УДК 004.42:004.4’4
DOI

Дмитрієва О.А., Ярош І.В., Черняк Т.О., Воротиленко П.Ю.

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ З ВБУДОВАНИМИ МОЖЛИВОСТЯМИ КОМПІЛЯЦІЇ І ТЕСТУВАННЯ

Статтю присвячено питанням проєктування та розробки мобільної Android-системи навчального призначення, з вбудованими сервісами для здійснення компіляції та тестування лістингів програм, що написані користувачем внаслідок вирішення поставлених перед ним завдань з Java-програмування. В роботі проаналізовано проблему, на розв’язання якої спрямовано розробку мобільного застосунку. Проведено аналіз предметної області, в межах чого досліджено наявні засоби для забезпечення вивчення основоположних принципів стосовно побудови кодів мовою Java та проаналізовано існуючі програмні системи-компілятори. Досліджено та оцінено сучасні методи і підходи до компіляції і тестування програмного засобу, складено перелік вимог до функціональності розроблюваної системи. Проєктування Android-додатку виконано із застосуванням розповсюдженої UML-методології, описано варіанти використання програмного застосунку, стани, які він може приймати під час роботи, та послідовність дій, які необхідно виконувати мобільній програмній системі для коректного функціонування та взаємодії з користувачами. Додатково сформульовано вимоги до людино-машинного інтерфейсу розроблюваної системи. Результати розробки програмного продукту представлено реалізованими програмними класами. Розглянуто особливості функціонування побудованої системи при різних комунікаційних користувацьких сценаріях, висвітлено питання взаємодії між складовими програмних екранів, потоками виконуваних обчислень, передачі даних. Наведено опис окремих компонентів, модулів та підсистем, з яких складається розроблений програмний засіб. Також визначено та описано послідовність дій користувача при виконанні основних операцій, передбачених і реалізованих у додатку для мобільних пристроїв, що працюють на базі платформи Android. Програмним результатом роботи є застосунок, що складається з модулів для підтримки навчального процесу та комплексу програм для забезпечення працездатності засобів автоматизації процесів підготовки і виконання користувальницьких програм.

Ключові слова: мобільний застосунок, компіляція, тестування, операційна система Android, мова Java, UML-методологія, програмні класи

Вступ. Однією з найперспективніших галузей на теперішній час є ІТ-сфера, яка стрімко розвивається, нарощує оберти та відкриває широке коло можливостей. Все більше представників суспільства прагне здобути освіту в галузі інформаційних технологій і освоїти професію програміста, і, як наслідок, спостерігається високий попит на навчання програмуванню. Зазначена тенденція спричинена й обумовлена тим, що вміння писати комп’ютерні програми є цікавим і високооплачуваним заняттям, яке передбачає престижну, виключно інтелектуальну діяльність. Кожна особа, яка залучається до вивчення концепцій і принципів програмування, стикається з потребою встановлення та налаштовування середовища створення програмного забезпечення (ПЗ), а потім трансляції (компіляції), запуску та перевірки працездатності сформованого нею коду програми [1]. Для програміста-новачка така послідовність дій здається доволі складною, спричиняє подальше небажання продовжувати навчання професії програміста. В більшій мірі це проявляється в ситуації, коли людина повністю не впевнена в тому, чи бажає бути програмістом, і приймає рішення спершу ознайомитися з початковими азами професії. Саме тому набуває актуальності процедура розробки програмного забезпечення, яка супроводжувалася б простотою встановлення та експлуатації, високою функціональ-ністю, характерною для середовища розробки (можливістю написання, відладки та виконання коду), а також підтримувалася інформацію навчального характеру для надання теоретичних відомостей користувачам. Сервіси вже пропонуються на ІТ-ринку, проте їм притаманні недоліки, серед яких можна виокремити наступне: тільки англомовний супровід навчання, завищена вартість тощо. Відповідно, доцільним є створення застосунку, який

за допомогою простих і прозорих користувацьких дій легко й швидко встановлювався, надавав для опрацьовування навчальні матеріали з програмування та міг би використовуватися в якості середовища розробки ПЗ.

Об'єкт дослідження в роботі – процеси процедур компіляції і тестування для розробки програмних Android-застосунків.

Предмет дослідження в роботі – мобільні програмні системи навчального спрямування та системи-компілятори.

Мета роботи полягає у проєктуванні та програмній реалізації мобільного застосунку у вигляді системи, що одночасно виконує освітні завдання з Java-програмування та вміщує засоби для здійснення автоматичної перевірки написаного користувачем програмного коду.

Завдання роботи полягають у виконанні огляду та дослідженні сучасних підходів до проєктування мобільних додатків систем навчального призначення, предметної області програмування мовою Java, аналізі технологій проєктування ПЗ, спрямованого на процеси навчання, розробці архітектури мобільного застосунку з передбаченим функціоналом для забезпечення можливості автоматичної перевірки програмного коду, проєктуванні користувальницького інтерфейсу програмного продукту, визначенні взаємодії його складових компонентів і елементів; програмній реалізації і тестуванні засобу.

Методи досліджень в роботі представлені способами та прийомами, які застосовуються для забезпечення й реалізації процесів трансляції програм, написаних на мовах високого рівня, їх тестування для виконання автоматичного способу перевірки програмного коду, що базуються на теоретичних і практичних положеннях теорії алгоритмів, кросплатформного програмування, моделювання, проєктування та архітектури систем, теорії синтаксичного аналізу та компіляції, тестування ПЗ.

Оцінка сучасного стану порушеної проблеми. Сьогодні спектр професій, які можна об'єднати під єдиною назвою «програміст», дуже актуальний для суспільства. Дану тенденцію напряму пов'язано з вивченням інформаційних наук і технологій, зокрема, мов програмування. Людина, яка вміє користуватися комп'ютером на високому рівні, може освоїти основи, необхідні для початкової роботи програмістом, протягом одного року, що є досить коротким періодом порівняно з освоєнням багатьох інших професій [2-3]. У той же час професія програміста продовжує залишатися затребуваною в усьому світі протягом багатьох років і вимагає від фахівця постійного перенавчання, самовдосконалення, опанування новими технологіями, платформами, мовами програмування.

Якщо відійти від сфери інформаційних технологій, особливо від проблеми отримання вигоди кожною людиною від роботи програмістом, то можна сказати, що розвиток інформаційних технологій відіграє дуже важливу роль у розвитку сучасного суспільства та розвитку економіки кожної держави. Таким чином, інформаційні технології відіграють важливу роль в освіті, наукових дослідженнях, автоматизації виробничих процесів і поліпшенні комунікацій [4]. Наприклад, автоматизація комунікації між різними підрозділами великої компанії (відправка спеціально згенерованих звітів, автоматизація обробки документів, даних про співробітників, корпоративних клієнтів тощо) може значно заощадити як час, так і фінансові та людські ресурси. Розвиток окремих підприємств в більшості випадків робить позитивний вплив на розвиток економіки держави в цілому. Так, люди, які хочуть жити і забезпечувати свої сім'ї матеріальною стабільністю та безпекою, мають амбіції до кар'єрного зростання, прагнуть до економічної незалежності і хочуть здійснювати цікаві й корисні для суспільства проєкти – отримують знання в області програмування. Так само розвиток інформаційних технологій позитивно впливає на розвиток

суспільства та економіку держави, тому держава і людське суспільство в цілому зацікавлені в підготовці нових фахівців в області програмування.

В результаті багато осіб зацікавлено в отриманні професії програміста, але існує досить високий поріг для вступу на цю спеціальність. Люди, які хочуть навчитися програмуванню, зазвичай, не знають, з чого почати. Також існує досить мало джерел, які дозволяють ознайомитися з професією, не перевантажуючи людину великою кількістю усталеної термінології, а також не блокуючи його зацікавленість в оволодінні професією. Сьогодні ринок праці потребує нових ІТ-фахівців і пропонує високу оплату за роботу в цьому напрямку, тому важливо заохочувати людей вивчати програмування.

2. Дослідження існуючих підходів до вирішення порушеної проблеми. На поточний момент вже існує досить велика кількість різних навчальних посібників (як у цифровому, так і в текстовому вигляді), спрямованих на вивчення технологій розробки ПЗ із застосуванням мови Java [5]. Можна виділити окремі цифрові посібники та продукти, націлені на Java-розробників, – «JavaRush» [6], «Codecademy» [7] і ін., але потрібно зауважити, що кожний з них має певні особливі риси і недоліки. У онлайн-підручниках з Java-програмування часто розміщені тільки відомості теоретичного характеру та відсутні практичні завдання. Вебресурси та портали, що вміщують теоретичний матеріал, не завжди надають його в зручній формі, і, як і значна частина книг, не пропонують практичних завдань. Деякі програмні продукти вміщують занадто деталізовані й дуже об'ємні матеріали, в яких лаконічно розібрані всі тонкощі і особливості мови програмування Java, проте формат їх подання найчастіше не дозволяє або ускладнює процедури пошуку потрібних даних серед їх скупчення та потребує тривалий час для їх опрацьовування. Також актуальна проблема, яка полягає в тому, що безліч навчальних ресурсів є виключно англомовними або є версіями неточного, спотвореного перекладу оригінальних джерел. Ті засоби, які, окрім надання теоретичного матеріалу, забезпечують також можливість перевірки програмного коду, зазвичай, є платними та відносно дорогими.

Важливо та бажано, щоб підручник і середовище розробки ПЗ поєднувалися в єдине ціле утворення, щоб користувачам-учням не доводилося перемикатися між засобами отримання інформації та застосування її на практиці. Зважаючи на цю інформацію, постає рішення про раціональну інтеграцію в єдиний програмний продукт середовища розробки та програми, що надаватиме потенційним майбутнім Java-програмістам потрібні теоретичні відомості. Розроблюваний програмний засіб повинен: поєднувати у собі компактне середовище розробки з джерелом теоретичних даних з Java, надавати користувачеві на додаток до теорії практичні завдання для реалізації з метою закріплення отриманих раніше теоретичних даних, надавати користувачеві можливість автоматично перевіряти написаний ним програмний код.

3. Проєктування програмного за стосунку. В якості платформи для розробки власного програмного засобу забезпечення та підтримки навчання основам Java-програмування було обрано операційну систему Android [8]. Серед специфічних вимог до Android-застосунків виділено декілька аспектів:

–зважаючи на невелику кількість оперативної пам'яті Android-пристроїв прийнято оптимізувати програмні застосунки (навіть при наявній шкоді продуктивності засобу);

–користувацький інтерфейс продукту повинен бути адаптивним: застосунок повинен мати схоже відображення на пристроях із різними параметрами екрану (потрібно враховувати розподільну здатність екрану, його діагональ та їхнє співвідношення);

–з метою потенційного охоплення максимальної користувальницької аудиторії рекомендовано створювати додатки, які сумісні з мінімально низькою версією Android

API (версію Android 4.3, підтримуваною на 98,4% пристроїв, використовуваних користувачами) [9].

Підсумки проєктування мобільного застосунку представлено комплексом UML-діаграм, що втілюють положення специфікації як функціональних, так і нефункціональних вимог.

Поведінка проєктованої системи при взаємодії з нею користувачів і специфіка її внутрішньої роботи (зокрема, поведінка додатку при виділенні окремих потоків для того, щоб компіляція або робота програмного коду, написаного користувачем, не могли призвести до виникнення так званих «глухих кутів», при яких система перестає реагувати на дії користувача) відображена діаграмою системних прецедентів (рис.1).

Рисунок 1 – Діаграма варіантів використання мобільного застосунку

На зображенні присутній актор, який представляє користувача, що вивчає навчальний матеріал курсу, присвячений Java-програмуванню. Додаток надає користувачеві можливість переглянути перелік доступних уроків. Користувач може одразу перейти до обраного уроку, в якому будуть відображені теоретичні навчальні матеріали з описом базових способів роботи з Java. Перелік доступних уроків формується з найменувань заголовків уроків, при натисканні на кожний з яких користувач переходить до вмісту обраного уроку.

При перегляді вмісту теоретичної частини користувач може прочитати відомості щодо Java-програмування, що подані з використанням простого тексту, містять запропоновані приклади лістингів, приклади роботи консольних мініпрограм при отриманні певних даних ззовні та виведенні відповідних даних на екран. Після ознайомлення з теоретичною частиною користувач або переходить до наступного екрану з теорією, що освітлює наступну тему, або до екрану з завданням для закріплення знань і практичним оволодінням розглянутого матеріалу. При цьому на екрані користувач бачить просте завдання з кодування, в якому використовуються засоби, розглянуті на попередніх уроках. Як правило, після виконання завдання відображуються деякі приклади даних програмного введення-виведення, які користувачеві необхідно створити в рамках завдання. Користувач може перейти на екран, щоб ввести текст коду. У цьому вікні користувач може написати Java-код і

виконати автоматичну перевірку або ручні тести. Щоб код вважався правильно написаним, призначена для користувача програма повинна відправляти і отримувати дані у форматі, визначеному завданням (приклад введення-виведення даних). Користувач може ввести і виконати будь-який простий код Java, який працює в консолі. Таким чином, користувач може протестувати різні властивості мови Java, щоб поглибити знання, отримані на уроці. Якщо обрати автоматичну перевірку коду, користувач отримає відповідні повідомлення про робочий стан (код працює правильно/неправильно, код не може бути скомпільований/виконаний, може виникнути нескінчений цикл і т.інш.).

На екрані ручної перевірки система емулює консоль введення-виведення. Користувач може особисто перевірити код, який він написав власноруч, дослідити реакцію на введення певних вихідних даних з консолі, тобто забезпечується процес відображення інформації на екрані. Якщо користувач випадково створить нескінченний цикл або виникнуть інші обставини, що перешкоджають подальшому тестуванню, йому надається можливість терміново зупинити виконання коду. Як тільки виконання зупиниться, є можливість знову запустити код і протестувати його на іншому наборі даних, не виходячи з екрану ручного перегляду. Користувач може перезапускати код стільки разів, скільки необхідно.

Для фіксації станів, які притаманні системі при здійсненні тих чи інших дій користувача або зміні деяких параметрів, що впливають на роботу додатку, застосовано UML-діаграму станів. На рис. 2 представлено побудовану діаграму станів для розроблюваної мобільної системи. При завантаженні система може відкрити не тільки теоретичну частину, але й умову завдання, над якою працював користувач, коли система працювала. Цей пошук організований завдяки тому, що при завершенні роботи користувача з кожним екраном (теорії або завдання) додаток протоколює та зберігає інформацію про завершення уроку або проходження завдання у спеціально відведеному для цього файлі («файлі уподобань»). Інформація про проходження певного практичного завдання зберігається тільки в разі успішного виконання тестів програмного коду, написаного користувачем.

З будь-якого екрану системи можна перейти на екран вибору уроку. При натисканні на заголовок уроку на цьому екрані система переводить користувача на екран, де користувач може ознайомитися з теорію щодо обраного ним уроку. При натисканні кнопки «Назад» користувач повертається на попередній екран, з якого здійснений перехід. Якщо на екрані, який відображає теоретичні матеріали, користувач натискає кнопку «Далі», то система переходить до наступного екрану (або до іншого екрану з теоретичними даними, або до екрану з завданням). В додатку передбачений розподіл теорії на декілька екранів, що має логічний сенс. Натискання кнопки «Далі» на екрані з завданням завжди призводить до однієї й тієї ж самої дії: система відкриває екран, на якому користувач може ввести програмний код. На цей екран передається інформація про те, яким саме завданням його було викликано. Використовується номер уроку та номер завдання в межах цього уроку (один урок може містити декілька практичних завдань). Коли код тестується автоматично, система використовує дані щодо номеру уроку та завдання для того, щоб сумістити з роботою програми відповідні набори даних.

На екрані введення програмного коду користувач побачить шаблон, який використовує стандартний Java-метод «main» (без тіла класу для ясності і простоти), в якому він може написати текст програми. Користувач не може змінити визначення методу «main», він має можливість редагувати тільки тіло методу. При натисканні на кнопку автоматичної перевірки на екрані введення коду програми, всі елементи інтерфейсу блокуються, і в полі введення коду відображається так званий «лічильник»

(віджет, який повідомляє вас про виконання обчислень або завантаження даних). Користувач повинен дочекатися закінчення цієї процедури.

Якщо процедура тестування не буде завершена протягом п’яти секунд, з’явиться спливаюче повідомлення, яке вказує, що виконання коду, ймовірно, призведе до утворення нескінченного циклу. Тестування і робота з призначеним для користувача кодом будуть завершені в терміновому порядку. Якщо тестування завершується вчасно, система надає користувачеві інформацію про те, чи був тест успішним чи ні. У разі успіху користувач перейде на інший екран і отримає повідомлення про те, що введений ним код працює правильно і тест пройдено успішно. В іншому випадку система відобразить спливаюче повідомлення, яке вказує, що код не пройшов перевірку. Якщо код не вдається скомпілювати, система сповіщує користувача відповідним спливаючим повідомленням.

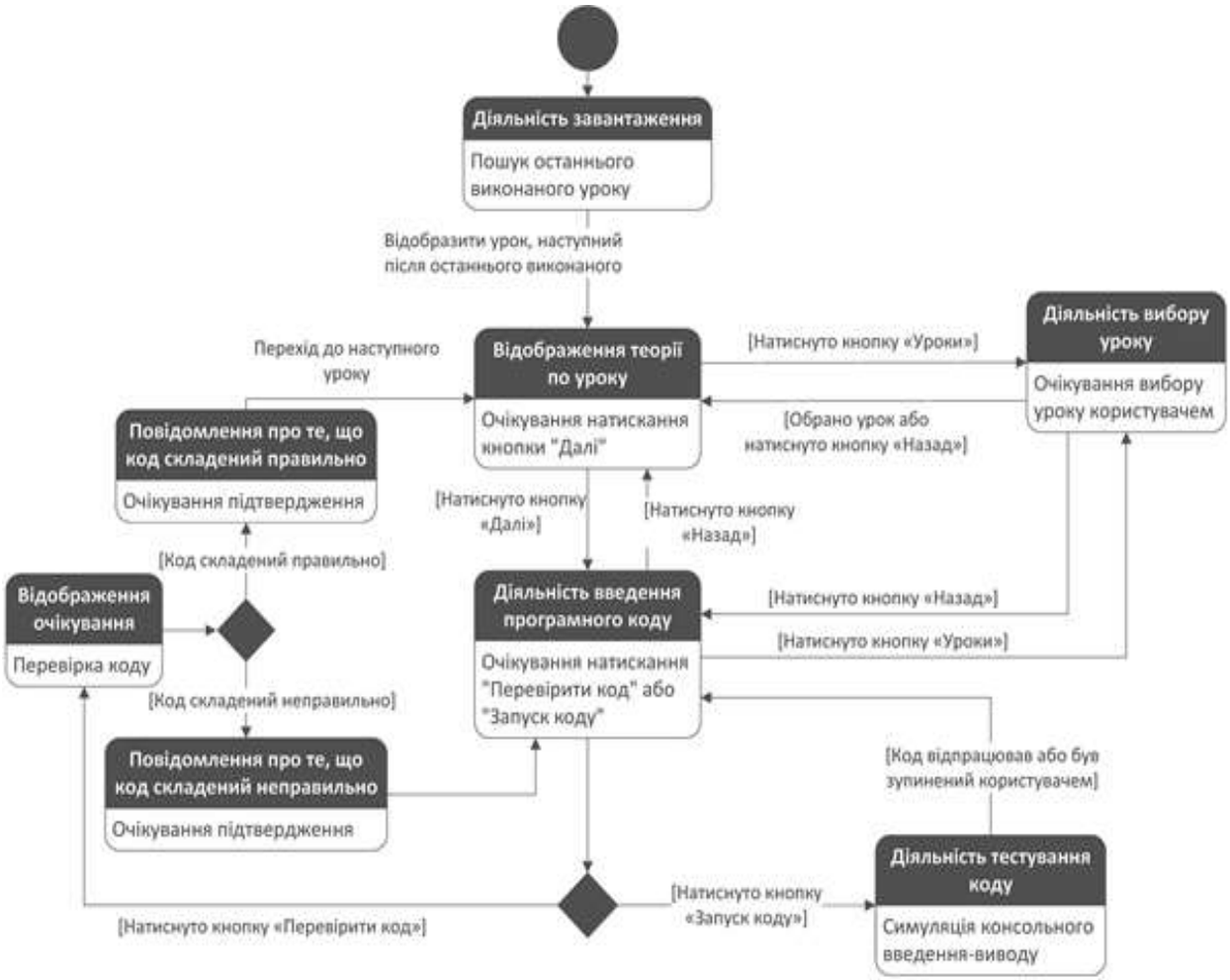

Рисунок 2 – Діаграма станів для мобільного застосунку

В системі передбачено, але не візуалізовано на діаграмі задля уникнення надмірності даних, екран для сповіщення користувачеві про успішне проходження тестів. При виклику цього екрану йому передаються дані про номер уроку та номер завдання в межах цього уроку. Це ті самі дані, які було передано екрану введення користувацького програмного коду. При натисканні на екрані успішного проходження тестів кнопки «Далі» система, спираючись на ці дані, визначає, який екран викликати наступним. Потрібно відмітити, що навчання в системі є лінійним, тобто за кожним

завданням слідує або наступне завдання, або новий урок.

Під час проєктування застосунку було виконано побудову діаграм послідовності для візуалізації взаємодій між певними об'єктами системи у часі, для графічної фіксації аспектів роботи пов'язаних між собою потоків багатопоточного додатку. Виконано побудову діаграм послідовностей для процесу автоматичного тестування у випадку, коли виконання коду призводить/не призводить до утворення нескінченного циклу, процесу ручного тестування коду у випадку, коли користувач викликає/не викликає екстрене завершення виконання коду.

Ґрунтуючись на результатах проєктування можна зробити висновок, що додаток повинен працювати на пристроях під управлінням Android-системи. Постійне підключення до Інтернету не потрібно. Серед апаратних компонентів системи можна виділити інструменти, з яких система запускається безпосередньо. Система зберігає невелику кількість даних про останній пройдений урок або завдання, тому створення бази даних не рекомендовано, це збільшить навантаження і вимоги до ємності накопичувача пристрою. В якості засобу для реалізації спроєктованого додатку обрано середовище Android Studio.

4. Обґрунтування архітектурних рішень та розробка програмного за стосунку. До моменту програмної реалізації додатку сплановано та відтворено структуру системи зсередини. Виконано визначення набору складових класів системи, зв'язків між ними, до реалізації. Частину класів успадковано від класів Android-системи або класів мови Java [10].

Розробку Android-застосунку неможливо розглядати окремо від роботи діяльностей застосунку, їх перевизначених методів та взаємодії діяльностей між собою. Під діяльністю в Android-додатку передбачається один окремий «екран», це є доцільним згідно логіки та специфіки Android-платформи. У Java кожна діяльність являє собою окремий клас, що є наслідником системного класу «androidx.appcompat.app.AppCompatActivity». Кожна діяльність повинна обов'язково перевизначити метод «onCreate()», який викликається при створенні діяльності. Зазвичай, діяльності перевизначають також ряд інших методів.

Для можливості компілювання введеного користувачем коду використана відкрита бібліотека BeanShell, що дозволяє запустити вихідний Java-код. Процес супроводжується створенням «BeanShell-інтерпретатора» – об'єкта, що приймає об'єкт String, який містить програмний код, запускає цей код і виконує його. Щоб отримати результати роботи інтерпретованого коду використовуються об'єкти, які передаються в якості параметрів.

Використання бібліотеки BeanShell має значний вплив на архітектуру розроблюваної системи. Бібліотека не є потокобезпечною, тому для того, щоб весь застосунок не припиняв роботу у разі, коли користувач вводить, наприклад, код, що містить нескінченний цикл, виникла потрібність застосовувати специфічні архітектурні рішення. При автоматичній перевірці набраного коду першочергово він редагується, поміщується в середину потокобезпечного контексту і тільки після цього передається інтерпретатору на виконання. Для забезпечення симуляції консольного введення-виведення при запуску користувацького коду для тестування в ручному режимі виникла навіть необхідність створення додаткових класів, в тому числі, потоків, які забезпечують обмін даними між об'єктами, що були передані в якості параметрів до інтерпретатора, та полем EditText, яке, безпосередньо, виступає емулятором консолі у GUI. Додатково до системи також підключено відкриту бібліотеку вихідного коду JustifiedTextView, яка дозволяє розміщувати на діяльності віджет, що містить вирівняний текст (це неможливо виконати у мінімальній цільовій платформі додатку стандартними підходами). Привабливий зовнішній вигляд тексту є важливою вимогою

до зовнішнього вигляду застосунку, тому що навчальний додаток передбачає значну кількість теоретичного матеріалу, який, в даному випадку, представлено в текстовому вигляді.

Бібліотека JustifiedTextView є недосконалою, віджет працює неправильно, якщо його додавати до XML-розмітки, як це робиться при створенні звичайної діяльності. Через це JustifiedTextView розміщується на екрані діяльності у програмному Java-коді, в методі створення діяльності («onCreate()»). Саме тому візуальні елементи всіх діяльностей, які містять у собі JustifiedTextView (діяльності теоретичної інформації та умов завдань) створюються завдяки Java-коду. Для їх позиціонування та масштабування було зручніше вказувати координати розміщення у відносних величинах (відповідно до стандартів розробки інтерфейсу Android-застосунків), система використовує «LayoutParamsData» – клас, написаний і застосовуваний для оптимізації створення діяльностей з групи TheoryActivity та TaskActivity, а також для спрощення коду створення цих діяльностей, де зберігається інформація щодо параметрів розташування елементів інтерфейсу, їх відступів і ін.

Застосунок містить велику кількість реалізованих класів, які щільно пов’язані між собою. Кожен клас, в назві якого є слово «Activity», є наслідником системного класу «androidx.appcompat.app.AppCompatActivity». Також реалізовано анонімні класи, що виконують роль методів, так як виконують функцію натискання для кнопки.

Більшість класів застосунку являє собою діяльності (робочі екрани, що містять користувацький інтерфейс).

Опис реалізованих класів/діяльностей застосунку: «MainActivity» –діяльність, яка першою запускається при завантаженні додатку, вміщує метод «onCreate()» для виклику «calculateAllLayoutParams()» і отримання доступу до файлу, в якому зберігається інформація щодо останнього пройденого уроку або завдання. «LessonsActivity» – клас, в якому розміщено список уроків, доступних для вивчення, «TheoryActivity» – діяльність, яка містить теоретичні відомості щодо основ програмування на Java, приклади лістингів і вхідних/вихідних даних програм, «TaskActivity» – діяльність, яка містить умови практичних завдань, «Constants» – клас, який у статичних полях зберігає константи, які допомагають різним частинам застосунку взаємодіяти одна з одною, «LayoutParamsData» –клас, що використовується для оптимізації створення діяльностей з групи TheoryActivity та TaskActivity, а також для спрощення коду створення цих діяльносте, зберігає інформацію щодо параметрів розташування елементів інтерфейсу. «LessonsAdapter» – клас, що вміщує можливості позначення різних елементів кольором; «ViewHolder» – клас, що вміщує можливості для оптимізації відображення елементів, «CodeWritingActivity» – діяльність, яка надає інтерфейс для введення коду, «CodeTextWatcher» – клас, що реалізує інтерфейс «android.text.TextWatcher» і реалізований для змінення слухачем курсу тексту у полі для введення коду, його задача – зробити введення коду у додатку схожим на введення коду у професійному IDE. «TestingTask» – клас, що створено з ціллю правильного відображення елементів інтерфейсу при запуску довгого автоматичного тестування, «TestingTask» – клас, що створений для сповіщення про перебіг тестування, кінцеві результати компіляції або перевірки, про виняткові ситуації, «Tester» – клас, що є основним класом для виконання автоматизованого тестування, «StreamsTransmitter» – клас, що забезпечує пересилку даних, які код користувача виводить на екран, до потоку GUI, «UserCodeTester» – клас, де приписана основна логіка роботи користувацького коду в режимі ручної перевірки та ін.

Реалізовано також декілька специфічних класів-діяльностей: «NoLessonActivity» – клас, що повідомляє користувачеві про те, що урок, до якого він хоче перейти, ще не готовий, але буде найближчим часом додано, «ThankYouActivity» – діяльність,

ініційована після проходження користувачем всіх доступних на даний момент уроків, вміщує подяку за проходження уроків, інформацію, що нові уроки з'являться найближчим часом.

Отже, основну логіку системи, що стосується компіляції і перевірки введеного користувацького коду, сконцентровано в базових класах «TestingTask», «Tester», «StreamsTransmitter», «UserCodeTestingActivity» та «UserCodeTester». При передачі даних між екранами та потоками часто фігурує клас «Constants», що містить основну частину констант програми. Для правильного розміщення та масштабування елементів інтерфейсу при створенні деяких діяльностей використовується клас «LayoutParamsData». Тексти уроків та завдань розміщено на діяльностях, що належать до груп «TheoryActivity» та «TaskActivity» відповідно. При перевірці вмісту введеного лістингу система створює об'єкти потоків, які розширюють класи «java.lang.Thread» та «android.os.AsyncTask».

Зовнішній вигляд розробленого застосунку реалізовано в стилі мінімалізму, щоб забезпечити максимальну зручність під час взаємодії із додатком. Екрани застосунку не містять надлишкових елементів, виглядають «лаконічно» та не відволікають увагу користувача, даючи змогу сконцентруватися виключно на навчальному процесі. Усі доступні для взаємодії елементи інтерфейсу побудовано так, щоб користувачеві було інтуїтивно зрозуміло їх призначення та передбачену функціональність. Кольорову палітру для оформлення дизайну форм/екранів застосунку підібрано так, щоб вона була приємна та спонукала до навчання.

Навчальні матеріали та повідомлення, розміщені в курсі, сформульовані в доброзичливій манері, без застосування важких для розуміння формулювань. Вміст навчальних компонент для уроків поєднує в собі опис технічних особливостей програмної мови та дружню бесіду з користувачем. На екрані для введення користувацького лістингу присутнє текстове поле, в якому можна друкувати програмний код, що автоматично форматується, допомагаючи користувачеві швидше набирати програму. При автоматичному тестуванні введеного коду екран відповідно змінює інтерфейс, сигналізуючи користувачеві про те, що йде процес обчислення. Екран, в якому можна тестувати програму в ручному режимі, представлено у вигляді текстового поля, що емулює роботу консолі. Після завершення виконання користувацького коду видається спливаюче повідомлення про отримані результати компіляції і тестування, відповідні ситуації можуть передбачати і повідомлення про наявні помилки. Програмно реалізована коректна поведінка з відображення додатку при зміні орієнтації екрану, забезпечена адаптація інтерфейсу застосунку під екрани з різною щільністю, розподільною властивістю та діагоналлю (за рахунок використання відносних способів завдання величин розмірів складових елементів, а також завдяки правильному розміщенню елементів у контейнерах).

Висновки. В рамках роботи проведено та описано ряд заходів із розробки мобільного застосунку для вивчення курсу «Основи Java-програмування» з передбаченими сервісами для компіляції та тестування користувацьких кодів програм. Виконано аналіз предметної області, складено перелік вимог до функціональності засобу, здійснено огляд проблеми, яку вирішує програмний додаток, а також існуючих способів її подолання. Здійснено огляд вимог до інтерфейсу програмного застосунку та до деяких особливостей його поведінки. За допомогою UML-діаграм візуалізовано функціональні та поведінкові аспекти програмної системи. Наведено опис і призначення реалізованих програмних класів. Програмний мобільний Android-застосунок розроблено в середовищі Android Studio, спрямовано на забезпечення та підтримки навчального процесу з курсу «Основи Java-програмування», передбачено вбудовані можливості щодо здійснення компіляції і тестування самописних

програмних кодів програмістів-початківців, які виступають результатами практичного вирішення запропонованих завдань з програмування на мові Java. В розробленому програмному продукті інтегровано компоненти з теоретичними відомостями з мови програмування Java, компоненти з пропонованими для вирішення практичними завданнями, компоненти для емуляції і імітації середовища розробки з автоматичною перевіркою коду за правилами синтаксису Java-мови.

Наукова новизна полягає у поєднанні сучасних алгоритмів синтаксичного аналізу, що базуються на граматиках операторного передування, з алгоритмами низхідного аналізу при трансляції програмного коду у створеному програмному застосунку.

Практичне значення роботи полягає у створені програмного застосунку, що поєднує в одному цілісному засобі електронний посібник з Java-програмування та середовище розробки програмного забезпечення вказаною мовою. Створений продукт дозволить користувачам з низьким рівнем комп'ютерної грамотності ознайомитись із базовими відомостями щодо написання програмного коду з застосуванням мови Java.

Список літератури

1. McConnell S. Rapid Development : Taming Wild Software Schedules. Pearson Education, 2014. 674 p.
2. Круглик В. С. Класифікація професійної діяльності інженера-програміста на основі аналізу ринку праці. Проблеми інженерно-педагогічної освіти : зб. наук. пр. Харків, 2016. № 50-51. С. 57-63.
3. Althoff C. The Self-Taught Programmer : The Definitive Guide to Programming Professionally. Kindle Edition, 2016. 299 p.
4. Сексенбаев К. Информационные технологии в развитии современного информационного общества. Молодой ученый. 2015. № 24 (104). С. 191-194.
5. Schildt H. Java : The Complete Reference. McGraw-Hill Education, 2014. 1312 p.
6. JavaRush. URL : https://javarush.ru (дата звернення: 08.11.2021).
7. Java. URL : https://www.codecademy.com/catalog/language/java (дата звернення: 08.11.2021).
8. Дейтел П., Дейтел Х., Дейтел Э. Android для разработчиков. СПб. : Питер, 2015. 384 с.
9. Humble H., Farley D. Continuous Delivery : Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional, 2011. 497 p.

References

1. McConnell S. (2014) Rapid Development : Taming Wild Software Schedules. Pearson Education. 674 p.
2. Kruhlyk V.S. (2016) [Klasyfikatsiia profesiinoi diialnosti inzhenera-prohramista na osnovi analizu rynku pratsi]. Problemy inzhenerno-pedahohichnoi osvity : zb. nauk. pr. Kharkiv. № 50-51. pp. 57-63. (in Ukrainian).
3. Althoff C. (2016) The Self-Taught Programmer : The Definitive Guide to Programming Professionally. Kindle Edition,. 299 p.
4. Seksenbayev K. (2015) Information technologies in the development of modern information society [Informatsionnyye tekhnologii v razvitii sovremennogo informatsionnogo obshchestva]. Young scientist. № 24 (104). P. 191-194. (in Russian).
5. Schildt H. (2014) Java : The Complete Reference. McGraw-Hill Education. 1312 p.
6. JavaRush. URL : https://javarush.ru (application date : 08.10.2021) (in Russian).
7. Java. URL : https://www.codecademy.com/catalog/language/java (дата звернення: 08.11.2021).
8. Deytel P., Deytel K., Deytel E. (2015) [Android dlya razrabotchikov]. Piter, 2015. 384 p. (in Russian).
9. Humble H., Farley D. (2011) Continuous Delivery : Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional. 497 p.

Надійшла до редакції 15.11.2021

O. Dmytriieva, I. Yarosh, T. Cherniak, P. Vorotylenko

DEVELOPMENT OF A MOBILE APPLICATION WITH BUILT-IN SERVICES FOR COMPILATION AND TESTING

Purpose. *The article is devoted to the design and development of a mobile Android-based educational system with built-in services for compiling and testing program listings written by the user as a result of solving Java programming tasks assigned to them.*

Methods. *The paper analyzes the problem which is resolved by mobile application development. The subject area analysis has been carried out. Its framework allows to investigate available means for studying the fundamentals of building codes in the Java language and analyze existing software compiling programs. Modern methods and approaches to compiling and testing a software tool have been investigated and evaluated, and the requirements for the functionality of the system being developed have been created.*

Results. *The Android application has been designed using a common UML methodology. The article includes option descriptions for using the software application, the states that it can take during operation, and the sequence of actions that the mobile software system needs to perform in order to function correctly and interact with users. Additionally, it presents the requirements for the human-machine interface of the developed system.*

Practical significance. *The results of software product development have been put in a form of implemented software classes. Functional features of the developed system have been considered under various communication user scenarios. It also covers the issues of interaction between the components of program screens, the calculation flows performed, and data transmission.*

Novelty. *Individual components, modules, and subsystems that make up the developed software tool have also been described. The article also defines and describes the sequence of user actions when performing basic operations provided for and implemented in the application for mobile devices powered by Android. The study result is represented by an application consisting of modules that support the educational process and program packages that ensure the operability of automation tools for preparing and executing user programs.*

Keywords: mobile application, compilation, testing, Android-based operating system, Java language, UML methodology, software classes.

Відомості про авторів

Дмитрієва Ольга Анатоліївна, докт. техн. наук, проф., зав. кафедри прикладної математики та інформатики ДВНЗ «Донецький національний технічний університет», e-mail: olga.dmytriyeva@donntu.edu.ua

Ярош Ірина Вікторівна, ст. викл. кафедри прикладної математики та інформатики ДВНЗ «Донецький національний технічний університет», e-mail: iryna.yarosh@donntu.edu.ua

Черняк Тетяна Олександрівна, асист. кафедри прикладної математики та інформатики ДВНЗ «Донецький національний технічний університет», e-mail: tetiana.cherniak@donntu.edu.ua

Воротиленко Павло Юрійович, бакалавр ДВНЗ «Донецький національний технічний університет», e-mail: pavlo.vorotylenko.kita@donntu.edu.ua

Dmytriieva Olha, Doctor of Technical Sciences, Professor, Head of the Department of Applied Mathematics and Informatics, SHEE "Donetsk National Technical University", e-mail: olga.dmytriyeva@donntu.edu.ua

Yarosh Iryna, Senior Lecturer of the Department of Applied Mathematics and Informatics, SHEE "Donetsk National Technical University", e-mail: iryna.yarosh@donntu.edu.ua

Cherniak Tetiana, Assistant of the Department of Applied Mathematics and Informatics, SHEE "Donetsk National Technical University", e-mail: tetiana.cherniak@donntu.edu.ua

Vorotylenko Pavlo, Bachelor Student, SHEE "Donetsk National Technical University", e-mail: pavlo.vorotylenko.kita@donntu.edu.ua