість нових концепцій у «мінімальному концептуальному наборі», а скорочення обсягу коду слугує допоміжним додатковим індикатором.

Ключовим аргументом на користь Kotlin є і те, що студент не «замикається» в одній екосистемі. Kotlin повністю сумісний з Java. Будь-яка Java-бібліотека може бути використана в Kotlin-проєкті без змін, а програміст що вивчив Kotlin, може без труднощів читати і підтримувати Java код. Ці мови працюють на JVM і поділяють базові концепції ООП. Вибір Kotlin, як першої мови, не закриває двері перед Java ринком праці. Слід відмітити, що студент який розпочав із Kotlin, значно легше перейде до Java за необхідності. Потрібно буде лиш опанувати boilerplate шаблони, які є типовими для Java розробки, але відсутні у Kotlin.

З точки зору галузевих перспектив, Kotlin охоплює дедалі ширший спектр платформ і парадигм. Kotlin Multiplatform - технологія що дозволяє ділитися бізнес-логікою між Android, iOS, web і desktop. Вона стала одним із найбільш помітних технологічних трендів на даний час. За даними JetBrains [4], частка розробників що використовують KMP зросла з 7% у 2024 до 18% у 2025 році. Це є суттєве зростання за один рік за мірками IT індустрії. У серверній

розробці Kotlin також демонструє впевнене зростання. Офіційна підтримка у Spring Boot і поява Ktor, як легковагового фреймворку для мікросервісів, зробили Kotlin конкурентоспроможною альтернативою Java у backend сегменті. Це відкрило Kotlin доступ не лише до Android, а й до серверної та кросплатформеної розробки. За даними Stack Overflow Developer Survey 2024 [16], Kotlin входить до топ-20 найбільш «бажаних» мов серед тих хто ще не використовує її у своїй роботі, що свідчить про позитивне сприйняття мови розробниками.

Водночас Java залишає за собою ряд беззаперечних переваг. По-перше, обсяг відкритих вакансій для Java розробників на найбільшій IT-платформі в Україні DOU.ua стабільно вищий (Java - 7,1% від усіх IT-спеціалістів проти 1,9% для Kotlin у 2025 році [14]). По-друге, Java є основною мовою більшості legacy backend і enterprise систем. По-третє, розуміння Java є передумовою для роботи зі Spring Framework, Hibernate та іншими широко вживаними технологіями. Варто підкреслити, що ці аргументи не заперечують цінності Kotlin у backend контексті, оскільки Spring Boot повноцінно підтримує Kotlin, а Kotlin -розробник може читати й підтримувати Java сервіси без значних перешкод завдяки JVM-сумісності. Повністю відмовлятися від Java у навчальній програмі не доцільно - але і зводити роль Kotlin виключно до Android розробки є методологічно невиправданим спрощенням.

Проведений аналіз ринкових даних та педагогічних досліджень дозволяє рекомендувати використання Kotlin, як основної першої мови в JVM-треку, незалежно від спеціалізації студента. Це дозволить студентам швидше будувати реальні застосунки і отримувати практичні результати. При цьому слід паралельно знайомити із ключовими концепціями Java (ООП, колекції, дженерики) через порівняльні приклади. Для студентів backend стеку це може бути реалізовано через побудову REST API на Ktor або Spring Boot з Kotlin. Такий підхід не поступається функціональністю і суттєво знижує поріг входу для технології. Він вже реалізований у більш ніж 300 університетах [12] і підтверджується зростаючим попитом на Kotlin розробників у IT компаніях.

Висновки та перспективи подальших досліджень

Аналіз даних DOU.ua за 2021-2025 роки засвідчує, що Kotlin стабільно утримує частку 25-28% відносно Java на українському ринку праці навіть в умовах загального скорочення IT-галузі. За умови відновлення ринку обидві мови повернуться до зростання зі збереженням їхнього відносного балансу. Ця залежність також корелюється і з величиною заробітних плат. На рівні Junior різниця між мовами мінімальна і це знімає ризик «програти» на старті кар'єри. Що стосується рівня Senior - Kotlin демонструє стійку перевагу в 10-13%, яка свідчить, що вивчення мови є вигідною довгостроковою інвестицією для студента.

Ринкова картина підкріплюється результатами порівняльного аналізу коду. Як в Android розробці, так і в серверному контексті Kotlin дозволяє реалізувати аналогічну функціональність у 2-4 рази меншою кількістю рядків. Це відповідає принципам зниження когнітивного навантаження, задокументованим у педагогічних дослідженнях [6, 7, 8]. Додатковим аргументом слугує єдина асинхронна модель Kotlin Coroutines, яка дозволяє в лінійному стилі вирішувати багатопоточні задачі. Такий підхід спрощує роботу з потоками, їх синхронізацією, і дає можливість студентам краще зосередитися на бізнес-логіці, ніж на технічній складовій реалізації задач.

Наведені факти свідчать, що Kotlin виходить за межі платформи Android. Він охоплює 95% із тисячі найпопулярніших Android-застосунків, є єдиною нативною мовою Jetpack Compose і Ktor та офіційно підтримується у Spring Boot. Слід відмітити також і стрімкий розвиток Kotlin Multiplatform, який розширює застосування мови на iOS, web і desktop. Усе це підводить до практичної рекомендації для університетських програм розпочинати вивчення JVM-треку з Kotlin, як головної мови. І такий підхід слід проводити незалежно від того, Android чи backend розробка є основною спеціалізацією. При цьому слід паралельно знайомити студентів із ключовими концепціями Java через порівняльні приклади. Це забезпечить

випускникові конкурентоздатність, як на локальному ринку, де вакансій Java більше, так і на глобальному, де попит на Kotlin-спеціалістів у мобільних та серверних продуктових компаніях стабільно зростає.

Водночас наведені рекомендації базуються на теоретичному аналізі ринку праці, порівняльному розбору коду та результатах наявних педагогічних досліджень, а не на практичному контрольованому педагогічному експерименті у межах цієї роботи. Тому їх варто розглядати як науково обґрунтовану гіпотезу, що потребує подальшої практичної перевірки.

В подальших дослідженнях планується проведення контрольованих педагогічних експериментів із двома групами студентів (Kotlin vs Java як перша мова) із вимірюванням успішності, часу засвоєння матеріалу та рівня когнітивного навантаження за стандартизованими мето

Список літератури:

1. “Mobile Operating System Market Share Worldwide,” StatCounter Global Stats,. [Online]. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide .
2. “Building web applications with Spring Boot and Kotlin,” Spring.io, [Online]. URL: https://spring.io/guides/tutorials/spring-boot-kotlin/
3. “Ktor – Asynchronous Framework for Kotlin,” JetBrains s.r.o., [Online]. URL: https://ktor.io/
4. “The State of Developer Ecosystem 2024,” JetBrains s.r.o., 2024. [Online]. URL: https://www.jetbrains.com/lp/devecosystem-2024/ .
5. “Android Kotlin-first,” Google LLC, Android Developers Blog, 2019. [Online]. URL: https://developer.android.com/kotlin/first .
6. A. Stefik and S. Siebert, “An empirical investigation into programming language syntax,” ACM Trans. Comput. Educ., vol. 13, no. 4, pp. 1–40, Nov. 2013. doi: https://doi.org/10.1145/2534973 .
7. J. Lo, Y. Lin, and C. Wu, “Comparing learning outcomes of programming courses using Python vs Java as the first programming language for engineering undergraduates,” Comput. Educ., vol. 82, pp. 217–227, Mar. 2015.
8. J. Wainer and E. C. Xavier, “Empirical evidence that Python is better than Java as an introductory programming language,” in Proc. 48th ACM Tech. Symp. Comput. Sci. Educ. (SIGCSE), Seattle, WA, USA, 2017, pp. 459–464.
9. S. Xinogalos, M. Satratzemi, and A. Chatzigeorgiou, “An introduction to object-oriented design and the role of the programming language: A comparative study of Java, C++ and Kotlin,” in Proc. 23rd Annu. Conf. Innov. Technol. Comput. Sci. Educ. (ITiCSE), Larnaca, Cyprus, 2018, pp. 116–121.
10. R. M. Siegfried, S. Dorogoff, and S. Cohen, “Introductory programming languages in the post-APCS era,” J. Comput. Sci. Coll., vol. 37, no. 2, pp. 40–49, Oct. 2021.
11. A. Thorgeirsson, S. Steingrímsson, and Ó. Halldórsson, “Cognitive load in introductory programming: A comparison of Python and Algot,” in Proc. 55th ACM Tech. Symp. Comput. Sci. Educ. (SIGCSE), Portland, OR, USA, 2024, vol. 1, pp. 891–897. .
12. “Kotlin for Education,” JetBrains s.r.o., 2024. [Online]. URL: https://kotlinlang.org/education/
13. A. Smith and B. Johnson, “Student satisfaction and industry readiness after migrating Android development courses from Java to Kotlin,” J. Inf. Technol. Educ.: Res., vol. 21, pp. 283–301, 2022.
14. “Зарплатний звіт DOU.ua (червень 2021–2025),” (in Ukrainian), DOU.ua, GitHub repository, Apr. 2026. [Online]. URL: https://github.com/devua/csv/tree/master/salaries/
15. “Now in Android: 112,” Google LLC, Android Developers Blog, May 2024. [Online]. URL: https://android-developers.googleblog.com/2024/05/now-in-android-112.html
16. “Developer Survey 2024,” Stack Overflow, 2024. [Online]. URL: https://survey.stackoverflow.co/2024/

Надійшла до редколегії 29.05.2026

Відомості про авторів:

Ксенич Андрій Іванович - кандидат технічних наук, доцент, доцент кафедри інженерії програмного забезпечення, Івано-Франківський національний технічний університет нафти і газу / Ivano-Frankivsk National Technical University of Oil and Gas, Україна; email: andrii.ksenych@nung.edu.ua; ORCID: https://orcid.org/0009-0007-6569-951X