Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

студента *Мінзар Микити Андрійовича*

академічної групи *125-18-2*

спеціальності *125 Кібербезпека*

спеціалізації[1]

за освітньо-професійною програмою *Кібербезпека*

на тему *Система підвищення безпеки робочих процесів з використанням ОС Android на базі виявлення несанкціонованого доступу до критичних ресурсів*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	ст.викл. Саксонов Г.М.	80	добре	
розділів:				
спеціальний	ст.викл. Саксонов Г.М.	80	добре	
економічний	к.е.н., доц. Романюк Н.М.	80	добре	
Рецензент	к.т.н. Шедковський І.Н.	80	добре	
Нормоконтролер	ст.викл. Тимофєєв Д.С.	80	добре	

Дніпро
2022

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавра

студенту *Мінзар Микиті Андрійовичу* **академічної групи** *125-18-2*
(прізвище ім'я по-батькові) (шифр)

спеціальності *125 Кібербезпека*

за освітньо-професійною програмою *Кібербезпека*

на тему *Система підвищення безпеки робочих процесів з використанням ОС Android на базі виявлення несанкціонованого доступу до критичних ресурсів*

затверджену наказом ректора НТУ «Дніпровська політехніка» від 18.05.2022 № 268-с

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз кіберзагроз, а саме усіх актуальних на даний момент вразливостей у комп'ютерних системах та класифікацій атак на них, також розгляд історії створення ОС Android, його версій і проблем кожної з них.	04.02.2022 – 25.02.2022
Розділ 2	Проектування і програмна реалізація системи підвищення безпеки робочих процесів з обмеженням доступу до критичних ресурсів, а також розробка графічного інтерфейсу користувача.	22.03.2022 – 08.05.2022
Розділ 3	Розрахунок витрат на розробку програмного забезпечення та визначення економічної ефективності данного рішення.	13.05.2022 – 10.06.2022

Завдання видано _____ Саксонов Г.М.
(підпис керівника) (прізвище, ініціали)

Дата видачі: _____

Дата подання до екзаменаційної комісії: _____

Прийнято до виконання _____ Мінзар М.А.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 80с., 9 рис., 7 табл., 113 джерел.

Об'єктом дослідження є поняття безпеки робочих процесів з використанням ОС Android..

Предметом дослідження є методи і засоби безпеки робочих процесів з використанням ОС Android.

Мета роботи полягає в створенні власної програмної реалізації системи підвищення безпеки робочих процесів з використанням ОС Android на базі виявлення несанкціонованого доступу до критичних ресурсів.

Перший розділ представляє собою огляд основних понять предметної області.

У другому розділі проведено проектування і розробку системи підвищення безпеки робочих процесів з використанням ОС Android на базі виявлення несанкціонованого доступу до критичних ресурсів.

Третій розділ присвячено економічному аналізу доцільності розробки системи підвищення безпеки робочих процесів з використанням ОС Android на базі виявлення несанкціонованого доступу до критичних ресурсів.

КІБЕРБЕЗПЕКА, КІБЕРЗАГРОЗА, НЕСАНЦІОНОВАНИЙ ДОСТУП, ОС АНДРОЇД..

ABSTRACT

Explanatory note: 80p., 9 fig., 7 tables., 113 sources.

The object of research is the concept of workflow security using Android.

The subject of the study are methods and tools for security of workflows using Android.

The purpose of work in creation of own software implementation of system of increase of safety of work processes with use of OS OS on the basis of detection of unauthorized access to critical resources.

The first section is an overview of the basic concepts of the subject area.

The second section designed and developed a system for improving the security of workflows using Android based on the detection of unauthorized access to critical resources.

The third section is devoted to the economic analysis of the feasibility of developing a system for improving the security of workflows using Android OS based on the detection of unauthorized access to critical resources.

CYBER SECURITY, CYBER THREAT, UNAUTHORIZED ACCESS, ANDROID OS ..

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ОС — Операційна система

NIST — National Institute of Standards and Technology

GPL — General Public License

API — Application Programming Interface

ЗМІСТ

ВСТУП

В сучасному світі, при тенденції на розвиток науково-технічного прогресу і глобальну діджиталізацію, питання безпечності зберігання інформації постає все більш гостро.

Кожного дня з'являються нові незаконні засоби отримати конфіденційну інформацію користувача з будь-якого девайсу, будь то смартфон чи ПК.

У випадку Андроїд-девайсів для протидії створюються захищені сховища, які блокують доступ до файлів і не дають змоги викрасти файли.

Виходячи з усього вищесказаного, можна зробити висновок про високу актуальність поняття підвищення безпеки робочих процесів на смартфонах в сучасному світі.

Об'єктом дослідження є поняття безпеки робочих процесів з використанням ОС Android..

Предметом дослідження є методи і засоби безпеки робочих процесів з використанням ОС Android.

Мета роботи полягає в створенні власної програмної реалізації системи підвищення безпеки робочих процесів з використанням ОС Android на базі виявлення несанкціонованого доступу до критичних ресурсів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття кібербезпеки і кіберзагрози

Кібербезпека або безпека інформаційних технологій (IT-безпека) — це захист комп'ютерних систем і мереж від розкриття інформації, крадіжки або пошкодження їх апаратного забезпечення, програмного забезпечення або електронних даних, а також від порушення. або неправильне спрямування послуг, які вони надають.[1]

Ця галузь стала значною завдяки розширеній залежності від комп'ютерних систем, Інтернету [2] та стандартів бездротових мереж, таких як Bluetooth і Wi-Fi, а також через зростання «розумних» пристроїв, включаючи смартфони, телевізори та різні пристрої, які утворюють Інтернет речей (IoT). Кібербезпека також є однією з важливих проблем у сучасному світі через її складність як з точки зору політичного використання, так і з точки зору технологій. Його головна мета — забезпечити надійність, цілісність та конфіденційність даних системи.[3][4]

Історія

З моменту появи Інтернету та цифрової трансформації, розпочатої в останні роки, поняття кібербезпеки стало знайомим предметом як у нашому професійному, так і в особистому житті. Кібербезпека та кіберзагрози постійно присутні протягом останніх 50 років технологічних змін. У 1970-х і 1980-х роках комп'ютерна безпека була в основному обмежена науковими закладами, аж до появи концепції Інтернету, де, зі збільшенням можливостей підключення, почали з'являтися комп'ютерні віруси та вторгнення в мережу. Після поширення вірусів у 1990-х роках 2000-ті знаменували собою інституціоналізацію [необхідне роз'яснення] кіберзагроз і кібербезпеки.

Нарешті, з 2010-х років почали з'являтися масштабні атаки та урядові постанови.

Квітнева сесія 1967 року, організована Віллісом Уером на Весняній об'єднаній комп'ютерній конференції, і пізніша публікація Ware Report, були фундаментальними моментами в історії галузі комп'ютерної безпеки.[5] Роботи Уера опинилися на перетині матеріальних, культурних, політичних і соціальних проблем.[5]

Публікація NIST 1977 року[6] представила «тріаду ЦРУ» конфіденційності, цілісності та доступності як чіткий і простий спосіб опису ключових цілей безпеки.[7] Незважаючи на свою актуальність, з тих пір було запропоновано багато більш складних структур.[8][9]

Однак у 1970-х і 1980-х роках серйозних комп'ютерних загроз не було, оскільки комп'ютери та Інтернет все ще розвивалися, а загрози безпеці було легко ідентифікувати. Найчастіше погрози надходили від зловмисників, які отримали несанкціонований доступ до конфіденційних документів і файлів. Хоча зловмисне програмне забезпечення та порушення мережі існували протягом перших років, вони не використовували їх для фінансової вигоди. Однак до другої половини 1970-х років такі комп'ютерні фірми, як IBM, почали пропонувати комерційні системи контролю доступу та програмні продукти для комп'ютерної безпеки.[10]

Це почалося з Creeper у 1971 році. Creeper — це експериментальна комп'ютерна програма, написана Бобом Томасом у BBN. Вважається першим комп'ютерним хробаком.

У 1972 році було створено перше антивірусне програмне забезпечення під назвою Reaper. Його створив Рей Томлінсон, щоб переміщатися по мережі ARPANET і видаляти хробака Creeper.

У період з вересня 1986 року по червень 1987 року група німецьких хакерів здійснила перший задокументований випадок кібершпигунства. Група зламала американські оборонні підрядники, університети та мережі військових баз і продала зібрану інформацію радянському КДБ. Групу очолював Маркус Гесс, який був заарештований 29 червня 1987 року. Він був

засуджений за шпигунство (разом з двома співучасниками) 15 лютого 1990 року.

У 1988 році один з перших комп'ютерних хробаків під назвою хробак Морріса був поширений через Інтернет. Він привернув значну увагу масових ЗМІ.

У 1993 році Netscape почав розробляти протокол SSL, незабаром після того, як Національний центр суперкомп'ютерних додатків (NCSA) запустив Mosaic 1.0, перший веб-браузер, у 1993 році. Netscape мав готову версію SSL 1.0 у 1994 році, але вона так і не була опублікована. через багато серйозних вразливостей безпеки. Ці слабкі сторони включали атаки повторного відтворення та вразливість, яка дозволяла хакерам змінювати незашифровані повідомлення, надіслані користувачами. Однак у лютому 1995 року Netscape випустив версію 2.0.

Уразливості та атаки

Уразливість — це слабкість у проектуванні, реалізації, експлуатації або внутрішньому контролі. Більшість виявлених уразливостей задокументовано в базі даних Common Vulnerabilities and Exposures (CVE). [Потрібна цитата]. Уразливість, яка може бути використана, — це така, для якої існує принаймні одна робоча атака або «експлойт».[13] Уразливості можна досліджувати, перепроектувати, переслідувати або використовувати за допомогою автоматизованих інструментів або налаштованих скриптів.[14][15] Щоб захистити комп'ютерну систему, важливо розуміти атаки, які можуть бути здійснені проти неї, і ці загрози зазвичай можна класифікувати в одну з наведених нижче категорій:

Задні двері

Бекдор у комп'ютерній системі, криптосистемі або алгоритмі — це будь-який секретний метод обходу звичайної аутентифікації або контролю безпеки. Вони можуть існувати з багатьох причин, включаючи оригінальний дизайн або погану конфігурацію. Вони могли бути додані уповноваженою стороною, щоб дозволити певний законний доступ, або зловмисником із

зловмисних причин; але незалежно від мотивів свого існування вони створюють вразливість. Бекдори буває дуже важко виявити, і виявлення бекдорів зазвичай виявляє той, хто має доступ до вихідного коду програми або глибоко знає операційну систему комп'ютера.

Атака відмови в обслуговуванні

Атаки відмови в обслуговуванні (DoS) призначені для того, щоб зробити машину або мережевий ресурс недоступними для передбачуваних користувачів.[16] Зловмисники можуть відмовити в обслуговуванні окремих жертв, наприклад, навмисно вводячи неправильний пароль достатньо разів поспіль, щоб заблокувати обліковий запис жертви, або вони можуть перевантажити можливості машини чи мережі та заблокувати всіх користувачів одночасно. Хоча мережеву атаку з однієї IP-адреси можна заблокувати, додавши нове правило брандмауера, можливі багато форм атак із розподіленою відмовою в обслуговуванні (DDoS), коли атака відбувається з великої кількості точок, а захист набагато складніше. . Такі атаки можуть походити від зомбі-комп'ютерів бот-мережі або від низки інших можливих методів, включаючи атаки відображення та посилення, коли невинні системи обманюють, надсилаючи трафік жертві.

Атаки прямого доступу

Неавторизований користувач, який отримує фізичний доступ до комп'ютера, швидше за все, зможе безпосередньо скопіювати дані з нього. Вони також можуть поставити під загрозу безпеку, вносячи зміни в операційну систему, встановлюючи програмні хробаки, кейлоггери, приховані пристрої прослуховування або використовуючи бездротові мікрофони. Навіть якщо система захищена стандартними заходами безпеки, їх можна обійти, завантаживши іншу операційну систему або інструмент з компакт-диска чи іншого завантажувального носія. Шифрування диска та модуль Trusted Platform Module призначені для запобігання цим атакам.

Підслуховування

Підслуховування — це акт таємного прослуховування «розмови» (зв'язку) приватного комп'ютера, як правило, між хостами в мережі. Наприклад, такі програми, як Carnivore і NarusInSight, використовувалися ФБР і АНБ для прослуховування систем інтернет-провайдерів. Навіть машини, які працюють як закрита система (тобто без контакту із зовнішнім світом), можна підслухати за допомогою моніторингу слабких електромагнітних передач, які генеруються апаратними засобами; TEMPEST — це специфікація АНБ щодо цих атак.

Багатовекторні, поліморфні атаки

У 2017 році з'явився новий клас багатовекторних[17] поліморфних[18] кіберзагроз, які поєднували кілька типів атак і змінювали форму, щоб уникнути контролю кібербезпеки в міру поширення.

Фішинг

Фішинг – це спроба отримати конфіденційну інформацію, таку як імена користувачів, паролі та дані кредитної картки, безпосередньо від користувачів шляхом обману користувачів.[19] Фішинг, як правило, здійснюється шляхом підробки електронної пошти або обміну миттєвими повідомленнями, і він часто спонукає користувачів вводити дані на підробленому веб-сайті, чиї «вигляд» і «відчуття» майже ідентичні легальним. Підроблений веб-сайт часто запитує особисту інформацію, таку як дані для входу та паролі. Цю інформацію потім можна використовувати для отримання доступу до реального облікового запису особи на реальному веб-сайті. Фішинг можна класифікувати як різновид соціальної інженерії. Зловмисники використовують творчі способи отримати доступ до реальних акаунтів. Поширеним є шахрайство, коли зловмисники надсилають особам підроблені електронні рахунки-фактури[20], які показують, що вони нещодавно придбали музику, додатки чи інші, та інструктують їх натиснути посилання, якщо покупки не були авторизовані.

Підвищення привілеїв

Підвищення привілеїв описує ситуацію, коли зловмисник з деяким рівнем обмеженого доступу може без авторизації підвищити свої привілеї або рівень доступу. Наприклад, звичайний користувач комп'ютера може скористатися вразливістю в системі, щоб отримати доступ до обмежених даних; або навіть стати «root» і мати повний необмежений доступ до системи.

Зворотна інженерія

Зворотна інженерія — це процес, за допомогою якого створений людиною об'єкт деконструюється, щоб розкрити його конструкції, код, архітектуру або отримати знання з об'єкта; подібні до наукових досліджень, з тією лише різницею, що наукове дослідження стосується природніх явища.[21]: 3

Атака бічного каналу

Будь-яка обчислювальна система в тій чи іншій формі впливає на своє середовище. Цей вплив, який він має на навколишнє середовище, включає широкий спектр критеріїв, які можуть варіюватися від електромагнітного випромінювання до залишкового впливу на осередки оперативної пам'яті, що, як наслідок, робить можливою атаку холодного завантаження, до збоїв в реалізації апаратних засобів, які дозволяють отримати доступ та/або вгадати. інших значень, які зазвичай мають бути недоступними. У сценаріях атаки за боковим каналом зловмисник збирає таку інформацію про систему або мережу, щоб здогадатися про її внутрішній стан, і в результаті отримати доступ до інформації, яка, на думку жертви, є захищеною.

Соціальна інженерія

Соціальна інженерія в контексті комп'ютерної безпеки має на меті переконати користувача розкрити такі секрети, як паролі, номери карток тощо, або надати фізичний доступ, наприклад, видаючи себе за топ-менеджера, банк, підрядника чи клієнта. [22] Це, як правило, передбачає використання довіри людей і покладання на їхні когнітивні упередження.

Звичайна афера включає електронні листи, які надсилаються співробітникам бухгалтерського та фінансового відділу, видають себе за свого генерального директора та терміново вимагають певних заходів. На початку 2016 року ФБР повідомило, що подібні шахрайства з «компромісом ділової електронної пошти» (BEC) обійшлися американським підприємствам у понад 2 мільярди доларів приблизно за два роки.[23]

У травні 2016 року команда НБА «Мілуокі Бакс» стала жертвою такого типу кібер-шахрайства, коли зловмисник видав себе за президента команди Пітера Фейгіна, в результаті чого всі співробітники команди передали податкові бланки W-2 за 2015 рік.[24]

Підробка

Спуфінг - це акт маскування під дійсну сутність шляхом фальсифікації даних (наприклад, IP-адреса або ім'я користувача) з метою отримання доступу до інформації або ресурсів, які іншим чином отримати несанкціоновано.[25][26] Існує кілька типів спуфінгу, в тому числі:

- Підробка електронної пошти – це коли зловмисник підробляє адресу відправлення (від або джерела) електронного листа.
- Спуфінг IP-адреси, коли зловмисник змінює вихідну IP-адресу в мережевому пакеті, щоб приховати свою особистість або видати себе за іншу обчислювальну систему.
- Спуфінг MAC, коли зловмисник змінює адресу керування доступом до медіа (MAC) свого контролера мережевого інтерфейсу, щоб приховати свою особистість або видати себе за іншого.
- Біометричний спуфінг, коли зловмисник створює підроблений біометричний зразок, щоб представити себе іншим користувачем.[27]

Підробка

Підробка описує шкідливу модифікацію або зміну даних. Прикладами є так звані атаки Evil Maid і введення служб безпеки в маршрутизатори можливості спостереження.[28]

Шкідливе програмне забезпечення

Шкідливе програмне забезпечення (зловмисне програмне забезпечення), встановлене на комп'ютері, може витікати будь-яку інформацію, таку як особисту інформацію, бізнес-інформацію та паролі, може дати контроль над системою зловмиснику, а також може пошкодити або видалити дані назавжди.[29]

Мотивація нападника

Як і у випадку з фізичною безпекою, мотивація порушень комп'ютерної безпеки у зловмисників різна. Хтось шукає гострих відчуттів чи вандали, хтось – активіст, хтось – злочинці, які шукають фінансової вигоди. Зловмисники, спонсоровані державою, зараз поширені й забезпечені багатими ресурсами, але почали з аматорів, таких як Маркус Гесс, який зламав для КДБ, як розповідає Кліффорд Столл у «Яйце зозулі».

Крім того, останні мотиви нападників можна простежити до екстремістських організацій, які прагнуть отримати політичну перевагу або порушити соціальні програми.[100] Зростання Інтернету, мобільних технологій та недорогих комп'ютерних пристроїв призвело до збільшення можливостей, але також до ризику для середовищ, які вважаються життєво важливими для операцій. Усі критичні цільові середовища схильні до компромісу, і це призвело до серії активних досліджень щодо того, як перенести ризик, беручи до уваги мотивацію цих типів суб'єктів. Існує кілька різких відмінностей між мотивацією хакерів і мотивацією акторів національної держави, які намагаються атакувати на основі ідеологічних уподобань.[101]

Стандартною частиною моделювання загроз для будь-якої конкретної системи є визначення того, що може спонукати до атаки на цю систему, і хто може бути мотивований її зламати. Рівень і детальність запобіжних заходів будуть відрізнятися залежно від системи, яку потрібно захищати. Домашній персональний комп'ютер, банк і секретна військова мережа стикаються з дуже різними загрозами, навіть якщо основні технології, що використовуються, подібні.[102]

1.2 ОС Андроїд

Android — це мобільна операційна система, заснована на модифікованій версії ядра Linux та іншого програмного забезпечення з відкритим кодом, призначена в першу чергу для мобільних пристроїв із сенсорним екраном, таких як смартфони та планшети. Android розробляється консорціумом розробників, відомим як Open Handset Alliance і комерційно спонсорується Google. Він був представлений у листопаді 2007 року, а перший комерційний пристрій Android, HTC Dream, був запущений у вересні 2008 року.

Більшість версій Android є фірмовими. Основні компоненти взяті з Android Open Source Project (AOSP), який є безкоштовним програмним забезпеченням з відкритим кодом (FOSS), переважно ліцензованим за ліцензією Apache. Коли Android інстальовано на пристроях, можливість модифікувати програмне забезпечення FOSS в іншому випадку зазвичай обмежена, або не надається відповідний вихідний код, або запобігає повторне встановлення за допомогою технічних заходів, що робить встановлену версію власною. Більшість пристроїв Android постачаються з попередньо встановленим додатковим власним програмним забезпеченням[13], зокрема Google Mobile Services (GMS)[14], який включає основні програми, такі як Google Chrome, платформа цифрового розповсюдження Google Play і пов'язана платформа розробки служб Google Play.

Понад 70 відсотків смартфонів Android працюють на екосистемі Google; деякі з налаштованим постачальником інтерфейсом користувача та пакетом програмного забезпечення, наприклад TouchWiz і пізніше One UI від Samsung, і HTC Sense.[15] Екосистеми та форки Android-конкуренти включають Fire OS (розроблена Amazon), ColorOS від OPPO, OriginOS від vivo і MagicUI від Honor або користувальницькі ROM, такі як LineageOS. Однак назва та логотип «Android» є торговими марками Google, яка встановлює стандарти для обмеження використання бренду Android «несертифікованими» пристроями за межами їхньої екосистеми.[16][17]

Вихідний код використовувався для розробки варіантів Android на ряді іншої електроніки, наприклад, ігрових приставок, цифрових камер, портативних медіа-плеєрів, ПК, кожен із яких має спеціалізований інтерфейс користувача. Деякі добре відомі деривативи включають Android TV для телевізорів і Wear OS для носимих пристроїв, обидва розроблені Google. Пакети програмного забезпечення для Android, які використовують формат APK, зазвичай поширюються через власні магазини програм, як-от Google Play Store, Amazon Appstore (у тому числі для Windows 11), Samsung Galaxy Store, Huawei AppGallery, Cafe Bazaar і GetJar, або платформи з відкритим кодом, як-от Aptoide або F-Droid.

Android є найбільш продаваною ОС у світі на смартфонах з 2011 року і на планшетах з 2013 року. Станом на травень 2021 року вона налічує понад три мільярди активних користувачів щомісяця, що є найбільшою базою встановлених операційних систем, [18] і станом на січень 2021 року , Google Play Store містить понад 3 мільйони додатків.[19] Остання версія Android 12, випущена 4 жовтня 2021 року.[3]

Компанія Android Inc. була заснована в Пало-Альто, Каліфорнія, у жовтні 2003 року Енді Рубіном, Річем Майнером, Ніком Сірсом і Крісом Уайтом.[20][21] Рубін описав проект Android як «грандіозний потенціал у розробці розумних мобільних пристроїв, які краще знають місцезнаходження та вподобання свого власника».[21] Перші наміри компанії полягали в розробці передової операційної системи для цифрових камер, і це було основою її презентації інвесторам у квітні 2004 року.[22] Потім компанія вирішила, що ринок камер недостатньо великий для досягнення поставлених цілей, і через п’ять місяців вона відволіклася від своїх зусиль і представила Android як операційну систему для мобільних телефонів, яка буде конкурувати з Symbian та Microsoft Windows Mobile.[22][23]

Rubin мав труднощі із залученням інвесторів на ранньому етапі, і Android зіткнувся з виселенням зі свого офісного приміщення. Стів Перлман, близький друг Рубіна, приніс йому 10 000 доларів готівкою в конверті, а

невдовзі після цього передав нерозкриту суму як початкове фінансування. Перлман відмовився від частки в компанії і заявив: «Я зробив це, тому що вірив у це, і я хотів допомогти Енді».[24][25]

У 2005 році Рубін намагався домовитися про угоди з Samsung[26] і HTC.[27] Незабаром після цього Google придбала компанію в липні того ж року за щонайменше 50 мільйонів доларів;[21][28] це була «найкраща угода Google, коли-небудь» за словами тодішнього віце-президента Google із корпоративного розвитку Девіда Лоуі у 2010 році. 26] Ключові співробітники Android, включаючи Rubin, Miner, Sears і White, приєдналися до Google у рамках придбання.[21] У той час про секретну Android Inc. було відомо небагато, оскільки компанія надала лише кілька подробиць, крім того, що вона створює програмне забезпечення для мобільних телефонів.[21] У Google команда під керівництвом Рубіна розробила платформу мобільних пристроїв на базі ядра Linux. Google запропонувала платформу виробникам телефонів і операторам, пообіцявши надати гнучку систему з можливістю оновлення.[29] Google «підготував ряд апаратних компонентів та партнерів із програмного забезпечення та повідомив операторам, що він відкритий для різних ступенів співпраці».[30]

Спекуляції щодо намірів Google вийти на ринок мобільного зв'язку продовжували наростати до грудня 2006 року.[31] Ранній прототип був дуже схожий на телефон BlackBerry, без сенсорного екрану та фізичної QWERTY-клавіатури, але поява Apple iPhone 2007 року означала, що Android «повинен було повернутися до креслярської дошки».[32][33] Пізніше Google змінив свої документи специфікації Android, щоб вказати, що «сенсорні екрани будуть підтримуватися», хоча «Продукт був розроблений з наявністю дискретних фізичних кнопок як припущення, тому сенсорний екран не може повністю замінити фізичні кнопки».[34] У 2008 році як Nokia, так і BlackBerry анонсували сенсорні смартфони, які будуть конкурувати з iPhone 3G, і Android зрештою зосередився на сенсорних екранах. Першим

комерційно доступним смартфоном під управлінням Android був HTC Dream, також відомий як T-Mobile G1, анонсований 23 вересня 2008 року.[35][36]

5 листопада 2007 року Open Handset Alliance, консорціум технологічних компаній, включаючи Google, виробників пристроїв, таких як HTC, Motorola і Samsung, операторів бездротового зв'язку, таких як Sprint і T-Mobile, а також виробників чіпсетів, таких як Qualcomm і Texas Instruments, представив з метою розробки «першої дійсно відкритої та всеосяжної платформи для мобільних пристроїв».[37][38][39] Протягом року Open Handset Alliance зіткнувся з двома іншими конкурентами з відкритим кодом, Symbian Foundation і LiMo Foundation, останній також розробляв мобільну операційну систему на базі Linux, як-от Google. У вересні 2007 року InformationWeek висвітлював дослідження Evalueserve, в якому повідомлялося, що Google подала кілька патентних заявок у сфері мобільного телефонного зв'язку.[40][41]

Починаючи з 2008 року, Android отримав численні оновлення, які поступово покращували операційну систему, додаючи нові функції та виправляючи помилки в попередніх випусках. Кожен основний випуск названий в алфавітному порядку на честь десерту або солодкого частування, причому перші кілька версій Android мають назву «Cupcake», «Donut», «Eclair» і «Froyo», у цьому порядку. Під час свого анонсу Android KitKat у 2013 році Google пояснив, що «оскільки ці пристрої роблять наше життя таким солодким, кожна версія Android названа на честь десерту», хоча представник Google сказав CNN в інтерв'ю, що «це наче внутрішня команда і ми вважаємо за краще бути трішки — як би сказати — трохи незрозумілими, я скажу».[42]

У 2010 році Google запустила серію пристроїв Nexus, в якій Google співпрацює з різними виробниками пристроїв для виробництва нових пристроїв і впровадження нових версій Android. Серія була описана як «відіграла ключову роль в історії Android, вводячи нові ітерації програмного забезпечення та стандарти обладнання», і стала відомою своїм програмним

забезпеченням «без роздуття» з «своєчасними ... оновленнями».[43] На своїй конференції для розробників у травні 2013 року Google оголосив про спеціальну версію Samsung Galaxy S4, де замість використання власних налаштувань Android від Samsung, телефон працював на «стандартному Android», і йому обіцяли швидко отримувати нові оновлення системи.[44] Цей пристрій став початком програми версії Google Play, а за ним послідували інші пристрої, включаючи HTC One Google Play edition [45] і Moto G Google Play Edition [46]. У 2015 році Ars Technica писала, що «Раніше цього тижня останні телефони Android версії Google Play в інтернет-магазині Google були вказані як «більше недоступні для продажу» і що «Тепер їх усіх немає, і це виглядає як єдине ціле». схоже, що програма закінчилася".[47][48]

З 2008 по 2013 рік Хьюго Барра був представником продуктів, представляючи Android на прес-конференціях і Google I/O, щорічній конференції Google для розробників. Він залишив Google у серпні 2013 року, щоб приєднатися до китайського виробника телефонів Xiaomi.[49][50] Менш ніж за півроку раніше тодішній генеральний директор Google Ларрі Пейдж оголосив у своєму блозі, що Енді Рубін перейшов із підрозділу Android, щоб взятися за нові проекти в Google, і що Сундар Пічаї стане новим керівником Android.[51][52]] Сам Пічаї зрештою змінив посади, ставши новим генеральним директором Google у серпні 2015 року після реструктуризації компанії в конгломерат Alphabet,[53][54] зробивши Хіроші Локхаймера новим керівником Android.[55][56]

У Android 4.4 Kit Kat спільний доступ для запису до карт пам'яті MicroSD був заблокований для програм, встановлених користувачем, до яких залишалися доступні лише спеціальні каталоги з відповідними іменами пакетів, розташованих всередині Android/data/. Доступ для запису було відновлено в Android 5 Lollipop через зворотно несумісний інтерфейс Google Storage Access Framework.[57]

У червні 2014 року Google анонсувала Android One, набір «еферентних моделей апаратного забезпечення», які «дозволили б [виробникам пристроїв]

легко створювати високоякісні телефони за низькою ціною», призначених для споживачів у країнах, що розвиваються.[58][59] [60] У вересні Google оголосила про випуск першого набору телефонів Android One в Індії.[61][62] Однак у червні 2015 року Recode повідомила, що проект був «розчаруванням», посилаючись на «неохоче споживачів і виробничих партнерів» і «осічку в пошуковій компанії, яка ніколи не зламала обладнання».[63] Плани перезапустити Android One з'явилися в серпні 2015 року [64], а через тиждень Африка оголосили наступним місцем для програми.[65][66] У звіті The Information за січень 2017 року зазначено, що Google розширює свою недорогу програму Android One у Сполучених Штатах, хоча The Verge зазначає, що компанія, імовірно, не буде сама виробляти справжні пристрої.[67][68] Google представила смартфони Pixel і Pixel XL у жовтні 2016 року, які продавалися як перші телефони, виготовлені Google,[69][70] і ексклюзивно передбачали певні функції програмного забезпечення, такі як Google Assistant, до ширшого впровадження.[71][72]] Телефони Pixel замінили серію Nexus[73] на нове покоління телефонів Pixel, випущене в жовтні 2017 року.[74]

У травні 2019 року операційна система заплуталася у торговельній війні між Китаєм та Сполученими Штатами за участю Huawei, яка, як і багато інших технологічних компаній, стала залежною від доступу до платформи Android [75][76]. Влітку 2019 року Huawei оголосила, що створить альтернативну операційну систему Android[77], відому як Harmony OS,[78] і подала заявку на права інтелектуальної власності на основних світових ринках.[79][80] Під такими санкціями Huawei має довгострокові плани замінити Android у 2022 році новою операційною системою, оскільки ОС Harmony спочатку була розроблена для пристроїв Інтернету речей, а не для смартфонів і планшетів.[81]

22 серпня 2019 року було оголошено, що Android «Q» буде офіційно брендувати як Android 10, що припиняє історичну практику іменування основних версій на честь десертів. Google стверджує, що ці назви не є

«включними» для міжнародних користувачів (через те, що вищезгадані продукти не відомі на міжнародному рівні, або їх важко вимовити деякими мовами).[82][83] Того ж дня Android Police повідомила, що Google замовив установку статуї гігантського номера «10» у фойє нового офісу розробників.[84] Android 10 був випущений 3 вересня 2019 року вперше для телефонів Google Pixel.

З обмеженим сховищем звичайний доступ для запису до спільного внутрішнього сховища користувача було заблоковано, і як зазвичай доступні лише каталоги, що стосуються програми. Файли та каталоги за межами залишаються доступними лише через несумісну систему доступу до сховища. Хоча стверджується, що ці обмеження покращують конфіденційність користувачів, приватні каталоги для окремих програм уже існували в /data/ з ранніх версій операційної системи.[85]

Інтерфейс

Стандартний користувальницький інтерфейс Android в основному заснований на прямому маніпуляції з використанням сенсорного вводу, який практично відповідає реальним діям, таким як проведення пальцем, натискання, стискання і зворотне стискання для маніпулювання об’єктами на екрані, а також віртуальна клавіатура.[86] Ігрові контролери та повнорозмірні фізичні клавіатури підтримуються через Bluetooth або USB.[87][88] Реакція на введення користувача розроблена так, щоб бути негайною та забезпечує плавний сенсорний інтерфейс, часто використовуючи можливості вібрації пристрою для надання тактильного зворотного зв’язку користувачеві. Внутрішнє апаратне забезпечення, таке як акселерометри, гіроскопи та датчики наближення, використовуються деякими програмами для реакції на додаткові дії користувача, наприклад, для налаштування екрана з портретної на альбомну в залежності від того, як орієнтований пристрій, [89] або надання користувачеві можливості керувати транспортний засіб у гоночній грі шляхом обертання пристрою, що імітує керування кермом.[90]

Домашній екран

Пристрої Android завантажуються на головний екран, основний навігаційний та інформаційний «центр» на пристроях Android, аналогічний робочому столу на персональних комп'ютерах. Головні екрани Android зазвичай складаються з піктограм програм і віджетів; Піктограми додатків запускають пов'язану програму, тоді як віджети відображають живий, автоматично оновлюваний вміст, наприклад, прогноз погоди, електронну скриньку користувача або мітку новин безпосередньо на головному екрані.[91] Головний екран може складатися з кількох сторінок, між якими користувач може гортати вперед і назад.[92] Додатки сторонніх розробників, доступні в Google Play та інших магазинах додатків, можуть значно переоформити головний екран [93] і навіть імітувати зовнішній вигляд інших операційних систем, таких як Windows Phone.[94] Більшість виробників налаштовують зовнішній вигляд і функції своїх пристроїв Android, щоб відрізнятися від своїх конкурентів.[95]

Рядок стану

У верхній частині екрана знаходиться рядок стану, що відображає інформацію про пристрій і його підключення. Цю рядок стану можна потягнути (протягнути) вниз, щоб відкрити екран сповіщень, на якому програми відображають важливу інформацію або оновлення, а також швидкий доступ до системних елементів керування та перемикачів, таких як яскравість дисплея, налаштування підключення (WiFi, Bluetooth, мобільні дані), аудіо режим і ліхтарик.[92] Постачальники можуть застосувати розширені налаштування, такі як можливість регулювання яскравості ліхтарика.[96]

Сповіщення

Сповіщення — це «коротка, своєчасна та релевантна інформація про вашу програму, коли вона не використовується», і при натисканні користувачі спрямовуються на екран всередині програми, що стосується сповіщення.[97] Починаючи з Android 4.1 «Jelly Bean», «розширювані сповіщення» дозволяють користувачеві торкнутися значка на сповіщенні,

щоб воно розгорнулося та відобразило більше інформації та можливі дії програми прямо з сповіщення.[98]

Списки програм

Екран «Усі програми» містить список усіх встановлених програм, з можливістю для користувачів перетягнути програму зі списку на головний екран. Доступ до списку програм можна отримати за допомогою жесту або кнопки, залежно від версії Android. Екран «Останні», також відомий як «Огляд», дозволяє користувачам перемикатися між нещодавно використаними програмами.[92]

Останній список може з'являтися поруч або накладатися, залежно від версії Android і виробника.[99]

Програмний стек

Крім ядра Linux, є проміжне програмне забезпечення, бібліотеки та API, написані на C, а також прикладне програмне забезпечення, що працює на базі додатків, яка включає Java-сумісні бібліотеки. Розробка ядра Linux продовжується незалежно від інших проектів вихідного коду Android.

Android використовує Android Runtime (ART) як середовище виконання (введене у версії 4.4), яке використовує попередню компіляцію (AOT) для повної компіляції байт-коду програми в машинний код після встановлення програми. У Android 4.4 ART була експериментальною функцією і не ввімкнена за замовчуванням; він став єдиним варіантом виконання в наступній основній версії Android, 5.0. У версіях, які більше не підтримуються, до версії 5.0, коли ART прийшов до влади, Android раніше використовував Dalvik як процесну віртуальну машину з компіляцією Just-in-time (JIT) на основі трасування для запуску Dalvik «dex-code» (Dalvik Executable), який зазвичай перекладається з байт-коду Java. Дотримуючись принципу JIT на основі трасування, окрім інтерпретації більшості коду програми, Dalvik виконує компіляцію та власне виконання вибраних часто виконуваних сегментів коду («слідів») кожного разу під час запуску програми. Для своєї бібліотеки Java платформа Android використовує

підмножину проекту Apache Harmony, який зараз припинено. У грудні 2015 року Google оголосив, що наступна версія Android перейде на реалізацію Java на основі проекту OpenJDK.

Стандартна бібліотека C для Android, Bionic, була розроблена Google спеціально для Android як похід від стандартного коду бібліотеки C BSD. Сам Bionic був розроблений з кількома основними функціями, характерними для ядра Linux. Основні переваги використання Bionic замість бібліотеки GNU C (glibc) або uClibc полягають у меншому розмірі часу виконання та оптимізації для низькочастотних процесорів. У той же час Bionic ліцензується за умовами ліцензії BSD, яку Google вважає більш підходящою для загальної моделі ліцензування Android.

Прагнучи до іншої моделі ліцензування, до кінця 2012 року Google змінив стек Bluetooth в Android з BlueZ, ліцензованого GPL, на BlueDroid з ліцензією Apache. Новий стек Bluetooth під назвою Gabeldorsche був розроблений, щоб спробувати виправити помилки в реалізації BlueDroid.

Android не має рідної системи X Window за замовчуванням, а також не підтримує повний набір стандартних бібліотек GNU. Це ускладнювало перенесення існуючих програм або бібліотек Linux на Android, доки версія r5 Android Native Development Kit не забезпечила підтримку програм, повністю написаних на C або C++. Бібліотеки, написані на C, можуть також використовуватися в програмах шляхом введення невеликої прокладки та використання JNI.

У поточних версіях Android «Toybox», набір утиліт командного рядка (в основному для використання програмами, оскільки Android не надає інтерфейс командного рядка за замовчуванням), використовується (з моменту випуску Marshmallow) замість аналогічного Колекція "Toolbox" знайдена в попередніх версіях Android

Android має іншу операційну систему, Trusty OS, як частину програмних компонентів Trusty, що підтримують середовище довіреного виконання (TEE) на мобільних пристроях. «Trusty та API Trusty можуть бути

змінені. Програми для ОС Trusty можна писати на C/C++ (підтримка C++ обмежена), і вони мають доступ до невеликої бібліотеки C. Усі Trusty програми є однопоточними; багатопоточність у просторі користувача Trusty наразі не підтримується. Розробка додатків сторонніх розробників не підтримується в поточній версії, а програмне забезпечення, що працює на ОС і процесор для нього, запускає "платформу DRM для захищеного. Існує багато інших варіантів використання TEE, таких як мобільні платежі, безпечне банківське обслуговування, повне шифрування диска, багатофакторна аутентифікація, захист від скидання пристрою, постійне сховище, захищене від повторного відтворення, бездротове відображення ("трансляція") захищений вміст, безпечна обробка PIN-коду та відбитків пальців і навіть виявлення шкідливого програмного забезпечення».

Загальні загрози безпеці

У дослідженні компанії Trend Micro з питань безпеки зловживання послугами преміум-класу є найпоширенішим типом шкідливого програмного забезпечення Android, коли текстові повідомлення надсилаються із заражених телефонів на платні телефонні номери без згоди або навіть відома користувача. Інші шкідливі програми відображають небажану та нав'язливу рекламу на пристрої або надсилають особисту інформацію неавторизованим третім сторонам. Як повідомляється, загрози безпеці на Android зростають в геометричній прогресії; однак інженери Google стверджують, що зловмисне програмне забезпечення та вірусна загроза для Android перебільшується охоронними компаніями з комерційних причин, і звинувачують індустрію безпеки у грі на страхах продавати програмне забезпечення для захисту від вірусів користувачам. Google стверджує, що небезпечне зловмисне програмне забезпечення насправді надзвичайно рідкісне, а опитування, проведене F-Secure, показало, що лише 0,5% зловмисного програмного забезпечення для Android надходило з магазину Google Play.

У 2021 році журналісти та дослідники повідомили про виявлення шпигунського програмного забезпечення під назвою Pegasus, розробленого

та розповсюджуваного приватною компанією, яке часто використовувалося і використовувалося для зараження смартфонів iOS і Android – частково за допомогою 0-денних експлойтів – без необхідності будь-яку взаємодію з користувачем або значущі підказки для користувача, а потім використовуватися для вилучення даних, відстеження місцезнаходження користувача, зйомки фільму через його камеру та активації мікрофона в будь-який час. Аналіз трафіку даних популярними смартфонами під керуванням Android-версій виявив значний збір даних за замовчуванням і обмін без відмови від цього попередньо встановленого програмного забезпечення. Обидві ці проблеми не вирішуються або не можуть бути вирішені за допомогою виправлень безпеки.

Патчі безпеки

У серпні 2015 року Google оголосила, що пристрої серії Google Nexus почнуть отримувати щомісячні виправлення безпеки. Google також написав, що "пристрої Nexus будуть отримувати основні оновлення протягом щонайменше двох років і виправлення безпеки протягом більше трьох років з моменту початкової доступності або 18 місяців після останнього продажу пристрою через Google Store". Наступного жовтня дослідники з Кембриджського університету дійшли висновку, що 87,7% телефонів Android, які використовувалися, мали відомі, але не виправлені вразливості безпеки через відсутність оновлень і підтримки. Рон Амадео з Ars Technica також написав у серпні 2015 року, що «Android спочатку був розроблений, перш за все, для широкого поширення. Google починала з нуля з нульовою часткою ринку, тому була рада відмовитися від контролю та дати кожному місце. Проте зараз Android займає приблизно 75–80 відсотків світового ринку смартфонів, що робить її не просто найпопулярнішою операційною системою для мобільних пристроїв у світі, але, можливо, найпопулярнішою операційною системою, і крапка . Тому безпека стала великою проблемою. Android все ще використовує ланцюг керування оновленням програмного забезпечення, розроблений ще тоді, коли екосистема Android не мала

пристроїв для оновлення, і це просто не працює".[266] Після новин про щомісячний розклад Google деякі виробники, включаючи Samsung і LG, пообіцяли випускати щомісячні оновлення безпеки, але, як зазначив Джеррі Хілденбранд у Android Central у лютому 2016 року, «замість цього ми отримали кілька оновлень щодо конкретних версій маленька жменя моделей. І купа порушених обіцянок».

У березні 2017 року в дописі Google Security Blog керівники з безпеки Android Адріан Людвіг і Мел Міллер написали, що «понад 735 мільйонів пристроїв від 200+ виробників отримали оновлення безпеки платформи в 2016 році» і що «наші оператори та партнери з обладнання допомогли розширити розгортання ці оновлення, які випустили оновлення для понад половини з 50 найкращих пристроїв у всьому світі в останньому кварталі 2016 року». Вони також написали, що «приблизно половина пристроїв, які використовувалися наприкінці 2016 року, не отримували оновлення безпеки платформи в попередньому році», заявляючи, що їхня робота й надалі буде зосереджена на оптимізації програми оновлень безпеки для спрощення розгортання виробниками. 269] Крім того, у коментарі TechCrunch Людвіг заявив, що час очікування оновлень безпеки було скорочено з «шести до дев'яти тижнів до кількох днів», при цьому 78% флагманських пристроїв у Північній Америці оновлюються. безпеки наприкінці 2016 року.

Виправлені помилки, виявлені в основній операційній системі, часто не охоплюють користувачів старих і дешевих пристроїв. Однак природа Android із відкритим кодом дозволяє підрядникам із безпеки використовувати наявні пристрої та адаптувати їх для високобезпечного використання. Наприклад, Samsung співпрацювала з General Dynamics через придбання Open Kernel Labs, щоб відновити Jelly Bean поверх свого загартованого мікровізора для проекту "Knox".

Подальші помітні подвиги

У 2018 році норвезька охоронна фірма Promon виявила серйозну діру в безпеці Android, яку можна використати для крадіжки облікових даних для

входу, повідомлень доступу та відстеження місцезнаходження, які можна було знайти в усіх версіях Android, включаючи Android 10. Уразливість виникла через використання помилка в системі багатозадачності, що дозволяє шкідливому додатку накладати законні програми фальшивими екранами входу, про які користувачі не знають, передаючи облікові дані безпеки. Користувачів також можна обдурити, щоб надати додаткові дозволи шкідливим програмам, які згодом дозволять їм виконувати різноманітні шкідливі дії, включаючи перехоплення текстів або дзвінків і крадіжку банківських облікових даних. Avast Threat Labs також виявила, що багато попередньо встановлених програм на кількох сотнях нових пристроях Android містять небезпечне зловмисне та рекламне програмне забезпечення. Деякі з попередньо встановлених зловмисних програм можуть вчинити шахрайство з рекламою або навіть заволодіти його хост-пристроєм.

У 2020 році, який? Watchdog повідомляє, що понад мільярд пристроїв Android, випущених у 2012 році або раніше, що становило 40% пристроїв Android у всьому світі, піддалися ризику злому. Цей висновок випливає з того факту, що у 2019 році жодних оновлень безпеки для версій Android нижче 7.0 не було. Яке? співпрацював з антивірусною лабораторією AV Comparatives, щоб заразити п’ять моделей телефонів шкідливим програмним забезпеченням, і в кожному випадку це вдалося. Google відмовився коментувати спекуляції сторожового пса.

5 серпня 2020 року Twitter опублікував блог, у якому закликав своїх користувачів оновити свої програми до останньої версії щодо проблем безпеки, які дозволяли іншим отримати доступ до прямих повідомлень. Хакер може легко використовувати «системні дозволи Android», щоб отримати облікові дані облікового запису для цього. Проблема безпеки виникає лише з Android 8 (Android Oreo) і Android 9 (Android Pie). Twitter підтвердив, що оновлення програми обмежить такі дії.

Характеристики технічної безпеки

Програми Android працюють у пісочниці, ізольованій області системи, яка не має доступу до решти ресурсів системи, якщо користувач не надає дозволи на доступ під час інсталяції програми, однак це може бути неможливим для попереднього встановлені програми. Неможливо, наприклад, вимкнути доступ до мікрофона попередньо встановленої програми камери, не вимкнувши камеру повністю. Це справедливо також у версіях Android 7 та 8.

З лютого 2012 року Google використовує свій сканер шкідливих програм Google Bouncer для перегляду та сканування програм, доступних у магазині Google Play. У листопаді 2012 року в рамках версії операційної системи Android 4.2 «Jelly Bean» була представлена функція «Перевіряти програми» для перевірки всіх додатків як із Google Play, так і зі сторонніх джерел на наявність зловмисної поведінки. Спочатку, роблячи це лише під час встановлення, у 2014 році Verify Apps отримав оновлення для «постійного» сканування програм, а в 2017 році ця функція стала видимою для користувачів через меню в налаштуваннях.

Перш ніж інсталювати програму, магазин Google Play відображає список вимог, необхідних для роботи програми. Переглянувши ці дозволи, користувач може прийняти або відмовитися від них, встановлюючи програму, лише якщо він приймає. В Android 6.0 «Marshmallow» була змінена система дозволів; програмам більше не надаються автоматично всі вказані дозволи під час встановлення. Замість цього використовується система підписки, в якій користувачам пропонується надати або відхилити окремі дозволи додатку, коли вони потрібні вперше. Програми запам'ятовують гранти, які користувач може скасувати в будь-який момент. Однак попередньо встановлені програми не завжди є частиною цього підходу. У деяких випадках неможливо відмовити в певних дозволах попередньо встановленим програмам або вимкнути їх. Програму Сервіси Google Play не можна видалити або вимкнути. Будь-яка спроба примусової зупинки призведе до перезапуску програми. Нова модель дозволів використовується

лише програмами, розробленими для Marshmallow з використанням комплекту розробки програмного забезпечення (SDK), а старіші програми продовжуватимуть використовувати попередній підхід «все або нічого». Дозволи все ще можна скасувати для цих програм, хоча це може завадити їм працювати належним чином, і про це відображається попередження.

У вересні 2014 року Джейсон Нова з Android Authority повідомив про дослідження німецької безпекової компанії Fraunhofer AISEC щодо антивірусного програмного забезпечення та загроз шкідливих програм на Android. Nova написала, що «Операційна система Android працює з програмними пакетами, зберігаючи їх у пісочниці; це не дозволяє програмам перераховувати вміст каталогів інших програм, щоб захистити систему. Не дозволяючи антивірусу відображати каталоги інших програм після встановлення, програми, які під час завантаження не виявляють підозрілої поведінки, очищаються як безпечні. Якщо пізніше будуть активовані частини програми, які виявляться шкідливими, антивірус не зможе дізнатися, оскільки він знаходиться всередині програми та поза антивірусом «юрисдикція». Дослідження Fraunhofer AISEC, яке вивчало антивірусне програмне забезпечення від Avast, AVG, Bitdefender, ESET, F-Secure, Kaspersky, Lookout, McAfee (раніше Intel Security), Norton, Sophos і Trend Micro, показало, що «тестовані антивірусні програми не забезпечують захист від налаштованого шкідливого програмного забезпечення або цільових атак», і що «тестовані антивірусні програми також не змогли виявити шкідливе програмне забезпечення, яке на сьогоднішній день абсолютно невідоме, але не робить жодних зусиль, щоб приховати його шкідливість».

У серпні 2013 року Google анонсував Диспетчер пристроїв Android (перейменований у травні 2017 року «Знайти мій пристрій»), сервіс, який дозволяє користувачам віддалено відстежувати, знаходити та стирати свої пристрої Android, за допомогою Програма для Android для служби випущена в грудні. У грудні 2016 року Google представила додаток Trusted Contacts, що дозволяє користувачам запитувати відстеження місцезнаходження близьких

людей під час надзвичайних ситуацій. У 2020 році довірені контакти було закрито, а функція надсилання геоданих увійшла в Карти Google.

8 жовтня 2018 року Google оголосила про нові вимоги до магазину Google Play для боротьби з надмірним поширенням потенційно конфіденційної інформації, включаючи журнали викликів і текстових повідомлень. Проблема пов'язана з тим, що багато програм запитують дозволи на доступ до особистої інформації користувачів (навіть якщо ця інформація не потрібна для роботи програми), а деякі користувачі, безсумнівно, надають ці дозволи. Крім того, у маніфесті програми може бути зазначено дозвіл (на відміну від необов'язкового), і програма не буде встановлена, якщо користувач не надасть дозвіл; користувачі можуть відкликати будь-які, навіть необхідні, дозволи для будь-якої програми в налаштуваннях пристрою після встановлення програми, але небагато користувачів роблять це. Google пообіцяв працювати з розробниками і створювати винятки, якщо їхні програми вимагають дозволу на телефон або SMS для «основної функціональності програми». Застосування нової політики розпочалося 6 січня 2019 року, через 90 днів після оголошення політики 8 жовтня 2018 року. Крім того, Google оголосила про нову «вимогу цільового рівня API» (targetSdkVersion у маніфесті) принаймні Android 8.0 (рівень API 26) для всіх нові програми та оновлення програм. Вимога рівня API може боротися з практикою, коли розробники додатків обходять деякі екрани дозволів, вказуючи ранні версії Android, які мали більш грубу модель дозволу.

РОЗДІЛ 2
ПРОЕКТУВАННЯ І ПРОГРАМНА РЕАЛІЗАЦІЯ

2.1 Проектування майбутнього функціоналу

Діаграма використання найпростіша - це представлення взаємодії користувача із системою, яка показує взаємозв'язок між користувачем та різними випадками використання, в яких користувач бере участь. Діаграма випадків використання може ідентифікувати різні типи користувачів системи та різні випадки використання, і часто вона супроводжується також іншими типами діаграм. Варіанти використання представлені кругами або еліпсами.

Незважаючи на те, що сам випадок використання може детально вивчити кожну можливість, діаграма прикладів використання може допомогти забезпечити огляд системи на більш високому рівні. Раніше вже було сказано, що "схеми використання - це принципи вашої системи".

Через їх спрощений характер, схеми використання можуть бути хорошим інструментом комунікації для зацікавлених сторін. Креслення намагаються імітувати реальний світ і дають зацікавленій стороні уявлення про те, як буде розроблена система. Сіау та Лі провели дослідження, щоб визначити, чи взагалі існувала дійсна ситуація для схем використання або вони були непотрібними. Було виявлено, що діаграми випадків використання передають намір системи більш спрощеним чином зацікавленим сторонам і що вони "інтерпретуються більш повно, ніж діаграми класів".

Метою діаграми використання є відображення динамічного аспекту системи. Додаткові схеми та документація можуть бути використані для забезпечення повного функціонального та технічного уявлення про систему. Вони забезпечують спрощене та графічне представлення того, що система насправді повинна робити.

Елементи:

- рамки системи (англ. system border) - прямоугольник із назвою у верхніх частинах та еліпсами (прецедентами) всередині. Часто може бути опущено без корисної інформації про полезну інформацію,
- актор (англ. актор) - стилізований людський персонаж, обозначаючий набір ролей користувача (розуміється в широкому змісті: людина, внешняя сутність, клас, інша система), взаємодіючого з деякою сутністю (системною, підсистемою, класом). Актори не можуть бути пов'язані між собою з іншим (за вимкнення відносин щодо обробки / дослідження),
- прецедент - еліпс із надписом, що обозначає виконувану систематичну дію (може включати можливі варіанти), що призводить до спостерігаємих акторами результатів. Надпис може бути ім'ям або описом (з точки зору актора) того, "що" робить система (а не "як"). Ім прецедента зв’язано з неперервним (атомарним) сценарієм - конкретною послідовністю дій, ілюстріруючою повіддю. Під час сценарію актори обмінюються із систематичними повідомленнями. Сценарій може бути приведений на діаграмі прецедентів у відео UML-коментарі. З одним прецедентом може бути пов'язано кілька різних сценаріїв

На рисунку 2.1 зображено діаграму варіантів використання, яка описує можливі дії користувача в системі.

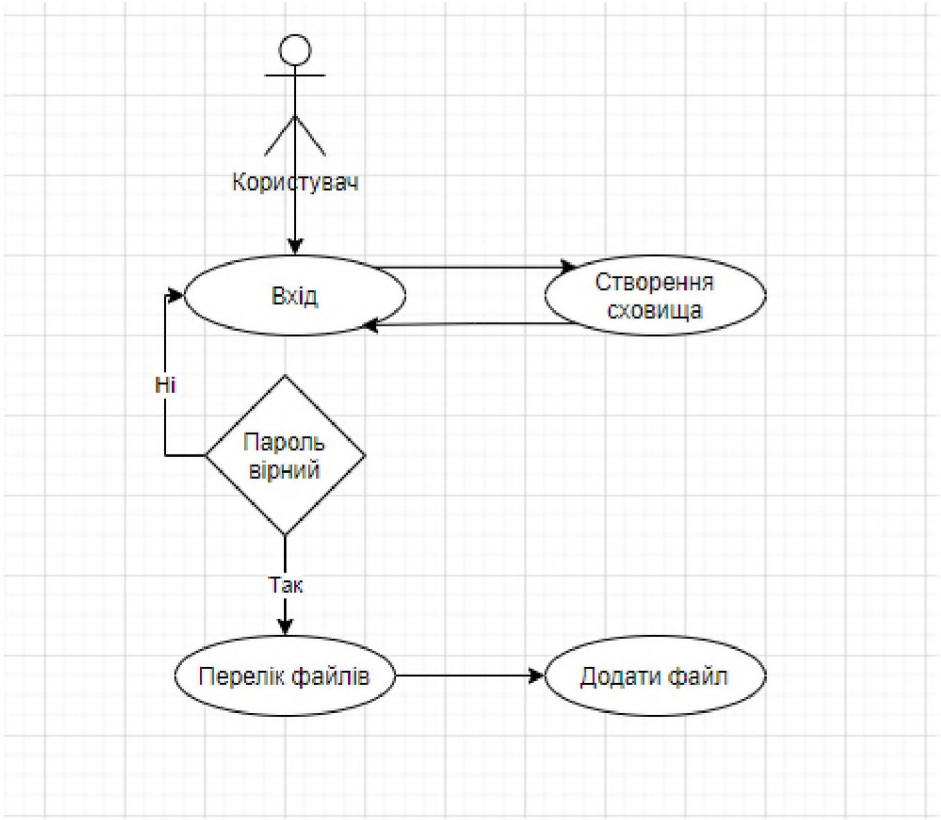

Рисунок 2.1 — Діаграма варіантів використання

2.2 Огляд інструментальних засобів розробки

Java — це високорівнева, заснована на класах, об'єктно-орієнтована мова програмування, яка розроблена так, щоб мати якомога менше залежностей реалізації. Це мова програмування загального призначення, призначена для того, щоб програмісти могли писати один раз, запускати в будь-якому місці (WORA),[17] що означає, що скомпільований код Java може працювати на всіх платформах, які підтримують Java, без необхідності перекомпіляції.[18] Програми Java зазвичай компілюються у байт-код, який може працювати на будь-якій віртуальній машині Java (JVM) незалежно від базової архітектури комп'ютера. Синтаксис Java подібний до C і C++, але має менше засобів низького рівня, ніж будь-який з них. Середовище виконання Java надає динамічні можливості (такі як відображення та модифікація коду під час виконання), які зазвичай недоступні в традиційних скомпільованих мовах. Станом на 2019 рік Java була однією з найпопулярніших мов програмування, що використовуються відповідно до GitHub, [19][20] особливо для веб-додатків клієнт-сервер, із 9 мільйонами розробників.[21]

Спочатку Java була розроблена Джеймсом Гослінгом у Sun Microsystems і випущена в травні 1995 року як основний компонент Java-платформи Sun Microsystems. Оригінальні та довідкові компілятори Java, віртуальні машини та бібліотеки класів спочатку були випущені Sun за власними ліцензіями. Станом на травень 2007 року, відповідно до специфікацій Java Community Process, Sun переліцензувала більшість своїх технологій Java за ліцензією лише GPL-2.0. Oracle пропонує власну віртуальну машину HotSpot Java, однак офіційною довідковою реалізацією є OpenJDK JVM, яка є безкоштовним програмним забезпеченням з відкритим вихідним кодом і використовується більшістю розробників і є JVM за замовчуванням для майже всіх дистрибутивів Linux.

Станом на березень 2022 року Java 18 є останньою версією, тоді як Java 17, 11 і 8 є поточними версіями довгострокової підтримки (LTS). Oracle випустила останнє загальнодоступне оновлення за нульовою вартістю для

застарілої версії Java 8 LTS у січні 2019 року для комерційного використання, хоча в іншому випадку вона все ще підтримуватиме Java 8 з загальнодоступними оновленнями для особистого використання на невизначений термін. Інші виробники почали пропонувати безкоштовні збірки OpenJDK 8 і 11, які все ще отримують безпеку та інші оновлення.

Oracle (та інші) настійно рекомендують видаляти застарілі та непідтримувані версії Java через невирішені проблеми безпеки в старих версіях.[22] Oracle радить своїм користувачам негайно перейти на підтримувану версію, наприклад одну з версій LTS (8, 11, 17).

Джеймс Гослінг, Майк Шерідан і Патрік Нотон ініціювали проект мови Java у червні 1991 року.[23] Спочатку Java була розроблена для інтерактивного телебачення, але в той час вона була надто просунутою для індустрії цифрового кабельного телебачення.[24] Спочатку мова називалася Oak на честь дуба, що стояв біля офісу Гослінга. Пізніше проект отримав назву Green і був остаточно перейменований в Java, від кави Java, типу кави з Індонезії.[25] Гослінг розробив Java із синтаксисом у стилі C/C++, який був би знайомим системним і прикладним програмістам.[26]

Sun Microsystems випустила першу публічну реалізацію як Java 1.0 у 1996 році.[27] Він обіцяв писати один раз, запускати будь-де (WORA), забезпечуючи безкоштовний час виконання на популярних платформах. Досить безпечний і з можливістю налаштування безпеки, він дозволяв обмеження доступу до мережі та файлів. Незабаром основні веб-браузери включили можливість запускати Java-аплети на веб-сторінках, і Java швидко стала популярною. Компілятор Java 1.0 був переписаний на Java Артуром ван Хоффом, щоб суворо відповідати специфікації мови Java 1.0.[28] З появою Java 2 (спочатку випущена як J2SE 1.2 у грудні 1998 – 1999 років), нові версії мали кілька конфігурацій, створених для різних типів платформ. J2EE містив технології та API для корпоративних програм, які зазвичай працюють у серверних середовищах, тоді як J2ME містив API, оптимізовані для мобільних додатків. Настільна версія була перейменована в J2SE. У 2006 році

в маркетингових цілях Sun перейменувала нові версії J2 на Java EE, Java ME та Java SE відповідно.

У 1997 році Sun Microsystems звернулася до органу стандартів ISO/IEC JTC 1, а пізніше до Ecma International, щоб формалізувати Java, але незабаром вийшла з процесу.[29][30][31] Java залишається де-факто стандартом, який контролюється через процес Java Community.[32] Свого часу Sun зробила більшість своїх реалізацій Java доступними безкоштовно, незважаючи на статус їх власного програмного забезпечення. Sun отримувала дохід від Java за рахунок продажу ліцензій на спеціалізовані продукти, такі як Java Enterprise System.

13 листопада 2006 року Sun випустила більшу частину своєї віртуальної машини Java (JVM) як безкоштовне програмне забезпечення з відкритим кодом (FOSS) за умовами ліцензії GPL-2.0. 8 травня 2007 року Sun завершила процес, зробивши весь основний код своєї JVM доступним на умовах вільного програмного забезпечення/розповсюдження з відкритим кодом, за винятком невеликої частини коду, на яку Sun не володіла авторським правом.[33]

Віце-президент Sun Річ Грін сказав, що ідеальною роллю Sun щодо Java була роль євангеліста.[34] Після придбання корпорацією Oracle Sun Microsystems у 2009–2010 роках Oracle назвала себе розпорядником технологій Java з невпинним прагненням підтримувати спільноту участі та прозорості.[35] Це не завадило Oracle незабаром після цього подати позов проти Google за використання Java всередині Android SDK (див. розділ Android).

2 квітня 2010 року Джеймс Гослінг пішов з Oracle.[36]

У січні 2016 року Oracle оголосила, що середовища виконання Java, засновані на JDK 9, припинять плагін для браузера.[37]

Програмне забезпечення Java працює на всьому, від ноутбуків до центрів обробки даних, ігрових консолей до наукових суперкомп'ютерів.[38]

Java JVM і байт-код

Однією з цілей дизайну Java є переносимість, що означає, що програми, написані для платформи Java, повинні працювати подібним чином на будь-якій комбінації апаратного забезпечення та операційної системи з належною підтримкою часу виконання. Це досягається шляхом компіляції коду мови Java у проміжне подання, яке називається байт-кодом Java, замість безпосереднього машинного коду, специфічного для архітектури. Інструкції байт-коду Java аналогічні машинному коду, але вони призначені для виконання віртуальною машиною (VM), написаною спеціально для апаратного забезпечення хоста. Кінцеві користувачі зазвичай використовують Java Runtime Environment (JRE), встановлену на своєму пристрої для автономних програм Java або веб-браузер для Java-аплетів.

Стандартні бібліотеки надають загальний спосіб доступу до специфічних для хоста функцій, таких як графіка, потоки та мережа.

Використання універсального байт-коду спрощує перенесення. Однак накладні витрати на інтерпретацію байт-коду в машинні інструкції змушували інтерпретовані програми майже завжди працювати повільніше, ніж рідні виконувані файли. Компілятори Just-in-time (JIT), які компілюють байт-код у машинний код під час виконання, були представлені на ранньому етапі. Компілятор Hotspot Java насправді є двома компіляторами в одному; і з GraalVM (включений, наприклад, у Java 11, але вилучений з Java 16), що дозволяє багаторівневу компіляцію.[47] Сама Java не залежить від платформи і адаптована до конкретної платформи, на якій вона має працювати, за допомогою віртуальної машини Java (JVM), яка перекладає байт-код Java на машинну мову платформи.[48]

Продуктивність

Програми, написані на Java, мають репутацію повільніших і вимагають більше пам'яті, ніж програми, написані на C++.[49][50] Однак швидкість виконання програм Java значно покращилася із запровадженням компіляції «точно вчасно» в 1997/1998 роках для програм на Java. Java 1.1,[51] додавання мовних функцій, які підтримують кращий аналіз коду (наприклад,

внутрішні класи, клас StringBuilder, необов'язкові твердження тощо), а також оптимізації у віртуальній машині Java, наприклад HotSpot, ставши JVM Sun за замовчуванням у 2000 році. З Java 1.5 продуктивність була покращена за допомогою додавання пакета java.util.concurrent, включаючи безблоковані реалізації ConcurrentMaps та інших багатоядерних колекцій, а також покращено з Java 1.6.

Не-JVM

Деякі платформи пропонують пряму апаратну підтримку Java; існують мікроконтролери, які можуть запускати байт-код Java в апаратному забезпеченні замість програмної віртуальної машини Java,[52] і деякі процесори на базі ARM можуть мати апаратну підтримку для виконання байт-коду Java через їх опцію Jazelle, хоча в поточних реалізаціях підтримку здебільшого було припинено. APM.

Автоматичне управління пам'яттю

Java використовує автоматичний збірник сміття для управління пам'яттю в життєвому циклі об'єкта. Програміст визначає час створення об'єктів, а середовище виконання Java відповідає за відновлення пам'яті, коли об'єкти більше не використовуються. Як тільки посилання на об'єкт не залишаються, недоступна пам'ять стає придатною для автоматичного звільнення збирачем сміття. Щось подібне до витоку пам'яті все ще може статися, якщо код програміста містить посилання на об'єкт, який більше не потрібен, як правило, коли об'єкти, які більше не потрібні, зберігаються в контейнерах, які все ще використовуються. Якщо викликаються методи для неіснуючого об'єкта, створюється виняток нульового покажчика.[53][54]

Одна з ідей, що лежить в основі моделі автоматичного управління пам'яттю Java, полягає в тому, що програмісти можуть позбутися від тягаря виконання ручного керування пам'яттю. У деяких мовах пам'ять для створення об'єктів неявно виділяється в стеку або явно виділяється і звільняється з купи. В останньому випадку відповідальність за управління пам'яттю покладається на програміста. Якщо програма не звільняє об'єкт,

відбувається витік пам’яті. Якщо програма намагається отримати доступ або звільнити пам’ять, яка вже була вивільнена, результат буде невизначеним і важко передбачити, і програма, швидше за все, стане нестабільною або аварійно завершить роботу. Це можна частково виправити за допомогою розумних покажчиків, але вони додають накладних витрат і ускладнюють. Зауважте, що збір сміття не запобігає витоку логічної пам’яті, тобто тих, де пам’ять все ще посилається, але ніколи не використовується.

Збір сміття може статися в будь-який момент. В ідеалі це відбуватиметься, коли програма неактивна. Він гарантовано спрацьовує, якщо в купі недостатньо вільної пам’яті для виділення нового об’єкта; це може призвести до миттєвої зупинки програми. Явне керування пам'яттю неможливе в Java.

Java не підтримує арифметику вказівників у стилі C/C++, де адресами об’єктів можна арифметично маніпулювати (наприклад, додаючи або віднімаючи зміщення). Це дозволяє збірнику сміття переміщувати об’єкти, на які посилаються, і гарантує безпеку та безпеку типу.

Як і в C++ та деяких інших об'єктно-орієнтованих мовах, змінні примітивних типів даних Java зберігаються або безпосередньо в полях (для об'єктів), або в стеку (для методів), а не в купі, як це зазвичай вірно для непримітивних даних. типи (але див. аналіз escape). Це було свідоме рішення дизайнерів Java з міркувань продуктивності.

Java містить кілька типів збирачів сміття. Починаючи з Java 9, HotSpot використовує Garbage First Garbage Collector (G1GC) за замовчуванням.[55] Однак є також кілька інших збирачів сміття, які можна використовувати для керування купою. Для більшості програм на Java достатньо G1GC. Раніше в Java 8 використовувався Parallel Garbage Collector.

Вирішення проблеми управління пам’яттю не звільняє програміста від тягаря належної обробки інших видів ресурсів, таких як мережеві з’єднання або підключення до бази даних, дескриптори файлів тощо, особливо за наявності винятків.

Синтаксис

На синтаксис Java значною мірою впливають C++ і C. На відміну від C++, який поєднує синтаксис для структурованого, загального та об'єктно-орієнтованого програмування, Java була створена майже виключно як об'єктно-орієнтована мова.[18] Весь код записується всередині класів, і кожен елемент даних є об’єктом, за винятком примітивних типів даних (тобто цілих чисел, чисел з плаваючою комою, логічних значень і символів), які не є об’єктами з міркувань продуктивності. Java повторно використовує деякі популярні аспекти C++ (наприклад, метод printf).

На відміну від C++, Java не підтримує перевантаження операторів[56] або множинного наслідування для класів, хоча для інтерфейсів підтримується множинне успадкування.[57]

Java використовує коментарі, схожі на коментарі C++. Існує три різні стилі коментарів: стиль одного рядка, позначений двома косими рисками (//), стиль кількох рядків, відкритий за допомогою /* і закритий */, і стиль коментування Javadoc, відкритий за допомогою /** і закритий */. . Стиль коментування Javadoc дозволяє користувачеві запускати виконуваний файл Javadoc для створення документації для програми і може бути прочитаний деякими інтегрованими середовищами розробки (IDE), такими як Eclipse, щоб дозволити розробникам отримати доступ до документації в IDE.

Привіт, світовий приклад

Традиційну програму Hello world можна написати на Java як:[58]

```
публічний клас HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Привіт, світ!"); // Виводить рядок на консоль.
    }
}
```

Усі вихідні файли мають бути названі відповідно до загальнодоступного класу, який вони містять, із додаванням суфікса .java, наприклад HelloWorldApp.java. Спочатку його потрібно скомпілювати в байт-код, використовуючи компілятор Java, створюючи файл із суфіксом .class (в даному випадку HelloWorldApp.class). Тільки після цього його можна виконати або запустити. Вихідний файл Java може містити лише один відкритий клас, але він може містити кілька класів з модифікатором непублічного доступу та будь-якою кількістю відкритих внутрішніх класів. Коли вихідний файл містить кілька класів, необхідно зробити один клас (введений ключовим словом class) відкритим (перед ним буде ключове слово public) і назвати вихідний файл цим ім'ям загальнодоступного класу.

Клас, який не оголошено публічним, може зберігатися в будь-якому файлі .java. Компілятор створить файл класу для кожного класу, визначеного у вихідному файлі. Ім'я файлу класу є ім'ям класу з додаванням .class. Для створення файлів класів анонімні класи розглядаються так, як якщо б їх ім'я було конкатенацією назви їх класу, що охоплює, $ і цілого числа.

Ключове слово public означає, що метод може бути викликаний з коду в інших класах, або що клас може використовуватися класами поза ієрархією класів. Ієрархія класів пов'язана з ім'ям каталогу, в якому знаходиться файл .java. Це називається модифікатором рівня доступу. Інші модифікатори рівня доступу включають ключові слова private (метод, до якого можна отримати доступ лише в тому самому класі) і protected (що дозволяє отримати доступ коду з того самого пакета). Якщо фрагмент коду намагається отримати доступ до приватних методів або захищених методів, JVM видає виключення SecurityException

Ключове слово static[19] перед методом вказує на статичний метод, який асоціюється лише з класом, а не з будь-яким конкретним екземпляром цього класу. Без посилання на об'єкт можна викликати лише статичні методи. Статичні методи не можуть отримати доступ до членів класу, які

також не є статичними. Методи, які не визначені як статичні, є методами екземплярів і для роботи вимагають певного екземпляра класу.

Ключове слово void вказує на те, що основний метод не повертає жодного значення абоненту. Якщо програма Java має завершити роботу з кодом помилки, вона повинна викликати System.exit() явно.

Ім'я методу main не є ключовим словом у мові Java. Це просто назва методу, який викликає програма запуску Java, щоб передати керування програмі. Класи Java, які працюють у керованих середовищах, таких як аплети та Enterprise JavaBeans, не використовують і не потребують методу main(). Програма Java може містити кілька класів, які мають основні методи, а це означає, що віртуальній машині потрібно чітко вказати, з якого класу запускати.

Основний метод повинен приймати масив об'єктів String. За домовленістю він посилається як args, хоча можна використовувати будь-яке інше юридичне ім’я ідентифікатора. Починаючи з Java 5, основний метод також може використовувати змінні аргументи у формі public static void main(String... args), що дозволяє викликати основний метод з довільною кількістю аргументів String. Ефект цього альтернативного оголошення семантично ідентичний (параметру args, який все ще є масивом об’єктів String), але він дозволяє використовувати альтернативний синтаксис для створення та передачі масиву.

Програма запуску Java запускає Java, завантажуючи заданий клас (вказаний у командному рядку або як атрибут у JAR) і запускаючи його публічний статичний метод main(String[]). Автономні програми повинні оголошувати цей метод явно. Параметр String[] args — це масив об’єктів String, що містить будь-які аргументи, передані класу. Параметри в main часто передаються за допомогою командного рядка.

Друк є частиною стандартної бібліотеки Java: клас System визначає відкрите статичне поле, яке викликається. Об’єкт out є екземпляром класу PrintStream і надає багато методів для друку даних на стандартний вихід,

включаючи println(String), який також додає новий рядок до переданого рядка.

Рядок "Hello World!" компілятор автоматично перетворює об’єкт String.

Реалізація

Корпорація Oracle є поточним власником офіційної реалізації платформи Java SE після придбання Sun Microsystems 27 січня 2010 року. Ця реалізація заснована на оригінальній реалізації Java від Sun. Реалізація Oracle доступна для Microsoft Windows (досі працює для XP, поки офіційно підтримуються лише новіші версії), macOS, Linux і Solaris. Оскільки Java не має жодної офіційної стандартизації, визнаної Ecma International, ISO/IEC, ANSI або іншими сторонніми організаціями зі стандартизації, реалізація Oracle є стандартом де-факто.

Реалізація Oracle складається з двох різних дистрибутивів: Java Runtime Environment (JRE), який містить частини платформи Java SE, необхідні для виконання програм Java і призначений для кінцевих користувачів, і Java Development Kit (JDK), який призначений для розробників програмного забезпечення та включає такі засоби розробки, як компілятор Java, Javadoc, Jar і налагоджувач. Oracle також випустила GraalVM, високопродуктивний динамічний компілятор та інтерпретатор Java.

OpenJDK — це ще одна помітна реалізація Java SE, яка ліцензована за GNU GPL. Реалізація почалася, коли Sun почала випускати вихідний код Java під ліцензією GPL. Починаючи з Java SE 7, OpenJDK є офіційною довідковою реалізацією Java.

Мета Java - зробити всі реалізації Java сумісними. Історично, ліцензія Sun на використання торгової марки Java наполягає на тому, щоб усі реалізації були сумісні. Це призвело до юридичної суперечки з Microsoft після того, як Sun заявила, що реалізація Microsoft не підтримує RMI або JNI і має власні функції, специфічні для платформи. Sun подала до суду в 1997 році, а в 2001 році виграла угоду в розмірі 20 мільйонів доларів США, а

також ухвалу суду про виконання умов ліцензії Sun.[77] Як наслідок, Microsoft більше не постачає Java разом із Windows.

Незалежна від платформи Java є важливою для Java EE, і для сертифікації реалізації потрібна ще більш сувора перевірка. Це середовище дозволяє переносити додатки на стороні сервера.

Android

Мова Java є ключовою опорою Android, мобільної операційної системи з відкритим кодом. Хоча Android, побудований на ядрі Linux, написаний переважно на C, Android SDK використовує мову Java як основу для додатків Android, але не використовує жодного зі своїх стандартних GUI, SE, ME або інших встановлених стандартів Java.[78] Мова байт-коду, яку підтримує Android SDK, несумісна з байт-кодом Java і працює на власній віртуальній машині, оптимізованій для пристроїв із низьким рівнем пам'яті, таких як смартфони та планшетні комп'ютери. Залежно від версії Android, байт-код або інтерпретується віртуальною машиною Dalvik, або компілюється в рідний код за допомогою середовища виконання Android.

Android не надає повної стандартної бібліотеки Java SE, хоча Android SDK містить незалежну реалізацію великої її підмножини. Він підтримує Java 6 і деякі функції Java 7, пропонуючи реалізацію, сумісну зі стандартною бібліотекою (Apache Harmony).

Android Studio — це офіційне[7] інтегроване середовище розробки (IDE) для операційної системи Google Android, побудоване на програмному забезпеченні IntelliJ IDEA від JetBrains і розроблене спеціально для розробки Android.[8] Він доступний для завантаження в операційних системах на базі Windows, macOS та Linux або як послуга на основі підписки у 2020 році.[9][10] Це заміна Eclipse Android Development Tools (E-ADT) як основної IDE для розробки додатків Android.

Android Studio було анонсовано 16 травня 2013 року на конференції Google I/O. Він був на стадії попереднього перегляду, починаючи з версії 0.1 у травні 2013 року, а потім перейшов на стадію бета-тестування, починаючи з

версії 0.8, яка була випущена в червні 2014 року.[11] Перша стабільна збірка була випущена в грудні 2014 року, починаючи з версії 1.0.[12]

7 травня 2019 року Kotlin замінив Java як бажану мову Google для розробки додатків для Android.[13] Java все ще підтримується, як і C++.[14]

Особливості

- У поточній стабільній версії надаються такі функції:[15][16]
- Підтримка збірки на основі Gradle
- Рефакторинг для Android і швидкі виправлення
- Інструменти Lint для виявлення проблем продуктивності, зручності використання, сумісності версій та інших проблем
- Інтеграція ProGuard та можливості підписання додатків
- Майстри на основі шаблонів для створення звичайних дизайнів і компонентів Android
- Розширений редактор макетів, який дозволяє користувачам перетягувати компоненти інтерфейсу користувача, можливість попереднього перегляду макетів на кількох конфігураціях екрана[17]
- Підтримка створення програм Android Wear
- Вбудована підтримка Google Cloud Platform, що забезпечує інтеграцію з Firebase Cloud Messaging (раніше «Google Cloud Messaging») і Google App Engine[18]
- Віртуальний пристрій Android (Емулятор) для запуску та налагодження програм у студії Android.

Android Studio підтримує всі ті ж мови програмування IntelliJ (і CLion), наприклад. Java, C++ та багато іншого з розширеннями, такими як Go;[19] і Android Studio 3.0 або новішої версії підтримують Kotlin[20] і «всі мовні функції Java 7 і підмножина функцій мови Java 8, які відрізняються залежно від версії платформи».[21] Зовнішні проекти підтримують деякі функції Java 9.[22] Хоча IntelliJ стверджує, що Android Studio підтримує всі випущені версії Java і Java 12, незрозуміло, на якому рівні Android Studio підтримує версії Java аж до Java 12 (документація згадує часткову підтримку Java 8).

Принаймні деякі нові мовні функції аж до Java 12 можна використовувати в Android.[23]

Після компіляції програми за допомогою Android Studio її можна опублікувати в магазині Google Play. Додаток має відповідати політиці щодо вмісту Google Play Store.

2.3 Проектування пакетної будови

Уся система розділена на окремі шари, кожен з яких несе свою логічну мету. Діаграма пакетів зображена на рисунку 2.2

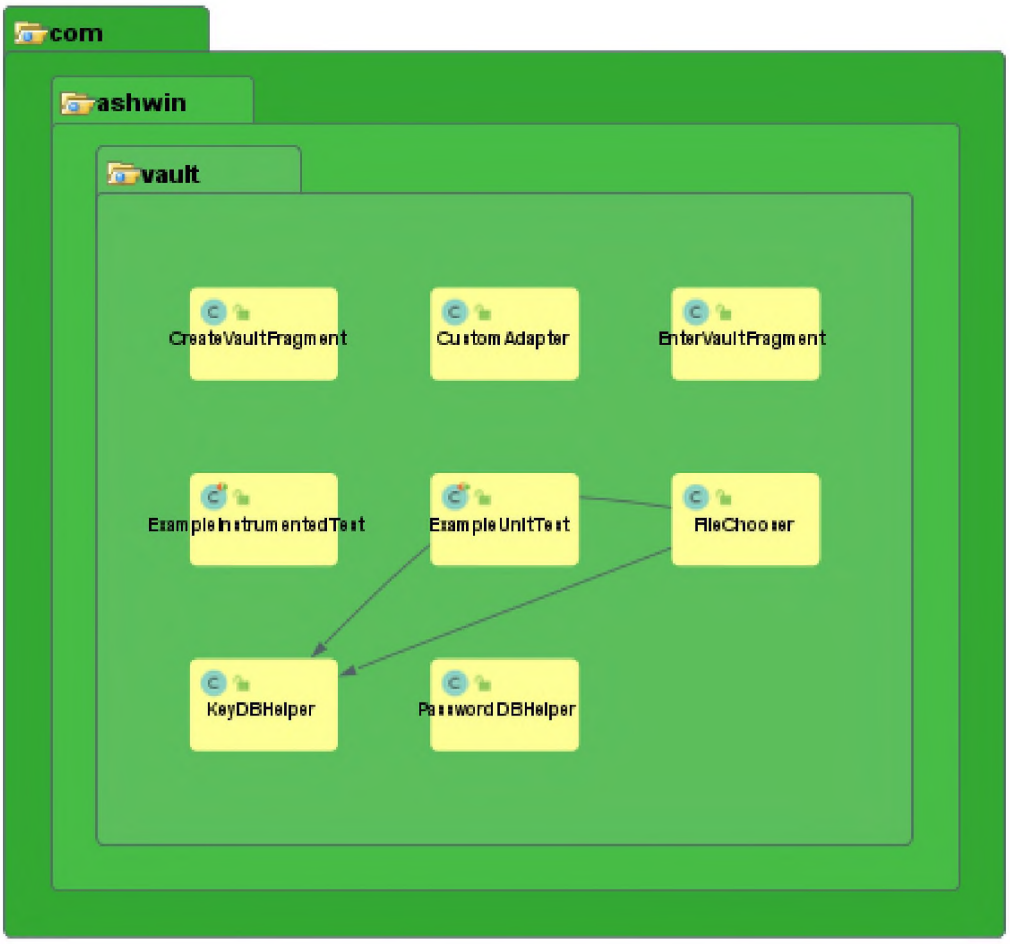

Рисунок 2.2 — Діаграма пакетів

2.4 Проектування структури класів

У програмній інженерії діаграма класів в Уніфікованій мові моделювання (UML) - це тип статичної структурної діаграми, що описує структуру системи, показуючи класи системи, їх атрибути, операції (або методи) та взаємозв'язки між об'єктами.

Діаграма класів є основним будівельним елементом об'єктно-орієнтованого моделювання. Він використовується для загального концептуального моделювання структури програми та для детального моделювання переведення моделей у програмовий код. Діаграми класів також можуть бути використані для моделювання даних. Класи на діаграмі класів представляють як основні елементи, взаємодії в програмі, так і класи, що програмуються.

На схемі класи представлені вікнами, які містять три відділення:

- У верхньому відділенні міститься назва класу. Надруковано жирним шрифтом і відцентровано, а перша літера написана великими літерами.
- Середній відсік містить атрибути класу. Вони вирівняні за лівим краєм, а перша буква мала.
- У нижньому відділенні містяться операції, які може виконувати клас. Вони також вирівняні за лівим краєм, а перша буква - мала.

При проектуванні системи ряд класів ідентифікується та згруповується у схему класів, яка допомагає визначити статичні відносини між ними. При детальному моделюванні класи концептуального проекту часто поділяються на ряд підкласів.

Залежність - це семантичний зв'язок між залежними та незалежними елементами моделі. Він існує між двома елементами, якщо зміни у визначенні одного елемента (сервера або цілі) можуть спричинити зміни для іншого (клієнта або джерела). Ця асоціація є односпрямованою. Залежність відображається у вигляді штрихової лінії з відкритою стрілкою, яка вказує від клієнта до постачальника.

Для подальшого опису поведінки систем ці діаграми класів можуть бути доповнені діаграмою стану або машиною стану UML.

Асоціація представляє родину посилань. Двійкова асоціація (з двома кінцями) зазвичай представляється у вигляді рядка. Асоціація може пов’язувати будь-яку кількість класів. Асоціація з трьома ланками називається потрійною асоціацією. Асоціацію можна назвати, а кінці асоціації можна прикрасити іменами ролей, показниками власності, кратністю, видимістю та іншими властивостями.

Існує чотири різні типи асоціацій: двонаправлена, односпрямована, агрегаційна (включає агрегацію композиції) та рефлексивна. Двонаправлені та односпрямовані асоціації є найбільш поширеними.

Наприклад, клас польоту асоціюється з класом літака двонаправлено. Асоціація представляє статичне відношення, яке ділиться між об'єктами двох класів.

Агрегація є варіантом взаємозв'язку "має"; агрегація є більш конкретною, ніж асоціація. Це асоціація, яка представляє частково цілі або часткові стосунки. Як показано на зображенні, професор "має" клас для викладання. Як тип асоціації, агрегація може бути названа та мати ті самі прикраси, що і асоціація. Однак агрегація не може включати більше двох класів; це має бути бінарна асоціація. Крім того, навряд чи існує різниця між агрегаціями та асоціаціями під час реалізації, і діаграма може взагалі пропустити відносини агрегування. [7]

Агрегація може відбуватися, коли клас є колекцією або контейнером інших класів, але вміщені класи не мають сильної залежності життєвого циклу від контейнера. Вміст контейнера все ще існує, коли контейнер знищений.

В UML він графічно представлений у вигляді порожнистої форми ромба на вміщуючому класі одним рядком, що зв’язує його із вміщеним класом. Сукупність - це семантично розширений об’єкт, який у багатьох

операціях трактується як одиниця, хоча фізично він складається з декількох менших об’єктів.

Приклад: Бібліотека та студенти. Тут студент може існувати без бібліотеки, зв'язок між студентом і бібліотекою є агрегацією.

Це вказує на те, що один із двох пов’язаних класів (підклас) вважається спеціалізованою формою іншого (супер тип), а суперклас - узагальненням підкласу. На практиці це означає, що будь-який екземпляр підтипу є також екземпляром суперкласу. Зразкове дерево узагальнень цієї форми зустрічається в біологічній класифікації: людина - це підклас маймуни, який є підкласом ссавців тощо. Зв’язок найлегше зрозуміти за допомогою фрази „A - це B” (людина - це ссавець, ссавець - тварина).

Графічне представлення UML узагальнення - це форма порожнистого трикутника на кінці суперкласу рядка (або дерева рядків), що зв’язує його з одним або кількома підтипами.

Відносини узагальнення також відомі як спадщина або відносини "є".

Суперклас (базовий клас) у відносинах узагальнення також відомий як "батьківський", суперклас, базовий клас або базовий тип.

Підтип у відносинах спеціалізації також відомий як "дочірній", підклас, похідний клас, похідний тип, клас успадкування або тип успадкування.

Зверніть увагу, що ці стосунки нічим не схожі на біологічні стосунки батьків та дітей: використання цих термінів надзвичайно поширене, але може ввести в оману.

A - це тип B

Наприклад, "дуб - це тип дерева", "автомобіль - це тип транспортного засобу"

Узагальнення може бути показано лише на діаграмах класів та на діаграмах використання.

При моделюванні UML взаємозв'язок реалізації - це взаємозв'язок між двома елементами моделі, в яких один елемент моделі (клієнт) реалізує

(реалізує або виконує) поведінку, яку вказує інший елемент моделі (постачальник).

Графічне представлення UML реалізації - це порожниста форма трикутника на кінці інтерфейсу штрихової лінії (або дерева рядків), яка з'єднує її з одним або кількома реалізаторами. Проста головка стрілки використовується на кінці інтерфейсу штрихової лінії, що з'єднує її з користувачами. У діаграмах компонентів використовується графічна умова «м'яч і сокет» (реалізатори виставляють кульку або льодяник, тоді як користувачі показують сокет). Реалізації можна показати лише на діаграмах класів або компонентів. Реалізація - це взаємозв'язок між класами, інтерфейсами, компонентами та пакетами, що з'єднує елемент клієнта з елементом постачальника. Зв'язок реалізації між класами / компонентами та інтерфейсами показує, що клас / компонент реалізує операції, пропоновані інтерфейсом.

Залежність - це слабша форма зв'язку, яка вказує на те, що один клас залежить від іншого, оскільки він використовує його в певний момент часу. Один клас залежить від іншого, якщо незалежний клас є змінною параметра або локальною змінною методу залежного класу. Це відрізняється від асоціації, де атрибут залежного класу є екземпляром незалежного класу. Іноді відносини між двома класами дуже слабкі. Вони взагалі не реалізовані зі змінними-членами. Швидше вони можуть бути реалізовані як аргументи функції-члена.

інший. Ці відносини зазвичай описуються як "А має В" (у матері-кота є кошенята, у кошенят - мати-кішка).

Представлення UML асоціації - це лінія, що з'єднує два пов'язані класи. На кожному кінці рядка є додаткові позначення. Наприклад, ми можемо вказати, використовуючи наконечник стрілки, що загострений кінець видно з хвоста стрілки. Ми можемо вказати власність шляхом розміщення кульки, ролі, яку відіграють елементи цього кінця, вказавши ім'я ролі та множинність

екземплярів цієї сутності (діапазон кількості об'єктів, які беруть участь в асоціації з точки зору іншого кінця).

Класи сутності моделюють довгоживучу інформацію, якою обробляє система, а іноді і поведінку, пов'язану з цією інформацією. Їх не слід ідентифікувати як таблиці баз даних чи інших сховищ даних.

Вони намальовані як кола з короткою лінією, прикріпленою до нижньої частини кола. Як варіант, їх можна намалювати як звичайні класи із позначенням стереотипу «сутність» над назвою класу.

На рисунку 2.3 зображено діаграму класів системи.

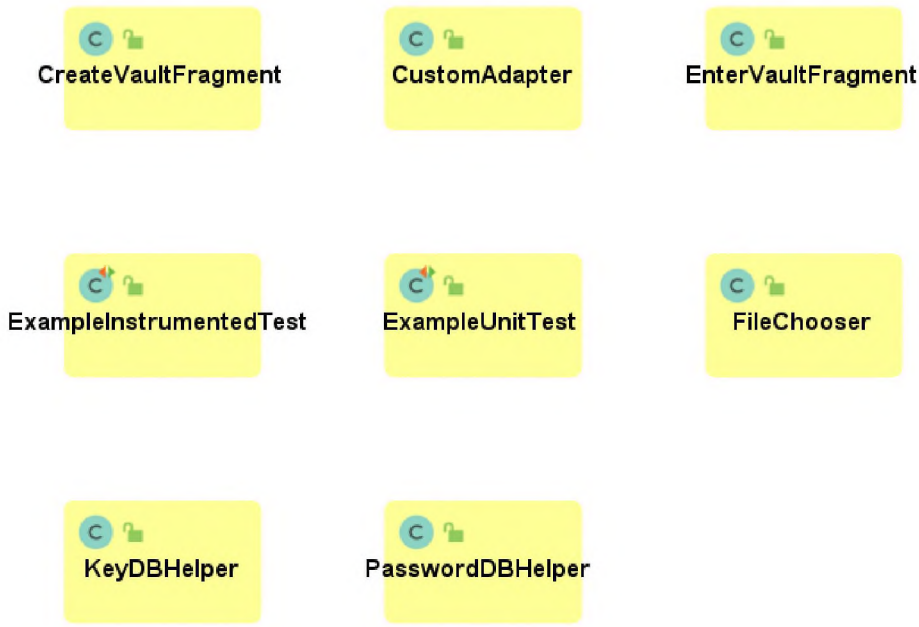

Рисунок 2.3 — Діаграма класів

2.5 Розробка графічного інтерфейсу користувача

Графічний інтерфейс складається з декількох основних активностей, скріншоти і коди яких приведено нижче.

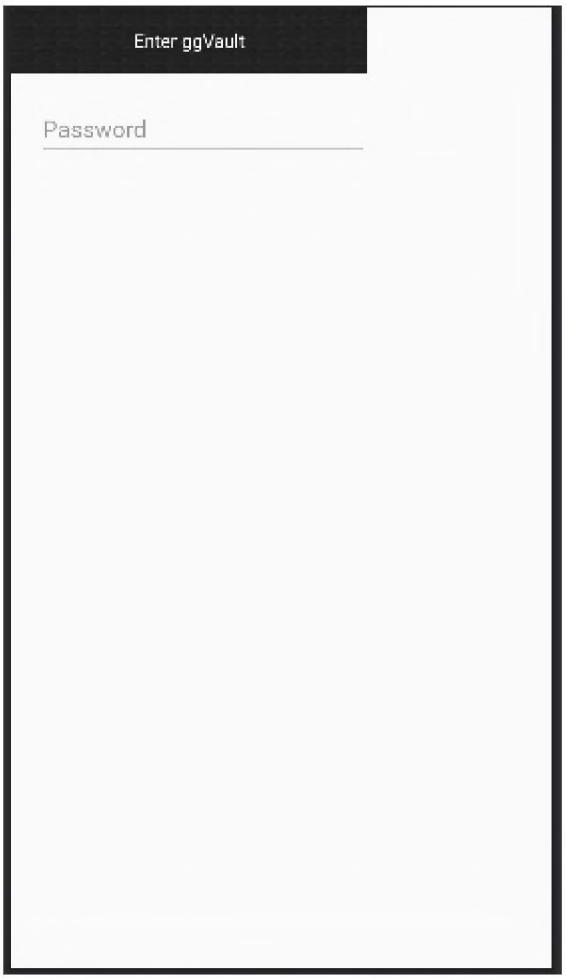

Рисунок 2.4 — Вхід в сховище

```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:gravity="center"
        android:background="@drawable/action_bar"
        android:textColor="#ffffff"
        android:text="Enter ggVault"/>

    <EditText
        android:id="@+id/password"
        android:inputType="textPassword"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:maxLength="25"
        android:ems="12"
        android:layout_marginBottom="20dp"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="20dp"
        style="@android:style/Widget.Material.EditText"
        android:fontFamily="sans-serif"
        android:hint="Password"
        tools:targetApi="lollipop" />
</LinearLayout>
```

Рисунок 2.5 — Код активності входу

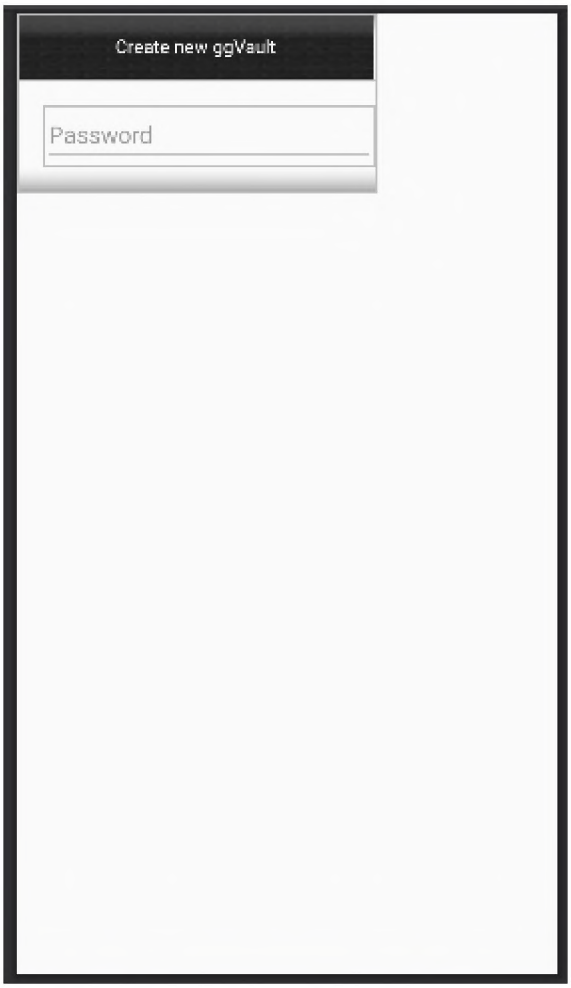

Рисунок 2.6 — Створення нового сховища

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:gravity="center"
    android:background="@drawable/action_bar"
    android:textColor="#ffffff"
    android:text="Create new ggVault"/>

<EditText
    android:id="@+id/create_password"
    android:inputType="textPassword"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:maxLength="25"
    android:ems="12"
    android:layout_marginBottom="20dp"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="20dp"
    style="@android:style/Widget.Material.EditText"
    android:fontFamily="sans-serif"
    android:hint="Password"
    tools:targetApi="lollipop" />
```

Рисунок 2.7 — Код сторінки створення сховища

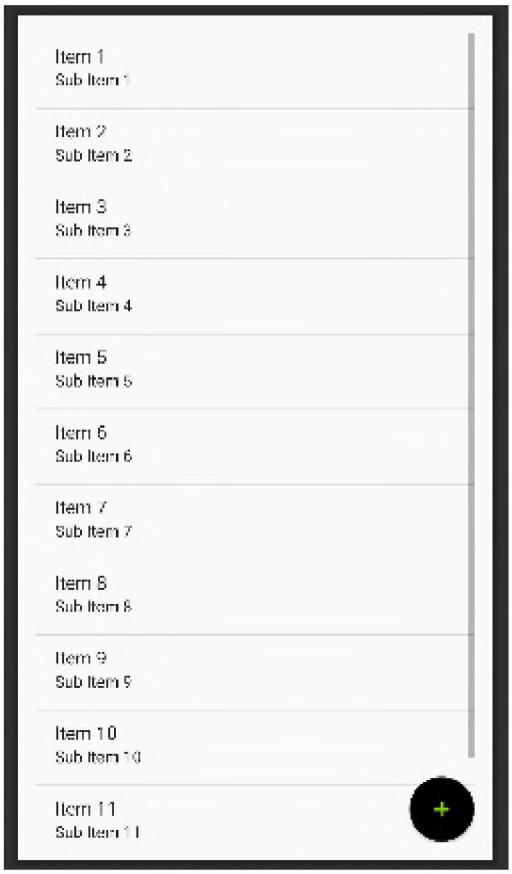

Рисунок 2.8 — Сторінка файлів

```
<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_gravity="end|bottom"
    android:src ="@android:drawable/ic_input_add"
    app:elevation="2dp"
    app:backgroundTint ="#000000"/>

<ListView
    android:id="@+id/file_list"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
</ListView>
```

Рисунок 2.9 — Код сторінки файлів

РОЗДІЛ 3
ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ

В даному розділі дипломної роботи проводиться економічне обґрунтування доцільності розробки програмного забезпечення. Зокрема, здійснюється розрахунок витрат на розробку програмного забезпечення, експлуатаційних витрат, ціни споживання проектного рішення. В заключній частині визначаються показники економічної ефективності нового програмного продукту, обґрунтовуються відповідні висновки.

Розроблене програмне забезпечення призначене для шифрування інформації методами стеганографії.

3.1 Розрахунок витрат на розробку програмного забезпечення

Витрати на розробку і впровадження програмних засобів (К) включають:

$$К = К_1+К_2 \quad (3.1)$$

де $К_1$ - витрати на розробку програмних засобів, грн.

$К_2$ - витрати на відлагодження і дослідну експлуатацію програми рішення задачі на ЕОМ, грн.

- Витрати на розробку програмних засобів включають:
- витрати на оплату праці розробників;
- витрати на відрахування у спеціальні державні фонди (Вф,);
- витрати на куповані вироби (Кв);
- витрати на придбання спецобладнання для експериментальних робіт (Об);
- накладні витрати (Н);
- інші витрати (Ів).

Витрати на оплату праці розробників проекту визначаються за формулою:

$$З = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} t_{ij} C_{ij} \qquad (3.2)$$

де nij - чисельність розробників і-ої спеціальності j-го тарифного розряду, які приймають участь в проектуванні, чол.;

t_{ij} - час, який затрачений на розробку проекту співробітника і-ої спеціальності j-го тарифного розряду, днів;

C_{ij} - денна заробітна плата і-ої спеціальності j-го тарифного розряду, грн.;

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{p} \qquad (433)$$

де C_{ij} - основна місячна заробітна плата розробника і-ої спеціальності j-го тарифного розряду, грн.;

h - коефіцієнт, що визначає розмір додаткової заробітної плати;

p - середня кількість робочих днів у місяці (21).

Таблиця 3.1.

Вихідні дані для розрахунку витрат на оплату праці

№	Посада виконавців	Місячний оклад, грн.	Погодинна ставка, грн./година
1	Керівник диплому,	8500	52,7
2	Консультант з економіки	9000	55,9
3	Студент	0	0

Таблиця 3.2.

Розрахунок витрат на оплату праці

№	Спеціальність розробника	Час розробки,	Погодинна заробітна	Витрати на розробку, грн.

1	Керівник диплому, асистент	18.5	52,7	974,95
2	Консультант з економіки	0.5	55,9	25,45
3	Студент	0	0	0
	Разом			1000,4

Величину відрахувань у спеціальні державні фонди визначають у процентному співвідношенні від суми основної та додаткової заробітної плати. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 22% від суми заробітної плати:

$$Вф=22: 100*З \quad (3.4)$$

$$Вф=0,22*1000,4=220 \text{ грн.}$$

Таблиця 3.3.

Розрахунок витрат на куповані вироби

№ п/п	Найменування купованих виробів	Одиниця виміру	Ціна за одиницю	Кількість купованих	Сума, грн.
1	Папір (формат А4)	500 листів	100,00	1	100,00
2	Зошит	Шт.	8,70	1	8,70
3	Диск (CD-RW)	Шт.	10,50	2	21,00
Всього	129,70 грн				

При розробці даного програмного забезпечення спеціальне обладнання не використовувалось, тому витрати на спеціальне обладнання відсутні.

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими процентами до витрат на оплату праці:

$$H = \frac{30}{100} З \quad (3.5)$$

$$Н = 0,3*1000,4=300,12 \text{ грн.}$$

Інші витрати відображають видатки, які не враховані в інших статтях витрат. Вони розраховуються за встановленими процентами до витрат на оплату праці:

$$I_в = 10:100*З \quad (3.6)$$

Ів=0,1*768,35=100,04 грн.

Витрати на розробку програмного забезпечення розраховуються за формулою:

$$К=З + Вф + Кв + Об + Н + Ів \quad (3.7)$$

К =1000,4+200+129,70+0+300,12+100,04=1730,26 грн.

Витрати на відлагодження і дослідну експлуатацію програмного забезпечення визначаються за формулою:

$$K_2=S_{Мг}*t_{Від} \quad (3.8)$$

де $S_{Мг}$ - вартість однієї машино-години роботи конкретного типу ЕОМ, грн./год.;

$t_{Від}$ - машинний час, витрачений на відлагодження і дослідну експлуатацію програмних засобів, год.

Загальна кількість днів роботи на ЕОМ рівна 60 днів. Середній щоденний час роботи на ЕОМ - 6 год., тому:

$$t_{від}=60*6=360 \text{ год.}$$

За занальними даними для ЕОМ типу IBM PC/AT $S_{Мг}$ = 5 грн.

Отже:

$$K_2= 5,0*360 = 1800 \text{ грн.}$$

Таблиця 3.4.

Кошторис витрат на розробку програмного забезпечення

№	Найменування елементів витрат	Сума витрат, грн.
1	Витрати на оплату праці	1000,04
2	Відрахування у спеціальні державні фонди	200
3	Витрати на куповані вироби	129,7
4	Накладні витрати	300,12

5	Інші витрати	100,04
6	Витрати на відлагодження і дослідну експлуатацію програмного забезпечення	1800
	Всього	3529,9

3.2 Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблюваного програмного продукту слід порівняти його з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість машино-годин роботи ЕОМ (за час дії програми):

$$E_{\text{П}} = E_{1\text{П}} + E_{2\text{П}} \tag{3.12}$$

де $E_{\text{п}}$ - одноразові експлуатаційні витрати на проектне рішення (аналог), грн.;

$E_{1\text{п}}$ - вартість підготовки даних для експлуатації проектного рішення (аналогу), грн.;

$E_{2\text{п}}$ - вартість машино-годин роботи ЕОМ для виконання проектного рішення (аналогу), грн.

Річні експлуатаційні витрати $B_{\text{еп}}$ визначаються за формулою:

$$B_{E\text{П}} = E_{\text{П}} * N_{\text{П}} \tag{3.13}$$

де $N_{\text{п}}$ - періодичність експлуатації проектного рішення (аналогу), раз/рік.

Вартість підготовки даних для роботи на ЕОМ визначається за формулою:

$$E_{1\text{П}} = \sum_{l=1}^{L} n_1 t_1 c_1 \tag{3.14}$$

де 1 - номери категорій персоналу, який приймає участь у підготовці даних (1=1,2,...L);

n_1, - чисельність співробітників 1-ої категорії, чол.;

t_1, - трудоємність роботи співробітників 1-ої категорії по підготовці даних, год.;

c_1 — середнього динна ставка співробітника 1-ої категорії з врахуванням додаткової заробітної плати та відрахувань у спеціальні державні фонди, грн./год.

$$c_1 = \frac{c_1^0(1+b)}{m} \quad (3.15)$$

де c_1^0 - основна місячна заробітна плата працівника 1-ої категорії, грн.;

b - коефіцієнт, який враховує додаткову заробітну плату і відрахування у спеціальні державні фонди;

m - кількість робочих годин у місяці, год.

Для роботи з даними як для проектного рішення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає: с° = 6700 грн. Тоді, враховуючи додаткову заробітну плату:

$$c_1 = \frac{6700(1+0.57)}{162} = 64{,}9 \text{ грн/год}$$

Трудомісткість працівника по підготовці даних для проектного рішення складає 1 год., для аналога 1,5 год.

Таблиця 3.7.

Розрахунок витрат на підготовку даних та реалізацію проектного рішення на ЕОМ

№	Час роботи співробітників, год.	Середньогодинна заробітна плата,	Витрати , грн.
	Проектне рішення		
1	1	64,9	64,9
	Аналог		
1	1,5	64,9	97,4

Витрати на експлуатацію ЕОМ визначається за формулою:

$$E_{2П} = t * S_{МГ} \quad (3.16)$$

де t - витрати машинного часу для реалізації проектного рішення (аналогу), год.;

$S_{МГ}$ - вартість однієї машино-години роботи ЕОМ, грн./год.

$E_{2п} = 1 * 5,0 = 5,0$ грн.

$E_{2а} = 1,5 * 5,0 = 7,5$ грн.

Еп = 64,9 + 5,0 = 69,9 грн.

Еа = 97,4 + 7,5 = 104,9 грн.

$В_{еп} = 64,9 * 252 = 16354,8$ грн.

$В_{еа} = 97,4 * 252 = 24544,8$ грн.

3.3 Розрахунок ціни споживання проектного рішення

Ціна споживання - це витрати на придбання і експлуатацію проектного рішення за весь строк його служби:

$$Цс_{п}=Ц_{п}+В_{епрv} \quad (3.17)$$

де $Ц_{п}$ - ціна придбання проектного рішення, грн.:

$$Ц_{П} = K(1+\frac{П_{Р}}{100}) + K_{0} + K_{к} \quad (3.18)$$

де Пр - норматив рентабельності;

$К_{о}$ - витрати на прив'язку та освоєння проектного рішення на конкретному об'єкті, грн.;

$К_{к}$ - витрати на доукомплектування технічних засобів на об'єкті, грн.;

$$Ц_{п}=3527,38*(1+0,3) = 4585,6 \text{ грн.}$$

$В_{епрv}$ - теперішня вартість витрат на експлуатацію проектного рішення (за весь час його експлуатації), грн.:

$$B_{enpv} = \sum_{t=0}^{T} \frac{B_{еП}}{(1+R)^{t}} \quad (3.19)$$

де $В_{еп}$ - річні експлуатаційні витрати, грн.;

Т - строк служби проектного рішення, років;

R - річна ставка проценту банківського

$$B_{enpv} = \sum_{t=1}^{5} \frac{16354,8}{(1+0.16)^{t}} = 55550.4 \text{ грн.}$$

$$B_{enpv} = \sum_{t=1}^{5} \frac{24544,8}{(1+0.16)^{t}} = 80366.8 \text{ грн.}$$

Вартість аналога становить 5000 грн., тому отримуємо:

$$Ц_{сп} = 4585,6 + 55550,4 = 60136 \text{ грн.}$$

$$Ц_{са} = 5000 + 80366,8 = 85366.8 \text{ грн.}$$

3.4 Визначення показників економічної ефективності

Економічний ефект в сфері проектування рішення:

$$Е_{пр}=Ц_а-Ц_п \quad (3.21)$$

$$Е_{пр} = 5000,0 – 4585,6 = 414,4 \text{ грн.}$$

Річний економічний ефект в сфері експлуатації:

$$Е_{кс}=В_{еа}-В_{еп}$$

$$Е_{кс} = 24544,8 - 16354,8 = 8190 \text{ грн.}$$

Додатковий економічний ефект у сфері експлуатації:

$$\Delta E_{екс} = \sum_{t=1}^{T} E_{екс}(1+R)^{T-t}$$

$$\Delta E_{екс} = \sum_{t=1}^{5} 8190 * (1+0,16)^{5-t} = 56323.7 \text{ грн.}$$

Сумарний ефект складає:

$$E = E_{пр} + \Delta E_{екс} = 414,4 + 56323.7 = 56738,1 \text{ грн}$$

Таблиця 3.8.

Показники економічної ефективності проектного рішення

№	Найменування	Одиниці вимірювання	Значення показників	
			Базовий варіант	Новий варіант
1	Капітальні вкладення	Грн.	-	3527,38
2	Ціна придбання	Грн.	5000,0	4585,6
3	Річні експлуатаційні витрати	Грн.	5922,0	3956,4
4	Ціна споживання	Грн.	24544,8	16354,8
5	Економічний ефект в сфері проектування	Грн.	-	414,4
6	Річний економічний ефект в сфері експлуатації	Грн.	-	8190
7	Додатковий ефект в сфері експлуатації	Грн.	-	56738,1
8	Сумарний ефект	Грн.	57152,5	

3.5 Висновки

В даному розділі проведено розрахунок витрат на розробку проектного рішення. Здійснено порівняння з існуючим аналогом, і цим показано, що дане проектне рішення має переваги в порівнянні з аналогами, зокрема: надійність, простота використання, гнучкість, зручність. Згідно проведеного економічного обґрунтування дане проектне рішення є конкурентноздатним. Крім того, отримано додатній економічний ефект у розмірі 56738,1грн. і тому розробка і впровадження цього проектного рішення є економічно доцільними.

ВИСНОВКИ

Мета роботи полягає в створенні власної програмної реалізації системи підвищення безпеки робочих процесів з використанням ОС Android на базі виявлення несанкціонованого доступу до критичних ресурсів.

Для досягнення поставленої мети було виконано наступні завдання:

- провести аналіз основних понять предметної області;
- провести огляд мови програмування;
- провести огляд середовища розробки;
- проаналізувати варіанти діяльності;
- спроектувати пакетну будову системи;
- спроектувати внутрішню будову системи;
- розробити графічний інтерфейс користувача;
- провести економічний аналіз доцільності розробки.

Завдяки чіткому виконанню даних завдань, в результаті було отримано повнофункціональний додаток підвищення безпеки робочих процесів з використанням ОС Android на базі виявлення несанкціонованого доступу до критичних ресурсів, який повністю виконує закладений в нього функціонал.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Binstock, Andrew (May 20, 2015). "Java's 20 Years of Innovation". Forbes. Archived from the original on March 14, 2016. Retrieved March 18, 2016.
2. https://www.oracle.com/java/technologies/downloads/#JDK17.
3. Barbara Liskov with John Guttag (2000). Program Development in Java - Abstraction, Specification, and Object-Oriented Design. USA, Addison Wesley. ISBN 9780201657685.
4. Chaudhary, Harry H. (July 28, 2014). "Cracking The Java Programming Interview :: 2000+ Java Interview Que/Ans". Retrieved May 29, 2016.
5. Java 5.0 added several new language features (the enhanced for loop, autoboxing, varargs and annotations), after they were introduced in the similar (and competing) C# language. [1] Archived March 19, 2011, at the Wayback Machine [2] Archived January 7, 2006, at the Wayback Machine
6. Gosling, James; McGilton, Henry (May 1996). "The Java Language Environment". Archived from the original on May 6, 2014. Retrieved May 6, 2014.
7. Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad. "The Java Language Specification, 2nd Edition". Archived from the original on August 5, 2011. Retrieved February 8, 2008.
8. "The A-Z of Programming Languages: Modula-3". Computerworld.com.au. Archived from the original on January 5, 2009. Retrieved June 9, 2010.
9. Niklaus Wirth stated on a number of public occasions, e.g. in a lecture at the Polytechnic Museum, Moscow in September 2005 (several independent first-hand accounts in Russian exist, e.g. one with an audio recording: Filippova, Elena (September 22, 2005). "Niklaus Wirth's lecture at the Polytechnic Museum in Moscow". Archived from the original on December 1, 2020. Retrieved November 20, 2011.), that the Sun Java design team licensed the Oberon compiler sources a number of years prior to the release of Java and examined it: a (relative) compactness, type safety, garbage

collection, no multiple inheritance for classes – all these key overall design features are shared by Java and Oberon.

10. Patrick Naughton cites Objective-C as a strong influence on the design of the Java programming language, stating that notable direct derivatives include Java interfaces (derived from Objective-C's protocol) and primitive wrapper classes. [3] Archived July 13, 2011, at the Wayback Machine
11. TechMetrix Research (1999). "History of Java" (PDF). Java Application Servers Report. Archived from the original (PDF) on December 29, 2010. The project went ahead under the name green and the language was based on an old model of UCSD Pascal, which makes it possible to generate interpretive code.
12. "A Conversation with James Gosling – ACM Queue". Queue.acm.org. August 31, 2004. Archived from the original on July 16, 2015. Retrieved June 9, 2010.
13. In the summer of 1996, Sun was designing the precursor to what is now the event model of the AWT and the JavaBeans component architecture. Borland contributed greatly to this process. We looked very carefully at Delphi Object Pascal and built a working prototype of bound method references in order to understand their interaction with the Java programming language and its APIs.White Paper About Microsoft's Delegates
14. "Chapel spec (Acknowledgements)" (PDF). Cray Inc. October 1, 2015. Archived (PDF) from the original on February 5, 2016. Retrieved January 14, 2016.
15. "Gambas Documentation Introduction". Gambas Website. Archived from the original on October 9, 2017. Retrieved October 9, 2017.
16. "Facebook Q&A: Hack brings static typing to PHP world". InfoWorld. March 26, 2014. Archived from the original on February 13, 2015. Retrieved January 11, 2015.

17."Write once, run anywhere?". Computer Weekly. May 2, 2002. Archived from the original on August 13, 2021. Retrieved July 27, 2009.

18."1.2 Design Goals of the Java™ Programming Language". Oracle. January 1, 1999. Archived from the original on January 23, 2013. Retrieved January 14, 2013.

19.McMillan, Robert (August 1, 2013). "Is Java Losing Its Mojo?". wired.com. Archived from the original on February 15, 2017. Retrieved March 8, 2017. Java is on the wane, at least according to one outfit that keeps on eye on the ever-changing world of computer programming languages. For more than a decade, it has dominated the TIOBE Programming Community Index, and is back on top – a snapshot of software developer enthusiasm that looks at things like internet search results to measure how much buzz different languages have. But lately, Java has been slipping.

20.Chan, Rosalie (January 22, 2019). "The 10 most popular programming languages, according to the 'Facebook for programmers'". Business Insider. Archived from the original on June 29, 2019. Retrieved June 29, 2019.

21."JavaOne 2013 Review: Java Takes on the Internet of Things". www.oracle.com. Archived from the original on April 19, 2016. Retrieved June 19, 2016. Alt URL

22."Why should I uninstall older versions of Java from my system?". Oracle. Archived from the original on February 12, 2018. Retrieved September 24, 2021.

23.Byous, Jon (c. 1998). "Java technology: The early years". Sun Developer Network. Sun Microsystems. Archived from the original on April 20, 2005. Retrieved April 22, 2005.

24.Object-oriented programming "The History of Java Technology". Sun Developer Network. c. 1995. Archived from the original on February 10, 2010. Retrieved April 30, 2010.

25. Murphy, Kieron (October 4, 1996). "So why did they decide to call it Java?". JavaWorld. Archived from the original on July 13, 2020. Retrieved 2020-07-13.
26. Kabutz, Heinz; Once Upon an Oak Archived April 13, 2007, at the Wayback Machine. Artima. Retrieved April 29, 2007.
27. "JAVASOFT SHIPS JAVA 1.0". Archived from the original on March 10, 2007. Retrieved May 13, 2018.
28. Object-oriented Programming with Java: Essentials and Applications. Tata McGraw-Hill Education. p. 34.
29. "JSG – Java Study Group". open-std.org. Archived from the original on August 25, 2006. Retrieved August 2, 2006.
30. "Why Java™ Was – Not – Standardized Twice" (PDF). Archived (PDF) from the original on January 13, 2014. Retrieved June 3, 2018.
31. "What is ECMA—and why Microsoft cares". ZDNet. Archived from the original on May 6, 2014. Retrieved May 6, 2014.
32. "Java Community Process website". Jcp.org. May 24, 2010. Archived from the original on August 8, 2006. Retrieved June 9, 2010.
33. "JAVAONE: Sun – The bulk of Java is open sourced". GrnLight.net. Archived from the original on May 27, 2014. Retrieved May 26, 2014.
34. "Sun's Evolving Role as Java Evangelist". O'Reilly Media. Archived from the original on September 15, 2010. Retrieved August 2, 2009.
35. "Oracle and Java". oracle.com. Oracle Corporation. Archived from the original on January 31, 2010. Retrieved August 23, 2010. Oracle has been a leading and substantive supporter of Java since its emergence in 1995 and takes on the new role as steward of Java technology with a relentless commitment to fostering a community of participation and transparency.
36. Gosling, James (April 9, 2010). "Time to move on..." On a New Road. Archived from the original on November 5, 2010. Retrieved November 16, 2011.

37. Topic, Dalibor. "Moving to a Plugin-Free Web". Archived from the original on March 16, 2016. Retrieved March 15, 2016.
38. "Learn About Java Technology". Oracle. Archived from the original on November 24, 2011. Retrieved November 21, 2011.
39. "Oracle Java SE Support Roadmap". Oracle. September 13, 2021. Retrieved September 18, 2021.
40. "JAVASOFT SHIPS JAVA 1.0". sun.com. Archived from the original on March 10, 2007. Retrieved February 5, 2008.
41. Chander, Sharat. "Introducing Java SE 11". oracle.com. Archived from the original on September 26, 2018. Retrieved September 26, 2018.
42. "The Arrival of Java 15!". Oracle. September 15, 2020. Archived from the original on September 16, 2020. Retrieved September 15, 2020.
43. "Java Card Overview". Oracle Technology Network. Oracle. Archived from the original on January 7, 2015. Retrieved December 18, 2014.
44. "Java Platform, Micro Edition (Java ME)". Oracle Technology Network. Oracle. Archived from the original on January 4, 2015. Retrieved December 18, 2014.
45. "Java SE". Oracle Technology Network. Oracle. Archived from the original on December 24, 2014. Retrieved December 18, 2014.
46. "Java Platform, Enterprise Edition (Java EE)". Oracle Technology Network. Oracle. Archived from the original on December 17, 2014. Retrieved December 18, 2014.
47. "Deep Dive Into the New Java JIT Compiler - Graal | Baeldung". www.baeldung.com. August 6, 2021. Retrieved October 13, 2021.
48. "Is the JVM (Java Virtual Machine) platform dependent or platform independent? What is the advantage of using the JVM, and having Java be a translated language?". Programmer Interview. Archived from the original on January 19, 2015. Retrieved January 19, 2015.
49. Jelovic, Dejan. "Why Java will always be slower than C++". Archived from the original on February 11, 2008. Retrieved February 15, 2008.

50. Google. "Loop Recognition in C++/Java/Go/Scala" (PDF). Archived (PDF) from the original on November 16, 2011. Retrieved July 12, 2012. {{cite web}}: |last= has generic name (help)
51. "Symantec's Just-In-Time Java Compiler To Be Integrated into Sun JDK 1.1". Archived from the original on June 28, 2010. Retrieved August 1, 2009.
52. Salcic, Zoran; Park, Heejong; Teich, Jürgen; Malik, Avinash; Nadeem, Muhammad (July 22, 2017). "Noc-HMP: A Heterogeneous Multicore Processor for Embedded Systems Designed in SystemJ". ACM Transactions on Design Automation of Electronic Systems. 22 (4): 73. doi:10.1145/3073416. ISSN 1084-4309. S2CID 11150290.
53. "NullPointerException". Oracle. Archived from the original on May 6, 2014. Retrieved May 6, 2014.
54. "Exceptions in Java". Artima.com. Archived from the original on January 21, 2009. Retrieved August 10, 2010.
55. "Java HotSpot™ Virtual Machine Performance Enhancements". Oracle.com. Archived from the original on May 29, 2017. Retrieved April 26, 2017.
56. "Operator Overloading (C# vs Java)". C# for Java Developers. Microsoft. Archived from the original on January 7, 2015. Retrieved December 10, 2014.
57. "Multiple Inheritance of State, Implementation, and Type". The Java™ Tutorials. Oracle. Archived from the original on November 9, 2014. Retrieved December 10, 2014.
58. "Lesson: A Closer Look at the Hello World Application". The Java™ Tutorials > Getting Started. Oracle Corporation. Archived from the original on March 17, 2011. Retrieved April 14, 2011.
59. "Deprecated APIs, Features, and Options". Oracle. Archived from the original on June 19, 2019. Retrieved May 31, 2019.
60. "Applet (Java Platform SE 7)". Docs. Oracle. Archived from the original on August 2, 2020. Retrieved May 1, 2020.

61. "What Is a JSP Page? - The Java EE 5 Tutorial". docs.oracle.com. Archived from the original on August 2, 2020. Retrieved May 1, 2020.
62. "Trail: Creating a GUI With JFC/Swing (The Java™ Tutorials)". docs.oracle.com. Archived from the original on April 29, 2020. Retrieved May 1, 2020.
63. "Removed from JDK 11, JavaFX 11 arrives as a standalone module". InfoWorld. Archived from the original on October 14, 2020. Retrieved October 13, 2020.
64. "Getting Started with JavaFX: Hello World, JavaFX Style". JavaFX 2 Tutorials and Documentation. Oracle. Archived from the original on August 2, 2020. Retrieved May 1, 2020.
65. "Java and Scala's Type Systems are Unsound" (PDF). Archived (PDF) from the original on November 28, 2016. Retrieved February 20, 2017.
66. Arnold, Ken. "Generics Considered Harmful". java.net. Archived from the original on October 10, 2007. Retrieved September 10, 2015. More comments to the original article available at earlier archive snapshots.
67. Jelovic, Dejan. "Why Java Will Always Be Slower than C++". www.jelovic.com. Archived from the original on February 11, 2008. Retrieved October 17, 2012.
68. Owens, Sean R. "Java and unsigned int, unsigned short, unsigned byte, unsigned long, etc. (Or rather, the lack thereof)". Archived from the original on February 20, 2009. Retrieved July 4, 2011.
69. Kahan, William. "How Java's Floating-Point Hurts Everyone Everywhere" (PDF). Electrical Engineering & Computer Science, University of California at Berkeley. Archived (PDF) from the original on September 5, 2012. Retrieved June 4, 2011.
70. "Have you checked the Java?". Archived from the original on September 21, 2012. Retrieved December 23, 2011.

71. Cadenhead, Rogers (November 20, 2017), Understanding How Java Programs Work, archived from the original on August 13, 2021, retrieved March 26, 2019
72. Woolf, Nicky (May 26, 2016). "Google wins six-year legal battle with Oracle over Android code copyright". The Guardian. ISSN 0261-3077. Archived from the original on March 26, 2019. Retrieved March 26, 2019.
73. "Collections Framework Overview". Java Documentation. Oracle. Archived from the original on December 31, 2014. Retrieved December 18, 2014.
74. "Java™ Security Overview". Java Documentation. Oracle. Archived from the original on January 3, 2015. Retrieved December 18, 2014.
75. "Trail: Internationalization". The Java™ Tutorials. Oracle. Archived from the original on December 31, 2014. Retrieved December 18, 2014.
76. "How to Write Doc Comments for the Javadoc Tool". Oracle Technology Network. Oracle. Archived from the original on December 18, 2014. Retrieved December 18, 2014.
77. Niccolai, James (January 24, 2001). "Sun, Microsoft settle Java lawsuit". JavaWorld. IDG News Service. Archived from the original on July 14, 2020. Retrieved 2020-07-13.
78. van Gurp, Jilles (November 13, 2007). "Google Android: Initial Impressions and Criticism". Javalobby. Archived from the original on August 28, 2008. Retrieved March 7, 2009. Frankly, I don't understand why Google intends to ignore the vast amount of existing implementation out there. It seems like a bad case of "not invented here" to me. Ultimately, this will slow adoption. There are already too many Java platforms for the mobile world and this is yet another one
79. https://android-developers.googleblog.com/2022/05/android-studio-chipmunk.html.
80. ^ "Android Studio Dolphin Canary 7 now available". Android Studio Release Updates. Retrieved March 30, 2022.

81.^ "Android Studio Bumblebee Canary 9 available". Android Studio Release Updates. Retrieved August 23, 2021.

82.^ Jump up to:a b "Download Options". developer.android.com. Retrieved November 16, 2021.

83.^ "Terms and Conditions". developer.android.com. Retrieved April 24, 2017.

84.^ "Build Overview". android.com.

85.^ "Building Android Studio". android.com.

86.^ "Download Android Studio and SDK tools". Android Developers.

87.^ Ducrohet, Xavier; Norbye, Tor; Chou, Katherine (May 15, 2013). "Android Studio: An IDE built for Android". Android Developers Blog. Retrieved May 16, 2013.

88.^ "Getting Started with Android Studio". Android Developers. Retrieved May 14, 2013.

89.^ Haslam, Oliver (May 16, 2013). "Download Android Studio IDE For Windows, OS X And Linux". Redmond Pie. Retrieved May 16, 2013.

90.^ "Download Android Studio". Android Developers. Retrieved June 13, 2015.

91.^ "Google Launches Android Studio And New Features For Developer Console, Including Beta Releases And Staged Rollout". VentureBeat. December 8, 2014. Retrieved December 9, 2014.

92.^ "Kotlin is now Google's preferred language for Android app development". TechCrunch. Retrieved May 8, 2019.

93.^ Sinicki, Adam (August 10, 2019). "I want to develop Android Apps — What languages and program I should use & learn?". Android Authority. Retrieved September 12, 2019.

94.^ Honig, Zach (May 15, 2013). "Google intros Android Studio, an IDE for building apps". Engadget. AOL. Retrieved May 16, 2013.

95.^ Dobie, Alex (May 15, 2013). "Android Studio unveiled at Google I/O keynote". Android Central. Mobile Nations. Retrieved May 16, 2013.

96.^ Olanoff, Drew (May 15, 2013). "Google Launches Android Studio And New Features For Developer Console, Including Beta Releases And Staged Rollout". TechCrunch. AOL. Retrieved May 16, 2013.

97.^ "Android Studio BETA". Google. May 15, 2013. Retrieved August 15, 2014.

98.^ Google Go language IDE built using the IntelliJ Platform: go-lang-plugin-org/go-lang-idea-plugin, Go Language support for IDEA based IDEs, February 23, 2019, retrieved February 23, 2019, Supported IDEs [..] Android Studio 1.2.1+

99.^ "Get Started with Kotlin on Android | Android Developers". developer.android.com. Retrieved October 25, 2017.

100. ^ "Use Java 8 language features | Android Developers". developer.android.com. Retrieved October 25, 2017.

101. ^ "android-retroflow: Backport of Java 9 (JEP 266) reactive-streams Flow and SubmissionPublisher API for Android Studio 3.0 desugar toolchain, forked from [..]". retrostreams. October 22, 2017. Retrieved October 25, 2017.

102. ^ "Android's Java 9, 10, 11, and 12 Support". Jake Wharton. November 27, 2018. Retrieved February 23, 2019. Hopefully by the time Java 12 is actually released D8 will have implemented desugaring for Java 11's nestmates. Otherwise the pain of being stuck on Java 10 will go up quite a bit!

103. ^ "Android Studio Release Notes". Android Developers Official Website. August 2019. Retrieved September 2, 2019.

104. ^ "Google Android Studio 3.4 Now Available, Here's What's New - Appetiser". April 25, 2019.

105. ^ "Android Studio 4.1".

106. ^ "Android Studio 4.2 available in the Stable channel".

107. ^ "Android Studio Arctic Fox available in the Stable channel".

108. ^ "Android Studio Bumblebee (2021.1.1) Stable".

109. ^ "Android Studio Chipmunk". Android Developers Blog. Retrieved May 12, 2022.

110. ^ Canary 1 https://androidstudio.googleblog.com/2022/01/android-studio-dolphin-canary-1-now.html Canary 1. {{cite web}}: Check |url= value (help); Missing or empty |title= (help)

111. ^ "Download Android Studio and SDK tools". Android Developers. Retrieved March 7, 2022.

112. ^ "Download Android Studio and SDK tools". Android Developers. Retrieved March 7, 2022.

113. "Emulator release notes". Android Developers. Retrieved May 12, 2022.

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№	Формат	Найменування	Кількість листів	Примітки
Документація				
1	А4	Реферат	2	
2	А4	Перелік умовних скорочень	1	
3	А4	Зміст	1	
4	А4	Вступ	1	
5	А4	Аналіз предметної області	25	
6	А4	Проектування і програмна реалізація	27	
7	А4	Економічне обгрунтування доцільності розробки	13	
8	А4	Висновки	1	

9	А4	Перелік використаних джерел	10	
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
13	А4	Додаток Г	1	

ДОДАТОК Б. Перелік документів на оптичному носії

1.Мінзар_МА_125_18_2_ПЗ.docx

2.Диплом_Мінзар_МА_125_18_2.pdf

3.Презентація_дипломна робота_Мінзар_МА_125_18_2.pptx

4.Програмні коди.zip

ДОДАТОК В. Відгук керівника економічного розділу

Керівник розділу _____ _____

(підпис) (Прізвище, ініціали)

Додаток Г. Відгук керівника кваліфікаційної роботи

В І Д Г У К

на кваліфікаційну роботу студента групи 125-18-2 Мінзар М.А. на тему: «Система підвищення безпеки робочих процесів з використанням ОС Android на базі виявлення несанкціонованого доступу до критичних ресурсів»

Пояснювальна записка складається зі вступу, трьох розділів і висновків, розташованих на 80 сторінках.

Мета роботи є актуальною, оскільки вона спрямована на підвищення безпеки робочих процесів при використанні портативних пристроїв під управлінням ОС Android, що є гострою проблемою сьогодення.

При виконанні роботи автор продемонстрував добрий рівень теоретичних знань і практичних навичок. На основі аналізу проблем різноманітних версій ОС Android та можливостей мови програмування Java, завдяки цьому було сформульовано найголовнішу проблему під час використання ПО, яка була вирішена завдяки програмній реалізації.

Практична цінність роботи полягає у тому, що віднайдений програмний засіб є повноцінною можливістю уникнення викрадення критичної інформації для будь яких структур.

Рівень запозичень у кваліфікаційній роботі не перевищує вимог «Положення про систему виявлення та запобігання плагіату».

В цілому робота задовольняє усім вимогам, а її автор Мінзар М.А. заслуговує на оцінку « » та присвоєння кваліфікації «Бакалавр з кібербезпеки» за спеціальністю 125 Кібербезпека.

Керівник роботи,

ст.викладач **Саксонов Г.М**