Зміст

Передмова

Про книгу й автора

Навчальний посібник присвячено мові програмування Java і її використанню в контексті створення кросплатформних застосунків. Посібник орієнтовано на навчальний курс "Кросплатформне програмування", який викладається у 5-му семестрі студентам третього курсу бакалаврату факультету інформаційних технологій Київського національного університету імені Тараса Шевченка. Ідеологія курсу ґрунтується на ідеї, що найнадійнішими способом створення справді кросплатформних застосунків є використання технологій, які в своїй основі є кросплатформними. Технологія Java у цьому сенсі є першим і найпріоритетнішим кандидатом.

Про що ця книга

Фактично, цю книгу присвячено мові програмування Java з урахуванням її універсальності, ефективності, надійності, і, як зазначалося, кросплатформності. Тому перше, про що йтиметься у цьому навчальному посібнику – це мова Java як така. Другий важливий момент, на якому ми зосереджуватимемося, пов’язаний з універсальністю кодів, написаних мовою Java. Тому значна (відносно значна) увага в посібнику приділяється загальній концепції мови (і технології) Java, принципам використання програмних кодів та універсальним можливостям мови. Іншими словами, йтиметься не просто про мову програмування Java, а про ті аспекти та особливості мови, які є наріжними з точки зору створення саме універсальних (кросплатформних) програм. Нарешті, теми, які розглядаються в посібнику, з одного боку перекривають весь основний спектр питань, необхідних для опанування мови програмування Java, а з іншого боку, акцент робиться на тих областях і напрямках застосування мови Java, де вона є найефективнішою при створенні кросплатформних застосунків. Це одна з причин, чому в посібнику значна увага приділяється створенню застосунків з графічним інтерфейсом.

Також не останню роль відіграє той факт, що на сьогоднішній день Java є не єдиною технологією, яка використовується в кросплатформному програмуванні. З моменту, коли світ познайомився з Java й унікальними можливостями (причому не лише в плані кросплатформності), які ця мова запропонувала розробникам, з’явилось багато цікавих і різнопланових рішень – у тому числі таких, що передбачають використання інших популярних мов програмування та відповідних бібліотек і фреймворків. Тому при поданні матеріалу в книзі доволі часто виконується порівняння тих або інших концепцій і підходів з аналогічними реалізаціями в інших мовах програмування – в першу чергу маються на увазі такі мови програмування, як C++ та C#. Такий підхід стане корисним не лише читачам, знайомим з цими мовами, але й відносним новачкам у програмуванні.

Для кого ця книга

Книга, з одного боку, розрахована на широке коло читачів, які бажають вивчити мову програмування Java в рамках певного навчального курсу (наприклад, "Об'єктно-орієнтовне програмування", "Кросплатформне програмування" чи "Програмування мовою Java") або самостійно. Але найкраще вона підійде тим, хто має попередній досвід програмування в інших об'єктно-орієнтованих мовах програмування (таких, наприклад, як C++ та C#). Такий досвід міг би стати значним фундаментом для знайомства з мовою Java. Хоча відразу слід зазначити, що цей досвід і не є обов'язковим. Тим не менш, правильним було би твердження, що книга вимагатиме від читача певної самостійності та наполегливості. Це не повинно лякати, оскільки вивчення чогось нового, особливо в галузі технічних наук, завжди вимагає зусиль. Добра новина полягає в тому, що ці зусилля, у нашому конкретному випадку, мають бути доволі помірними.

Як уже зазначалося, посібник містить необхідний теоретичний матеріал і значну кількість прикладів, які цей теоретичний матеріал підкріплюють. Отже, у читача є можливість не лише вивчати теорію, але й застосовувати її на практиці. Ба більше, саме такий підхід у вивченні програмування видається найефективнішим. Тому очікується, що читачі, за бажання засвоїти курс, будуть готові ще й до самостійної роботи з програмними кодами. У цьому сенсі посібник забезпечує необхідною інформацією, але успішність подальших кроків у вивченні Java та кросплатформного програмування багато в чому залежить від наполегливості та працездатності читача. Хочеться вірити, що цих якостей не бракуватиме для подолання складного але цікавого шляху з вивчення технології Java.

Про автора

Автор книги – *Васильєв Олексій Миколайович*, професор кафедри програмних систем і технологій факультету інформаційних технологій Київського національного університету імені Тараса Шевченка, доктор фізико-математичних наук, автор книг з програмування та математичного моделювання. Сфера наукових інтересів: квантитативна лінгвістика, біофізика, математична економіка та моделювання соціально-політичних процесів.

Зворотний зв'язок

Читачі можуть висловити свої зауваження та пропозиції стосовно цієї або інших книг автора, надіславши листа на електронну пошту `oleksii@vasyliev.kyiv.ua`. Конструктивна критика вітається. Інформацію про книги, які видавались раніше, та деякі матеріали прикладного й навчального характеру (які стосуються

книг і не лише) можна знайти на сайті www.vasyliev.kyiv.ua та на YouTube-каналі www.youtube.com/@oleksii.vasyliev.

Подяки

Автор вдячний читачам, студентам та слухачам курсів за їхню увагу, цікавість до книг і готовність до навчання.

Вступ

Про навчальний курс

Як уже зазначалося, навчальний посібник ґрунтується на курсі "Кросплатформне програмування", прочитаному для бакалаврів факультету інформаційних технологій Київського національного університету імені Тараса Шевченка. У рамках цього курсу вивчається мова програмування Java, однак акцент у вивченні мови та відповідної технології робиться саме на кросплатформності. Почнемо ми власне з кросплатформного програмування, окресливши його особливості як технології і як навчальної дисципліни.

Кросплатформне програмування як технологія

У світі сучасних програмних технологій кросплатформне програмування посідає особливе місце. Причин декілька. Головна пов'язана з тим, що в рамках кросплатформного програмування створюються програмні продукти, які здатні працювати на різних платформах, таких як Windows, Linux, macOS, Android, iOS тощо. Сам по собі цей факт означає, що кросплатформні застосунки розраховано на широку аудиторію, а, отже, вони мають достатній рівень універсальності та є перспективними в прикладному плані. Фактично, мовиться про універсальність і масштабованість. Відповідні програми можуть використовуватися широким колом користувачів незалежно від операційної системи. Вони легко масштабуються та адаптуються до нових платформ, забезпечують довший життєвий цикл програмного продукту.

Окрім цієї очевидної переваги, кросплатформність уможливлює зменшення витрат на розроблення, тестування та підтримку програмного забезпечення, оскільки знімає необхідність (або зводить її до мінімуму) у створенні окремих версій програмного продукту для різних платформ. Як наслідок, команди розробників працюють над єдиним проєктом замість кількох окремих. Це, зрозуміло, скорочує час виходу продукту на ринок, що є вкрай важливим, наприклад, для стартапів чи освітніх проєктів.

Також в рамках кросплатформного підходу вдається зменшити залежність від конкретного виробника операційної системи та апаратної архітектури. Ця обставина є фундаментально важливою для критичних систем у науці, освіті, сфері національної безпеки.

Використання кросплатформного коду дозволяє, за рахунок залучення єдиної кодової бази, спростити контроль якості програмного забезпечення, полегшує пошук помилок та автоматизованого тестування. Додатково, кросплатформні програми забезпечують доступ до сервісів з різними пристроями, платформами, архітектурами.

До уваги

До питання щодо особливостей кросплатформних технологій ми повернемося в основній частині книги.

Кросплатформне програмування як навчальна дисципліна

Важливість дисципліни "Кросплатформне програмування" випливає, з одного боку, з важливості кросплатформного програмування як технології. Однак відповідний список не є вичерпним.

Окрім очевидних бонусів, вивчення кросплатформного програмування формує універсальні навички програмування і привчає зосереджувати увагу не на особливостях конкретної платформи чи операційної системи, а на реалізації алгоритмів та архітектури.

Вивчення кросплатформного програмування як навчальної дисципліни формує універсальне інженерне мислення без прив'язки до однієї мови програмування, операційної системи чи середовища виконання. Тому в певному сенсі кросплатформне програмування можна розглядати як дисципліну міждисциплінарну й стратегічно важливу для сучасних програмних технологій. Серед важливих факторів, які виділяють кросплатформне програмування на фоні інших програмних дисциплін, можна згадати такі:

- Першочергова орієнтація на програмні принципи, а не на платформу.
- Абстрагування щодо апаратного рівня та операційної системи.
- Широке використання віртуальних машин і супутніх технологій.
- Використання стандартизованих підходів та універсальних методик.

У методичному плані ті, хто вивчають кросплатформне програмування, навчаються створювати переносні програми замість того, щоб створювати програми під конкретну операційну систему. При цьому відбувається поєднання теорії та практики. Адже така дисципліна, як кросплатформне програмування, органічно об'єднує різні компоненти, як-то розроблення застосунків для різних операційних систем з єдиною кодовою базою, збирання, розгортання і тестування програмних продуктів на різних платформах, аналіз відмінностей поведінки програми в різних середовищах. Також вивчення кросплатформного програмування сприяє формуванню системного мислення, навчає розглядати програму як систему, що складається з логіки, інтерфейсу, середовища виконання. Опанування кросплатформного програмування формує навички з відокремлення платформозалежного коду, мінімізації таких залежностей, проєктування модульних та масштабованих рішень.

Зазвичай в рамках курсу з кросплатформного програмування вивчають одну або декілька технологій. Серед найперспективніших можна вказати такі:

- Мови програмування з вбудованою кросплатформною підтримкою (наприклад, Java, C#, JavaScript чи Python).
- Кросплатформні фреймворки та платформи (скажімо, .Net, Qt чи Flutter).
- Сучасні кросплатформні інструменти збірки й керування проєктами.

При цьому акцент робиться на переносимість програмного коду, його супровід та масштабування. У порівнянні з іншими базовими курсами програмування, дисципліна передбачає знайомство з довготривалою підтримкою програмного продукту, повторним використанням коду, методами визначення економічної

доцільності застосування кросплатформних рішень. Такий підхід сприяє підготовці програмно-інженерних кадрів, які за фаховим рівнем відповідають сучасним вимогам програмної індустрії.

Отже, кросплатформне програмування як навчальна дисципліна моделює реальні виробничі сценарії, навчає працювати з різними середовищами виконання, готує до командної роботи. Досить часто вивчення цього курсу стає переходом від створення навчального коду до промислового програмування.

На додаток, дисципліна "Кросплатформне програмування" інтегрує знання з алгоритмів і структур даних, операційних систем, комп'ютерних мереж, програмної інженерії, архітектури програмного забезпечення.

Структура книги

Книга складається зі вступу, тринадцяти розділів, післямови та списку літератури.

У першому розділі обговорюються загальні принципи кросплатформності. Ми дізнаємось, що таке кросплатформність, як і за рахунок чого вона досягається, проаналізуємо мови програмування з кросплатформною підтримкою, та розглянемо механізми, за допомогою яких кросплатформність реалізується в Java.

У другому розділі ми познайомимось з мовою програмування Java. Цей розділ багато в чому є оглядовим. Зокрема, там дається коротка історія мови Java, визначаються її особливості, пояснюються способи компіляції Java-програм, відбувається знайомство з простими (і не дуже простими) Java-кодами. Призначення цього розділу подвійне – по-перше, познайомити читача з основними прийомами створення Java-програм, а, по-друге, дати загальне уявлення про структуру та характер застосунків Java.

У третьому розділі розглядаються примітивні типи даних та базові оператори. Крім примітивних типів даних, у цьому розділі розглядаються методи роботи з текстом. Також там йдеться про літерали, змінні та приведення типів, арифметичні, логічні, побітові оператори та оператори порівняння, тернарний оператор та оператор присвоєння.

Четвертий розділ присвячено керуючим інструкціям Java. Мовиться про умовний оператор `if`, оператор вибору `switch`, оператори циклу `while`, `do-while` та `for`, а також про оператор циклу за колекцією.

У п'ятому розділі обговорюються масиви. А саме, там визначаються принципи використання масивів, розповідається про методи створення одновимірних масивів, розглядаються двовимірні масиви та аналізуються деякі важливі операції, які зазвичай виконуються з масивами.

Класи та об'єкти розглядаються в шостому розділі. Там розповідається, як описувати класи в Java, як перевантажувати методи та конструктори, а також мовиться про статичні поля та методи.

У сьомому розділі розглядається такий важливий механізм об'єктно-орієнтованого програмування, як спадкування. У контексті спадкування пояснюються особливості створення конструкторів підкласів, визначаються рівні доступу членів класу, пояснюється переозначення методів, а також

віртуальність методів та конструкторів. Крім того, увага приділяється заміщенню полів та особливостям спадкування стосовно закритих членів.

У восьмому розділі обговорюються абстрактні класи та інтерфейси. Серед іншого, там розповідається як створюються абстрактні класи та інтерфейси, як розширюються інтерфейси, як створюються анонімні класи. Крім цього, увагу приділено функціональним інтерфейсам та лямбда-виразам та описуються методи роботи з посиланнями на методи.

Винятковим ситуаціям присвячено дев'ятий розділ. Там ми дізнаємось про принципи оброблення винятків, вкладені конструкції `try-catch-finally`, штучне генерування винятків, контрольовані та неконтрольовані винятки, а також навчимося створювати користувацькі класи винятків.

У десятому розділі розповідається про принципи багатопотокового програмування в Java. Там ми познайомимось зі створенням потоків, дізнаємось, які операції з ними можна виконувати, що таке демон-потоки та як виконується синхронізація потоків.

Узагальненим типам присвячено одинадцятий розділ. Там розглядаються принципи реалізації узагальнених типів, обмеження на параметри типу, узагальнені методи та інтерфейси та використання підстановок.

У дванадцятому розділі розглядаються стандартні колекції та методи роботи з ними. Там мовиться про основні інтерфейси й класи колекцій, їхню імплементацію при створенні списків, множин, стеків та словників. Також в розділі дається інформація про ітератори та компаратори.

У тринадцятому розділі описуються методи створення застосунків з графічним інтерфейсом. Там (доволі коротко, в режимі ознайомлення) розповідається про те, як створювати графічні компоненти, як їх розміщувати в контейнерах та як виконувати оброблення подій.

Післямова підсумовує матеріал, який розглянуто в книзі, та фокусує увагу читача на планах та перспективах. На завершення книги пропонується список літературних джерел, які можуть стати в нагоді для продовження знайомства з мовою програмування Java.

Кожен розділ книги закінчується списком питань, які охоплюють матеріал відповідного розділу. Основне призначення цих питань полягає в першу чергу в тому, щоб звернути увагу читача на моменти, які є важливими при вивченні матеріалу розділу.

Матеріал книги подається, з одного боку, максимально просто, а з іншого, його структуровано – так, щоб читач мав можливість не лише отримати потрібну інформацію, але й міг скласти певне уявлення про місце та роль Java-технологій серед сучасних мов програмування. При цьому певна інформація, залежно від призначення та специфіки, подається у спеціальних блоках.

Довідка

У такому блоці подається інформація загального характеру, яка покликана розширити уявлення читача про сучасні програмні технології та базові підходи щодо створення програмних продуктів.

До уваги

У такому блоці міститься інформація, яку слід прийняти до уваги в процесі вивчення тої чи іншої теми чи питання.

Деталі

У такому блоці продається інформація технічного характеру або інформація, яка має радше прикладне, ніж теоретичне значення.

Кросплатформність Java

Оскільки книгу присвячено не просто мові програмування Java, а вивченню мови програмування Java в контексті кросплатформного програмування, то та інформація, яка має прямий або опосередкований стосунок до кросплатформності мови Java (в основному прикладного характеру) міститься в спеціальних блоках. Але в основному в таких блоках подається, для порівняння, результат виконання тої чи іншої програми в різних операційних системах (враховуючи популярність і розповсюдженість, було обрано операційні системи Windows та Ubuntu). Якщо явно не вказано, то результат виконання програми (вигляд діалогових вікон) подається для випадку використання операційної системи Windows.

Окрім власне теорії, книга містить певну кількість готових до використання прикладів, які допоможуть читачеві сформувати уявлення про можливості й ефективність мови програмування Java.

Програмне забезпечення

У процесі роботи з прикладами з книги знадобиться програмне забезпечення, яке уможливить перевірку функціональності запропонованих програм. Далі дається досить коротка довідка щодо програмного забезпечення, яке може знадобитись (а якщо відверто, то точно знадобиться) читачеві.

Традиційний спосіб роботи з Java передбачає використання середовища розроблення (або IDE від Integrated Development Environment), яке акумулює значний функціонал і дозволяє виконувати різні підзадачі (як-то редагування коду, його налагодження, компіляцію та перевірку результату) у режимі "єдиного вікна". На сьогодні існує певна кількість якісних середовищ розроблення, оптимізованих для роботи з Java, але, мабуть, найпопулярнішим є IntelliJ IDEA від компанії JetBrains. Відповідний програмний продукт має як комерційну, так і безкоштовну версії. Обидві можна завантажити на сайті `www.jetbrains.com` у відповідному розділі (там пропонуються різні IDE для різних технологій, тому необхідно вибрати потрібний продукт і версію).

Однак цього замало. Попередньо, перед встановленням IDE, необхідно встановити пакет утиліт, до якого, крім іншого, входять компілятор та віртуальна машина (Java Virtual Machine, про яку мовиться далі в книзі). Все це ховається за абревіатурою JDK (від Java Development Kit).

Деталі

"Правильна" послідовність така, що спочатку встановлюється JDK, а потім IDE. У принципі, можна вчинити й навпаки (спочатку встановити IDE, а вже потім JDK), але тоді частину налаштувань доведеться виконати вручну.

Файли для завантаження та встановлення системи JDK слід шукати на сайті `www.oracle.com` корпорації Oracle. А саме, необхідно перейти на розділ

завантажень (розділ підтримки розробників) і вибрати необхідну версію JDK для завантаження та встановлення.

Деталі

При виборі програмного забезпечення для встановлення вибираємо операційну систему, платформу (а саме, нас цікавить SE від Standard Edition) та версію Java. На момент написання книги нумерація версій перетнула позначку 20, але для ефективної роботи й використання більшості запропонованих програмних кодів достатньо встановити версію Java не меншу за Java 8. У випадках, коли для коректного виконання коду з книги необхідна вища версія Java, у тексті та поясненнях до прикладів зазвичай даються відповідні коментарі.

Розділ 1

Принципи кросплатформності

У цьому розділі йтиметься про загальні підходи, які використовуються (чи можуть використовуватися) для створення кросплатформних застосунків. Ми проаналізуємо загальні тенденції, розглянемо базові підходи та технології, а також згадаємо про мову Java у контексті кросплатформності.

Що таке кросплатформне програмування

Якщо переглянути зміст різних навчальних курсів, об'єднаних назвою "Кросплатформне програмування", можна виявити, що там спостерігається значне різноманіття, починаючи від власне структури того чи іншого курсу, і закінчуючи темами й технологіями, які розглядаються в рамках курсу. Так що ж таке кросплатформне програмування? Насправді це не настільки просте питання, як видається на перший погляд. Історично так склалось, що коли ми говоримо про кросплатформне програмування, то мається на увазі певний підхід щодо розроблення програмного забезпечення, завдяки якому це програмне забезпечення зможе працювати на різних платформах. Така відповідь породжує інше запитання: а що таке "різні платформи"? Зазвичай під різними платформами мають на увазі різні операційні системи (наприклад, Windows, macOS чи Linux). Але "зазвичай" – це не те саме, що "завжди".

До уваги

Чому важливо писати кросплатформні коди й чому взагалі ця тема є важливою – в принципі, зрозуміло. У сучасному світі платформ дуже багато, навіть якщо говорити про операційні системи. Але проблема геть не в тому, що їх якась кількість. Проблема в тому, що коли ми створюємо сучасне програмне забезпечення, то не завжди знаємо, яка платформа використовуватиметься кінцевим споживачем. А це означає, що коли створюється код, маємо розглянути всі варіанти.

Якщо зробити мінімальний історичний екскурс, то питання створення кросплатформних застосунків спочатку дійсно зводилось в основному до реалізації таких кодів, які можна було би запускати без змін (чи з мінімальними змінами) на різних операційних системах. Однак неухильний технічний прогрес і бурхливий розвиток програмних технологій не лише розширив наші можливості в плані створення програмних застосунків, але й значно збільшив кількість різноманітних платформ, на яких ці застосунки в принципі можуть виконуватись. У контексті цього різноманіття питання кросплатформності застосунків засяяло новими фарбами, а відповідь на запитання "Що таке кросплатформність?" стає доволі неоднозначною. Тому зазвичай використовують відповідь просту й універсальну. Полягає вона в тому, що будь-яка технологія, яка дозволяє створювати програмні продукти, які функціонують

хоча б на двох різних платформах, може вважатись кросплатформною. Власне від цього визначення кросплатформності ми й відштовхуватимемось.

Ще один важливий аспект щодо кросплатформності стосується того, чому це питання є актуальним (як для вивчення, так і для практичного застосування). Основний і найочевидніший ефект – це зменшення зусиль для створення та підтримки програмних застосунків для роботи на різних платформах. Адже в ідеалі немає потреби створювати окремий код для кожної платформи. Відповідно, підтримка працездатності коду також вимагатиме менше зусиль.

Деталі

Якщо на загал аналізувати сучасні тенденції щодо розвитку програмного забезпечення, то є очевидними декілька позицій. А саме, постійно зростає кількість застосунків і програмних систем. Збільшуються обсяги програмного коду й зростає складність програм, складнішими стають технології. Відповідно, зростає вартість розроблення. Як з цим боротись? Рішення також є очевидними. Слід стандартизувати підходи, уможливити багатократне використання вже існуючих рішень, застосовувати гнучке комбінування різних компонентів. Ідея кросплатформності дуже добре узгоджується з цими підходами.

Крім того, важливим бонусом кросплатформності є розширення потенційної аудиторії користувачів програмного продукту. Адже характерною особливістю сучасного стану речей в програмній індустрії є широке різноманіття платформ чи операційних систем, які використовують кінцеві споживачі в своїй роботі.

Довідка

Тут можна згадати історію, яка називається "браузерними війнами" чи "війнами браузерів". Свого часу виробники браузерів боролись за "місце під сонцем" – тобто за домінування на ринку. І це створювало певні проблеми для розробників, оскільки за відсутності єдиного уніфікованого стандарту доводилось створювати сценарії та писати коди гіпертекстової розмітки з урахуванням можливого використання різних браузерів, кожен з яких мав свої особливості.

Перша "браузерна війна" відбувалась з 1995 по 1999 рік. Основними контрагентами були браузери Microsoft Internet Explorer та Netscape Navigator. У результаті останній був фактично витіснений з ринку. Друга "браузерна війна" проходила з 2005 по 2015 рік і закінчилась перемогою Google Chrome над Microsoft Internet Explorer.

Якщо на цю ситуацію подивитись з точки зору реалізації кросплатформного підходу, то наявність "переможця" фактично означає наявність певного стандарту, який домінує на ринку. Орієнтуючись у процесі розроблення програмного забезпечення на цей стандарт можна задовольнити потреби переважної кількості користувачів. А от якщо переможця немає (чи поки що немає), а є сильні конкуруючі платформи, то створення ефективних застосунків під ці платформи видається дещо проблематичним, оскільки різні користувачі використовують різні платформи, а програмний продукт має працювати на кожній з них.

Як би там не було, але в принципі питань щодо актуальності розвитку кросплатформних технологій зазвичай не виникає. Та залишається інше питання: а як же її досягти? Про це поговоримо далі.

Методи досягнення кросплатформності

Існує декілька фундаментальних підходів та технологій, які можуть використовуватися для досягнення кросплатформності програмних застосунків.

• Першою чергою це мови програмування з кросплатформною підтримкою. Зазвичай в таких мовах програмування забезпечення кросплатформності застосунків вирішується на етапі компіляції програми або за рахунок структури застосунку. Причому підтримка кросплатформності може бути як повною (чи майже повною), так і частковою. Прикладами таких кросплатформних мов є Java, Python, C#, які дозволяють розробникам створювати програми, здатні працювати на різних операційних системах без потреби переписувати код під кожну конкретну платформу.

Деталі

Кросплатформність мови Python в першу чергу пов'язана з реалізацією інтерпретатора та стандартної бібліотеки. Оскільки Python належить до інтерпретованих мов, то програма, написана мовою Python, виконується не безпосередньо процесором, а інтерпретатором. Існують Python-інтерпретатори для більшості популярних операційних систем. Тому один і той самий код Python може виконуватися на будь-якій системі, де встановлено відповідний інтерпретатор. Хоча, зрозуміло, при цьому існують певні обмеження щодо універсальності технологій, які використовуються в програмі.

Кросплатформність мови C# пов'язана з використанням сучасної платформи .Net, що уможливлює виконання однієї й тої самої програми на різних операційних системах. Хоча на початку історії платформи .Net вона була орієнтована суто на використання операційної системи Windows, з виходом .Net Core компанія Microsoft зробила мову C# кросплатформною з підтримкою низки технологій, серед яких можливість використання різних операційних систем, створення мобільних додатків (наприклад, з використанням .Net MAUI) та веб-застосунків (наприклад, з використанням фреймворку ASP .Net Core).

Кросплатформність мови (і відповідної технології) Java реалізується завдяки використанню віртуальної машини Java (JVM) і компіляції програм у проміжний байт-код. Такий підхід забезпечує незалежність програм від конкретної операційної системи чи апаратної архітектури. Детальніше особливості мови Java та природу її кросплатформності ми обговорюватимемо згодом.

• Ще один напрямок реалізації кросплатформності ґрунтується на використанні кросплатформних фреймворків та бібліотек. Мовиться про спеціальні фреймворки та бібліотеки, які дають змогу розробникам створювати програми, що працюють на різних платформах. Прикладами таких фреймворків можуть бути React Native, Flutter чи Xamarin та його наступник – технологія .Net MAUI. Такий підхід не завжди є універсальним у тому сенсі, що кросплатформність може бути доволі обмеженою (щодо кількості різних платформ, на яких працюють застосунки). Разом з тим, відповідні рішення знаходять свою нішу на ринку програмних продуктів і успішно використовуються на практиці.

Деталі

Фреймворк React Native від Facebook (Meta) використовується для створення мобільних застосунків, які працюють на операційних системах Android та iOS і при цьому створюються мовою JavaScript (чи TypeScript). Ключовим елементом у реалізації кросплатформності є так званий "міст" – це компонент, який пов'язує код JavaScript з нативними компонентами Android (Java/Kotlin) та iOS (Objective-C/Swift). При цьому JavaScript-логіка виконується в окремому потоці. Через "міст" JavaScript надсилає команди нативним модулям. Результати виконання цих команд повертаються JavaScript. Завдяки цьому реалізується взаємодія з реальними елементами операційної системи (маються на увазі кнопки, текст, сенсори) однак без написання нативного коду.

Фреймворк Flutter від Google використовує мову Dart і підтримує, серед іншого, операційні системи Android та iOS. Фреймворк Flutter адаптує інтерфейс під кожну операційну систему компілюючи вихідний Dart-код у нативний код. Тому кінцевий додаток є нативною програмою без проміжних шарів або емуляції.
Кросплатформність у Xamarin та .Net MAUI реалізується на схожих принципах, що дозволяє створювати мобільні та десктопні застосунки для різних платформ, використовуючи один спільний код мовою C#. Якщо фреймворк Xamarin орієнтовано на розроблення кросплатформних мобільних застосунків для операційних систем Android та iOS, .Net MAUI об'єднує мобільні та десктопні платформи в єдину кросплатформну систему.

• Важливим елементом кросплатформності є віртуалізація та контейнеризація. За допомогою віртуалізації (наприклад, шляхом використання віртуальних машин) або контейнеризації (такої, скажімо, яка реалізується використанням Docker), розробники можуть створювати ізольовані середовища для програм, які мають однакову робочу конфігурацію на різних платформах.

• Ще одним способом чи напрямком реалізації кросплатформності є створення веб-додатків. Мовиться про додатки, які працюють у браузерах, доступних на багатьох операційних системах.

• Якщо застосунок не є кросплатформним але при цьому необхідно, щоб він працював на різних платформах, доведеться під кожну платформу створювати нативний код. Фактично, для кожної платформи створюється окремий застосунок. Така ситуація може бути непереборною в тому сенсі, що використовувані з необхідністю технології не дозволяють застосувати ефективний кросплатформний підхід щодо створення застосунків. Але навіть за таких обставин певного кросплатформного ефекту можна досягти за рахунок використання загальних абстракцій на рівні архітектури застосунку. Тобто можна використовувати нативний код для кожної платформи, але одночасно створювати абстракції та спільний код для загальних функцій. Це допоможе суттєво зменшити обсяг роботи шляхом уникнення непотрібного дублювання.

Цей перелік, зрозуміло, не є вичерпним. Однак він дає певне уявлення щодо шляхів, якими можуть реалізовуватися кросплатформні програми. Кожен з цих підходів має свої переваги та недоліки, і очевидно, що універсальних технологій не існує (хоча й існують більш ніж ефективні технології). Вибір того або іншого підходу залежить від призначення та конкретних вимог проєкту, наявних ресурсів та пріоритетів розробників.

Мови з кросплатформною підтримкою

Вище вже згадувались деякі мови програмування, які мають у своєму "арсеналі" засоби для розроблення й підтримки кросплатформних проєктів.

До уваги

Причини, в силу яких та або інша мова вважається кросплатформною, є доволі різними. Різними є й рівні кросплатформності. Також сама по собі мова, якою вона б не була, – це лише мова програмування. Коли згадується кросплатформність, явно чи неявно маються на увазі ті технології, які стоять за мовою програмування і які базово використовуються при створенні

програм. Все це означає, що поділ мов програмування на кросплатформні та некросплатформні є досить умовним. Однак важливо розуміти, що кросплатформність, реалізована на рівні мови, зазвичай є найфундаментальнішим підходом, який дозволяє відносно просто створювати застосунки з кросплатформними характеристиками.

Далі ми коротко розглянемо основні мови програмування, які в силу тих або інших обставин чи за певних умов можна віднести до мов з кросплатформною підтримкою. Деякі з них ми вже згадували раніше.

Кросплатформність Java

Нагадаємо, що книгу присвячено Java як мові кросплатформного програмування. Тому тут згадуються лише основні властивості та особливості цієї мови. Детальніше вони розглядатимуться згодом.

• Отже, першою в списку кросплатформних мов програмування є, безумовно, мова Java. Це одна з найпопулярніших мов програмування, і кросплатформність є однією з її візитівок. Програми, написані мовою Java, компілюються в байт-код, який виконується віртуальною машиною Java (JVM). Це уможливлює виконання програми на різних операційних системах без необхідності їх перекомпільовувати. Справа в тому, що особливості платформи враховує віртуальна машина, яка встановлюється на платформу один раз і потім виконує байт-коди, які є універсальними (в тому сенсі, що в них не враховуються особливості тої чи іншої платформи).

Довідка

Девіз мови Java: "Написано один раз – працює скрізь". Зрозуміло, що це не завжди так, але тим не менш, доволі близько до істини.

Мова Java задумувалась саме як кросплатформна. Правда, початкова ідея дещо відрізнялась від того, що вийшло в підсумку, однак на загал вийшло доволі добре. Як наслідок, сьогодні в розробників є такий потужний, безпечний і кросплатформний інструмент, як Java (і суміжні до неї технології).

• Кросплатформність є однією з переваг мови Kotlin, завдяки чому мова стала універсальним інструментом для розроблення застосунків під Android, iOS, створення десктопних та веб-застосунків. Ключова ідея полягає в тому, що один і той самий код, написаний мовою Kotlin, може працювати на різних платформах з мінімальними змінами. Важливою технологією у використанні мови Kotlin є Kotlin Multiplatform, яка уможливлює спільне використання коду (логіки, моделей, алгоритмів) різними платформами. Ця технологія використовується у фреймворку Kotlin Multiplatform Mobile (KMM), призначеному для створення застосунків, здатних працювати одночасно на Android та iOS. У рамках цього підходу розробник пише спільний код на Kotlin і додає лише специфічні частини для кожної платформи.

• Мова Python на сьогодні є однією з найпопулярніших. Вона використовується багато і в найрізноманітніших сферах. Як зазначалося вище, Python є інтерпретованою мовою програмування, яка має реалізації для багатьох платформ. Значна кількість бібліотек та фреймворків відкривають широкі перспективи щодо створення різноманітних додатків, які можуть працювати на різних операційних системах.

• Мова C# разом з технологією .Net Core дозволяє створювати кросплатформні застосунки (наприклад, за допомогою таких інструментів, як .Net MAUI). Завдяки впровадженню .Net Core, C# стала кросплатформною мовою й може використовуватись для розроблення додатків для різних операційних систем. А саме, існує можливість розробляти кросплатформні додатки для Windows, macOS та Linux. Підтримуються додатки різних типів: консольні, веб-застосунки, серверні та десктопні програми.

Деталі

Платформа .Net Core починаючи з версії .Net 5 (з'явилась у 2020 році) має офіційну назву .Net (без використання слова Core у назві). Актуальною на момент написання книги є версія .Net 11. Вихід версії .Net 12 заплановано на 2027 рік.

Окремо можна відмітити фреймворк .Net MAUI, який використовується для створення мобільних застосунків для платформ Android та iOS з використанням C#. Він уможливлює використання спільного коду для обох платформ.

Ще один сучасний високопродуктивний фреймворк ASP.NET Core є кросплатформним інструментом для створення веб-застосунків. Він працює на Windows, Linux та macOS, підтримує Docker, що дозволяє запускати застосунки в контейнерах на різних системах.

• Кросплатформність мови JavaScript базується на тому, що це інтерпретована мова для створення сценаріїв, які виконується у середовищі виконання, доступному на різних платформах. Основна платформа для JavaScript – це веб-браузери (такі як Chrome, Firefox, Safari, Edge). Кожен браузер має вбудовану підсистему, яка виконує JavaScript-код незалежно від операційної системи. Тому один і той самий код працює на Windows, Linux, macOS, Android, iOS (якщо браузер підтримує стандарти JavaScript). Усе це робить JavaScript універсальною мовою для веб-розроблення та не лише. Крім того, завдяки таким технологіям, як Node.js та Electron, існує можливість створювати серверні або десктопні застосунки для різних платформ.

• Кросплатформність мови програмування TypeScript у першу чергу пов'язана з тим, що ця мова є надмножиною мови JavaScript. Вона розширює мову JavaScript типізацією, однак програма врешті-решт трансформується у традиційний JavaScript-код. Як наслідок, мова TypeScript реалізує таку саму кросплатформність, як і мова JavaScript. Однак при використанні TypeScript існують певні переваги для розробників.

• Мова програмування Rust є компільованою, тому код, написаний мовою Rust, у процесі компіляції перетворюється на машинний код для конкретної платформи. Однак при цьому підтримується велика кількість архітектур та цільових платформ, серед яких, крім іншого, є Windows, Linux, macOS, Android та iOS. Тому область застосування програм, написаних мовою Rust, не обмежується конкретною операційною системою. Такі програми є універсальними на рівні компіляції.

До уваги

Відверто кажучи, мова Rust – не єдина мова програмування з означеними "кросплатформними" властивостями. Існують також інші мови, які в тому чи іншому аспекті

можна віднести до мов з кросплатформною підтримкою. Тому тут Rust слід розглядати радше як приклад такої мови.

Цей список можна продовжувати, і у такому разі він міг би розширитись доволі сильно. Причина в тому, що навіть для дуже специфічної мови, створеної для розроблення застосунків під певну платформу, завжди можна знайти механізми, які так або інакше стосуються парадигми кросплатформності.

Кросплатформність і Java

У книзі ми познайомимось з мовою програмування Java і тим, як в Java реалізується кросплатформність. Ми обговоримо, що в цій мові особливо цікавого і за що її люблять (чи навпаки, не люблять) розробники. Але у будь якому разі Java – мова особлива. Крім того, вона одна з найпопулярніших.

Деталі

Популярність визначається рейтингами. А рейтинги можна складати по-різному. Не існує якогось критерія щодо того, який рейтинг правильний, а який неправильний. Можна, наприклад, проводити опитування серед програмістів з приводу того, якою мовою програмування вони користуються на роботі. Можна провести опитування з приводу того, яку мову програмування вважають найкращою професіональні програмісти. Можна аналізувати оголошення про прийом на роботу. При цьому будь-яка статистика не завжди є об'єктивною і може бути викривлена в силу об'єктивних чи суб'єктивних обставин. Тому всі рейтинги є досить умовними. Але якщо у різних рейтингах певна мова програмування з'являється хоча б і на різних позиціях, але знаходиться у верхній частині рейтингу, то таку мову сміливо можна називати популярною. Такою є мова Java.

Популярність мови означає, що вона використовується для створення нових проєктів. З іншого боку, мова має довгу історію й застосовується протягом багатьох років. Тому створено багато бібліотек для цієї мови. Також існує багато програмних продуктів, створених на Java. Їх потрібно підтримувати. У цьому сенсі все, що вже зроблено з використанням мови Java, є серйозним стимулом щоб не відмовлятись від цієї мови.

Довідка

Можна навести такий історичний приклад. Довгий час наукові обчислення (в області фізики, хімії, тощо) науковцям доводилось виконувати самостійно (без залучення фахових програмістів та спеціальних програмних пакетів). Мовиться в першу чергу про математичні обчислення. Для цього використовували мову програмування FORTRAN. Для цієї мови було створено багато спеціалізованих бібліотек, надійних і добре протестованих та перевірених у справі. Наявність таких бібліотек і спеціальних утиліт для обчислень стала серйозним аргументом для подальшого використання FORTRAN у обчисленнях навіть після того, як почали з'являтись нові перспективніші програмні засоби. Зрозуміло, що з часом цю тенденцію було змінено, однак факт залишається фактом – наявність зручних готових рішень продовжує час використання технології, в рамках якої ці рішення було створено.

Отже, мова Java є цікавою й перспективною, так би мовити, сама по собі. Нас же вона цікавить в силу її фундаментальної особливості – кросплатформності. Питання, чому саме ця мова стала кросплатформною, варто розглядати у контексті її створення. Коли мова Java створювалася, ідея щодо забезпечення роботи програм на різних платформах поступово виходила на передній план. Але

задача поставлена перед інженерами-розробниками, була дещо іншою. Їм потрібно було створити технологію для програмування побутових пристроїв. А це автоматично означає, що програмні коди повинні працювати на різних платформах.

Довідка

Розробленням мови Java займалася компанія Sun Microsystems, яку згодом поглинула корпорація Oracle. Сьогодні саме Oracle асоціюється з Java. І хоча корпорація реалізує значно більше проєктів, ніж підтримка Java, саме ця діяльність є одним з ключових напрямів.

Спочатку мова Java мала іншу назву – Oak (що означає "дуб"). Згодом, отримавши сучасну назву Java, вона почала активно поширюватися, а її універсальність і зручність стали головними факторами популярності.

Точне походження назви Java залишається дискусійним. За однією з версій розробники просто обрали назву сорту кави, яку вони часто купували під час роботи. Чашка кави стала впізнаваним символом та своєрідною "візитівкою" Java. Цей логотип знають навіть ті, хто не має стосунку до програмування.

На момент офіційного релізу мови в 1995 році не існувало іншої схожої за потужністю і ефективністю мови, яка б мала кросплатформну підтримку. Для порівняння, мова C# виникла пізніше, у 2000 році. На момент виникнення ця мова призначалась винятково для роботи з операційною системою Windows. На сьогодні мова C# – це кросплатформна мова, красива і доволі ефективна. Але як би там не було, ми матимемо справу з мовою Java.

Хоча первісно Java планувалася як платформа для програмування побутових електронних пристроїв, згодом сфера застосування значно розширилася, і кросплатформність стала головною перевагою мови, яка й визначила її подальший успіх. Оскільки побутові пристрої, для яких створювалась технологія, мали різну апаратну основу, виникла проблема уніфікації програмного забезпечення. Потрібен був підхід, який дозволяв би писати єдиний програмний код, здатний працювати на різних типах пристроїв. Саме з цією метою було запропоновано концепцію проміжного байт-коду. А саме, програми, написані мовою Java, не компілюються безпосередньо в машинний код для конкретної платформи. Натомість вихідним результатом компіляції є так званий байт-код (універсальний проміжний код). Цей байт-код виконується спеціальною програмою, яка називається віртуальною машиною (Java Virtual Machine, або JVM), яка встановлювалася на кожному конкретному пристрої. Отже, універсальність забезпечується тим, що байт-код залишається однаковим, а JVM адаптує його виконання до особливостей конкретної платформи. Це й стало ключовою технічною ідеєю, яка відрізняла Java від інших мов програмування.

До уваги

Поява мови Java стала дійсно чимось новим у програмних технологіях, оскільки з'явилась можливість створювати єдиний програмний код, який працював би і під Windows, і під Linux, і навіть створювати кросплатформні застосунки з графічним інтерфейсом. Сьогодні цим уже нікого не здивуєш, але на момент появи мови Java це була нова й ефективна (та й ефектна) технологія.

Деталі

Навряд чи можна стверджувати, що кросплатформність на основі мови Java є якоюсь унікальною. Разом з тим, кросплатформність мови Java є дещо іншого рівня в порівнянні з більшістю інших мов програмування. У принципі, майже кожна мова