Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____д.т.н., проф. С. І. Шматков
«___»_____2024 р

Пояснювальна записка
до кваліфікаційної роботи
бакалавра

на тему: «**МОДЕЛЬ ЗАСТОСУНКУ МЕСЕНДЖЕРУ НА МОВІ ПРОГРАМУВАННЯ JAVA**»

Захищено на засіданні
Атестаційної комісії № 42
протокол № від .06.2024 р.
Оцінка_/ _____
Голова Атестаційної комісії
_____ **СКОБ Ю. О.**

Виконав:
студент 4 курсу, групи КІ–41
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 – Комп'ютерна інженерія.
КОВАЛЕНКО Іван Владиславович

Керівник: д.т.н, професор кафедри ТПС
МІРОШНИК Марина Анатолієвна

Рецензент: д.т.н., проф., професор кафедри спеціалізованих комп'ютерних систем українського державного університету залізничного транспорту
ДОЦЕНКО Сергій Ілліч

Харків - 2024

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висвновку, списку використаних джерел та чотирьох додатків. Загальних обсяг роботи складає 53 сторінок, із яких 35 сторінок основної частини з 3 малюнками та 10 найменуваннями списку використаних джерел та чотирма додатками.

Метою кваліфікаційної роботи є модель застосунку на мові програмування Java, який предствляє собою месенджер.

Об'єкт досліджень - мова програмування Java, саме для здобуття знань та відповідних новичок поза офіційними освітніми установами.

Предмет досліджень - комп'ютерна модель, спрямована на розвиток месенджерів написаних мовою Java.

Ключові слова: модель, архітектура, мова програмування Java

ABSTRACT

The explanatory note to the bachelor's thesis consists of an introduction, three chapters, a conclusion, a list of references and four appendices. The total volume of the work is 53 pages, including 35 pages of the main part with 3 figures and 10 references and four appendices.

The purpose of the qualification work is to model an application in the Java programming language, which is a messenger.

The object of research is the Java programming language, specifically for the acquisition of knowledge and relevant beginners outside of formal educational institutions.

The subject of research is a computer model aimed at developing messengers written in Java.

Keywords: model, architecture, Java programming language

ЗМІСТ

ВСТУП

У сучасному цифровому ландшафті комунікація вийшла за межі традиційних кордонів і перетворилася на сферу, де додатки для обміну миттєвими повідомленнями відіграють центральну роль в особистій, освітній та професійній взаємодії.

Повсюдне поширення смартфонів і зростаюча залежність від цифрових комунікаційних платформ вимагають розробки надійних, ефективних і зручних у використанні додатків для обміну повідомленнями. У моїй дипломній роботі досліджується розробка та реалізація типового додатку месенджера за допомогою мови програмування Java, забезпечуючи всебічний розгляд технічних та функціональних аспектів, пов'язаних з цим.

Java, відома своєю платформонезалежністю, об'єктно-орієнтованими можливостями та великими бібліотеками, забезпечує ідеальну основу для розробки масштабованих та підтримуваних програмних додатків.

В клафікаційні роботі проведено дослідження існуючих додатків для обміну повідомленнями, їх архітектури та технологій, які вони використовують. Визначено сильні та слабкі сторони існуючих рішень і надає контекстуальну основу для запропонованої моделі.

Проведено покроковій звіт про процес розробки.

В ході роботи проведено тестування та оційнка методів що використовуються для забезпечення надійності.

Метою кваліфікаційної роботи є модель застосунку на мові програмування Java, який предствляє собою месенджер.

Об’єкт досліджень - мова програмування Java, саме для здобуття знань та відповідних новичок поза офіційними освітніми установами.

Предмет досліджень - комп’ютерна модель, спрямована на розвиток месенджерів написаних мовою Java.

Вибір Java як основної мови програмування для створення додатку обґрунтований її стабільністю, надійністю та широкими можливостями для реалізації різноманітних функціоналів. Java є однією з найпопулярніших мов

програмування, що активно використовується для розробки корпоративних додатків, мобільних застосунків та систем реального часу.

Таким чином, ця робота є актуальною, оскільки вона поєднує в собі аналіз сучасних технологій обміну повідомленнями, порівняння різних мов програмування та практичну реалізацію додатку на основі найсучасніших стандартів та підходів. Результати дослідження можуть бути корисними як для розробників програмного забезпечення, так і для науковців, які займаються дослідженнями у галузі комп'ютерних наук.

РОЗДІЛ 1

АНАЛІЗ ВИМОГ ДО МЕСЕНДЖЕРІВ

1.1 Історія розвитку месенджерів

Хоча термін «миттєвий обмін повідомленнями» з'явився в 1990-х роках, він виник ще до Інтернету, з'явившись в середині 1960-х років у багатокористувацьких операційних системах, таких як Compatible Time-Sharing System (CTSS) і Multiplexed Information and Computing Service (Multics). Ці системи спочатку слугували інструментами сповіщення для таких послуг, як друк, але швидко еволюціонували, дозволивши спілкуватися між користувачами, що увійшли до одного комп'ютера. CTSS, наприклад, дозволяла обмінюватися текстовими повідомленнями між 30 користувачами.

У той самий час з'явилися перші програми для онлайн-чату, однією з перших з яких була Talkomatic в системі PLATO в 1973 році, що дозволяла п'ятьом людям спілкуватися одночасно. Під час феномену системи дошок оголошень (BBS) у 1980-х роках деякі системи включали функції чату, подібні до обміну миттєвими повідомленнями; Freelancin' Roundtable є яскравим прикладом. Першою широко доступною комерційною службою онлайн-чату був CB Simulator від CompuServe у 1980 році, створений Олександром «Сенді» Тревором у Колумбусі, штат Огайо.

З розширенням мереж розширювалися і протоколи обміну миттєвими повідомленнями, деякі з них використовували однорангові протоколи (наприклад, talk, ntalk і ytalk), а інші вимагали підключення до сервера (наприклад, talker і IRC). Служба сповіщень Zephyr, розроблена в рамках проекту MIT Project Athena у 1980-х роках, досі використовується деякими установами для надсилання повідомлень користувачам.

Ранні програми обміну миттєвими повідомленнями використовували текст у реальному часі, відображаючи символи в міру їхнього введення. До них відносяться програма командного рядка Unix «talk», популярна в 1980-х і на початку 1990-х років, і деякі програми чату BBS, такі як Celerity BBS. Сучасні

реалізації тексту в реальному часі також існують в програмах обміну миттєвими повідомленнями, таких як AOL's Real-Time IM.

Наприкінці 1980-х - на початку 1990-х років онлайн-сервіс Quantum Link для комп'ютерів Commodore 64 пропонував користувацькі повідомлення під назвою «On-Line Messages» (OLM), пізніше перейменовані на «FlashMail». Ці повідомлення з'являлися у вигляді жовтої смуги у верхній частині екрану користувача, пропонуючи варіанти відповідей. Цей примітивний графічний інтерфейс користувача (GUI) еволюціонував, коли Quantum Link став America Online (AOL) і розробив AOL Instant Messenger (AIM).

Сучасні клієнти обміну повідомленнями на основі графічного інтерфейсу з'явилися в середині 1990-х років з PowWow, ICQ та AOL Instant Messenger. CU-SeeMe, передусім аудіо/відео чат, також дозволяв обмінюватися текстовими повідомленнями. Придбання AOL компанії Mirabilis, творців ICQ, закріпило її домінування на ринку миттєвих повідомлень. Конкуруючі сервіси від Excite, MSN, Yahoo! та інших вимагали декількох клієнтів для доступу до різних мереж. У 1998 році IBM випустила IBM Lotus Sametime, використовуючи технології компаній Ubique з Хайфи та Databeam з Лексінгтона.

У 2000 році було випущено додаток і протокол Jabber з відкритим вихідним кодом (пізніше стандартизований як Extensible Messaging and Presence Protocol, XMPP), що дозволило серверам діяти як шлюзи до інших протоколів обміну миттєвими повідомленнями і зменшило потребу в декількох клієнтах. Випуск IBM Lotus Sametime 2007 року включав підтримку XMPP.

Відеодзвінки через веб-камеру також набули популярності, і такі першопрохідці, як Microsoft NetMeeting, а згодом Skype, випущений у 2003 році, принесли цю функцію ширшій аудиторії.

До 2006 року AIM домінував на ринку обміну миттєвими повідомленнями, займаючи 52% ринку, але зі зростанням конкуренції його частка зменшилася. До 2010 року миттєві веб-повідомлення занепали на користь функцій обміну повідомленнями в соціальних мережах, таких як Facebook Chat і Twitter, які пропонували миттєві повідомлення Веб 2.0.

Серверні чати також стали поширеними на сайтах знайомств, таких як OkCupid і Plenty of Fish.

Популярність миттєвих повідомлень відродилася з появою мобільних додатків, таких як BlackBerry Messenger (2005) і WhatsApp (2009). Розроблені для мобільних пристроїв, ці програми зростали разом із поширенням смартфонів з доступом до Інтернету, що призвело до того, що до 2013 року месенджери випередили SMS за обсягом повідомлень. До 2014 року месенджери мали більше користувачів, ніж соціальні мережі, і лише WhatsApp обробляв 30 мільярдів повідомлень на день до січня 2015 року, порівняно з 20 мільярдами для SMS.

У 2016 році Google запустив Allo, інтелектуальний додаток для обміну повідомленнями з використанням машинного навчання, який було закрито 12 березня 2019 року.

1.2 Аналіз існуючих месенджерів

1. WhatsApp

Функції:

-Наскрізне шифрування за замовчуванням.

-Обмін текстовими повідомленнями, голосовими та відеодзвінками, голосовими повідомленнями та файлами.

-Оновлення статусу (як історії в інших додатках).

-Групові чати до 1024 осіб.

-WhatsApp Web/Desktop для обміну повідомленнями на ПК.

-Інтеграція з бізнес-акаунтами.

Безпека:

-Надійне наскрізне шифрування (E2EE) за замовчуванням.

-Додаткові зашифровані резервні копії.

-База користувачів:

-Понад 2 мільярди активних користувачів щомісяця по всьому світу.

-Популярний в Індії, Бразилії, деяких країнах Європи та Африки.

Зручність використання:

-Простий, легкий у використанні інтерфейс.

-Часті оновлення та нові функції.

-Обмежені можливості кастомізації.

2. Telegram

Можливості:

-Хмарний обмін повідомленнями.

-Секретні чати з E2EE.

-Повідомлення, що самознищуються.

-Текстові повідомлення, голосові та відеодзвінки, голосова пошта та обмін файлами.

-Великі групові чати (до 200 000 учасників) і канали.

-Боти та сторонні інтеграції.

-Кросплатформенна підтримка десктопних і веб-клієнтів.

Безпека:

-Секретні чати з E2EE.

-Звичайні чати зашифровані, але зберігаються на серверах.

-Опціональна двоетапна перевірка.

База користувачів:

-Понад 700 мільйонів активних користувачів щомісяця.

-Зростає на Близькому Сході, в Європі та Росії.

-Зручність використання:

-Багатофункціональний, але простий.

-Легко налаштовується.

-Розширені функції можуть мати більш круту криву навчання.

3. Сигнал

Можливості:

-Наскрізне шифрування для всіх комунікацій.

-Текстові повідомлення, голосові та відеодзвінки, голосова пошта та обмін файлами.

-Зникаючі повідомлення.

-Групові чати та зашифровані групові дзвінки.

-Блокування PIN-кодом і реєстрацією.

Безпека:

-Відомий надійним E2EE та суворою політикою конфіденційності.

-Використовує протокол Signal з відкритим вихідним кодом.

База користувачів:

-Понад 50 мільйонів завантажень.

-Користується популярністю серед користувачів та активістів, які піклуються про конфіденційність.

-Зручність використання:

-Простий і мінімалістичний дизайн.

-Обмежена кастомізація та розширені функції порівняно з конкурентами.

4. Facebook Messenger

Особливості:

-Інтегрований з Facebook.

-Текстові повідомлення, голосові та відеодзвінки, голосова пошта та обмін файлами.

-Історії (подібно до Instagram Stories).

-Ігри та сторонні додатки на платформі.

-Бізнес-інтеграції для обслуговування клієнтів.

Безпека:

-Додаткове наскрізне шифрування (Secret Conversations).

-Звичайні повідомлення шифруються під час пересилання.

База користувачів:

-Понад 1,3 мільярда активних користувачів щомісяця.

-Популярний у Північній Америці, Європі та деяких частинах Азії.

-Зручність використання:

-Багатофункціональний і простий у використанні.

-Інтегрований з Facebook, що може турбувати користувачів, які піклуються про конфіденційність.

5. iMessage (Apple Messages)

Можливості:

-Наскрізне шифрування для обміну повідомленнями iMessage-to-IMessage.

-Текстові повідомлення, голосові та відеодзвінки (через FaceTime), голосова пошта та обмін файлами.

-Підтримка мультимедійних даних (Animoji, Memoji тощо).

-Інтеграція з іншими сервісами та пристроями Apple.

-Резервне SMS для користувачів, які не мають iOS.

Безпека:

-Сильний E2EE для iMessage.

-Звичайні SMS/MMS не шифруються.

База користувачів:

-Переважно використовується користувачами iOS.

-Популярний у Північній Америці та деяких частинах Європи.

Зручність використання:

-Бездоганно інтегрований з екосистемою Apple.

-Обмежена пристроями Apple.

1.3 Вимоги до сучасних месенджерів

Наступну інформацію було запозичено з сайду “teamwire.eu”

Ви можете подумати, що месенджери досягли кінця свого життєвого циклу, особливо в домашніх умовах. Зараз ці технології є стандартною частиною повсякденної цифрової комунікації. Наприклад, WhatsApp, завдяки своїй простоті та надійності, є загальновизнаним лідером у сфері обміну особистими повідомленнями.

Однак останнім часом бізнес-комунікація та її гнучкість зіткнулися зі значними проблемами. Для IT-адміністраторів ця ситуація є складною, оскільки стало зрозуміло, що багато комунікаційних інструментів не забезпечують належним чином безпеку даних та відповідність вимогам GDPR. WhatsApp, наприклад, не призначений для ділових розмов, безпечного обміну інформацією або професійного IT-нагляду.

Нові комунікаційні рішення швидко розвиваються, особливо у сфері текстових та відео-розмов онлайн. Але що відрізняє хороший додаток для бізнес-повідомлень від лідера ринку?

Нижче надані основні вимоги до сучасних месенджерів:

1. Просте розгортання та хостинг
2. Висока масштабованість та продуктивність
3. Провідне управління мобільними додатками
4. Професійне адміністрування
5. Синхронізований крос-платформний додаток
6. Провідний захист даних
7. Провідна безпека даних
8. Гарантія відповідності
9. Відкритий API
10. Інтуїтивно зрозумілий інтерфейс користувача
11. Обмін цифровим контентом
12. Посилання
13. Мережа IT-партнерів
14. Провідна підтримка та оновлення
15. Лідер за сукупною вартістю володіння (TCO)

Висновки за розділом 1

У даному розділі я дізнався загальну історію розвитку месенджерів. Дізнався з чого починалася історія месенджерів, і те що як з початку їх істрії один месенджер зміняв інший, більш технологічний та розвинений.

Провели аналіз популярних месенджерів, дізнався їх недоліки та переваги. Зрозумів які зараз вимого до сучасних месенджерів та для чого вони потрібні.

РОЗДІЛ 2
ВИБІР ТЕХНОЛОГІЙ ПРОГРАМУВАННЯ ТА УПРАВЛІННЯ БАЗАМИ

2.1 Аналіз та порівняння мов програмування

Мова Python

Синтаксис і простота вивчення:

-Python відомий своїм зрозумілим, читабельним синтаксисом, що робить її чудовим вибором для початківців.

-Він використовує відступи для визначення блоків коду, що покращує читабельність.

Продуктивність:

-Python є інтерпретованою мовою і може працювати повільніше, ніж скомпільовані мови.

-Підходить для додатків, де продуктивність не є критичною.

Варіанти використання:

-Широко використовується у веб-розробці, аналізі даних, штучному інтелекті, наукових обчисленнях, автоматизації та написанні сценаріїв.

Спільнота та екосистема:

-Python має велику, активну спільноту з великими бібліотеками та фреймворками (наприклад, Django, Flask, NumPy, Pandas, TensorFlow).

Мова Java

Синтаксис та простота вивчення:

-Java має складніший синтаксис, ніж Python, але все одно вважається відносно легкою у вивченні, особливо для тих, хто знайомий з мовами у стилі

Мова C.

-Використовує фігурні дужки {} для визначення блоків коду.

Продуктивність:

-Java - це скомпільована мова, яка, як правило, має кращу продуктивність, ніж інтерпретовані мови.

-Вона використовує компіляцію Just-In-Time (JIT) для оптимізації продуктивності.

Варіанти використання:

-Часто використовується в корпоративних додатках, розробці додатків для Android, веб-додатках і великих системах.

Спільнота та екосистема:

Java має велику спільноту та багату екосистему потужних фреймворків (наприклад, Spring, Hibernate) та інструментів (наприклад, Maven, Gradle).

Мова C++

Синтаксис і простота вивчення:

-C++ має складний синтаксис, який може бути складним для початківців.

-Вона пропонує низькорівневі можливості маніпулювання пам'яттю.

Продуктивність:

-Відома своєю високою продуктивністю та ефективністю завдяки своїй компільованій природі.

-Підходить для критичних до продуктивності додатків, таких як розробка ігор, систем реального часу та високопродуктивних обчислень.

Області застосування:

-Системне програмування, розробка ігор, вбудовані системи та програми, що потребують безпосередньої роботи з апаратним забезпеченням.

Спільнота та екосистема:

-C++ має сильну спільноту, хоча і не таку велику, як Python або Java.

-Широкі бібліотеки та фреймворки, особливо для розробки ігор (наприклад, Unreal Engine).

JavaScript

Синтаксис і простота вивчення:

-JavaScript має синтаксис, який відносно легко вивчити початківцям, особливо тим, хто знайомий з веб-розробкою.

-Це динамічна, вільно типізована мова.

Продуктивність:

-Інтерпретована мова, але сучасні движки JavaScript (наприклад, V8) забезпечують високу продуктивність.

-В основному використовується для написання сценаріїв на стороні клієнта зі швидким виконанням у браузері.

Сфери застосування:

-Веб-розробка (на стороні клієнта і все частіше на стороні сервера з Node.js), інтерактивні веб-додатки, фронтенд-розробка.

Спільнота та екосистема:

-Велика та активна спільнота з багатою екосистемою бібліотек та фреймворків (наприклад, React, Angular, Vue.js, Node.js).

Мова C#

Синтаксис і простота вивчення:

-C# має синтаксис, подібний до Java, і є відносно легкою у вивченні, особливо для тих, хто знайомий з мовами у стилі C.

-Вона розроблена з думкою про простоту та сучасні можливості.

Продуктивність:

-Скомпільована на проміжній мові (IL) і виконується середовищем виконання .NET, вона забезпечує хорошу продуктивність.

-Підходить для широкого спектру додатків, від настільних до веб та мобільних.

Варіанти використання:

-В основному використовується в корпоративних додатках, додатках для Windows, розробці ігор (з Unity) та веб-розробці (з ASP.NET).

Спільнота та екосистема:

-Активна спільнота з сильною екосистемою, особливо в стеку Microsoft.

-Широкі бібліотеки та фреймворки, особливо для корпоративної та ігрової розробки.

2.2 Безпека та відповідність стандартам.

Вступ до мов програмування

Мови програмування є основою розробки програмного забезпечення, надаючи засоби для створення різноманітних додатків, які є рушійною силою сучасного цифрового світу. Вибір мови програмування може мати значний вплив на різні аспекти проекту, включаючи його продуктивність, ремонтопридатність і безпеку. Крім того, дотримання стандартів і правил має важливе значення для забезпечення надійності та безпеки програмного забезпечення. У цьому звіті розглядаються різні мови програмування, їх вплив на безпеку та важливість дотримання стандартів.

Огляд мов програмування

Мови програмування можна умовно поділити на низькорівневі та високорівневі. Мови низького рівня, такі як асемблер і C, забезпечують тонкий контроль над апаратним забезпеченням, але вимагають детального управління пам'яттю і ресурсами. Мови високого рівня, такі як Python та Java, абстрагують більшу частину складності, дозволяючи розробникам писати більш читабельний і зручний для обслуговування код.

1. C та C++

-Сильні сторони: висока продуктивність, тонкий контроль над системними ресурсами.

-Слабкі сторони: Схильні до вразливостей, пов'язаних з пам'яттю, таких як переповнення буферів та зловживання вказівниками.

2. Java

-Сильні сторони: незалежність від платформи завдяки віртуальній машині Java (JVM), надійні стандартні бібліотеки та автоматичне керування пам'яттю (збір сміття).

-Слабкі сторони: Нижча продуктивність, ніж у C/C++, потенціал для вразливостей, якщо не керувати ними належним чином.

3. Python

-Сильні сторони: Читабельність, простота використання, велика стандартна бібліотека, динамічна типізація та швидка розробка.

-Слабкі сторони: Менша швидкість виконання, динамічна типізація може призвести до помилок під час виконання, а також іноді слабші практики безпеки через простоту використання.

4. JavaScript

-Сильні сторони: повсюдне використання у веб-розробці, можливості асинхронного програмування та розгалужена екосистема.

-Слабкі сторони: Вразливість до міжсайтових скриптів (XSS) та інших атак на стороні клієнта, непослідовна поведінка в різних середовищах.

5. Rust

-Сильні сторони: безпека пам'яті без збирача сміття, паралельність та продуктивність.

-Слабкі сторони: Більш крута крива навчання, ніж у інших мов високого рівня.

Міркування щодо безпеки

Безпека є важливим аспектом розробки програмного забезпечення. Кожна мова програмування має свій власний набір проблем безпеки та функцій для зменшення потенційних ризиків.

Управління пам'яттю

-C та C++: Ручне керування пам'яттю в C та C++ може призвести до

таких вразливостей, як переповнення буферів, якими часто користуються зловмисники. Безпечні практики кодування та інструменти, такі як AddressSanitizer, можуть допомогти зменшити ці ризики.

-Java та Python: Автоматичне керування пам'яттю (збір сміття) зменшує ризик витоку пам'яті та переповнення буфера. Однак воно не усуває всі ризики безпеки, наприклад, ті, що виникають через неправильне поводження з об'єктами.

Безпека типів

-Сильно типізовані мови: Java, Rust та C# забезпечують сувору перевірку типів, що дозволяє запобігти низці помилок та вразливостей. Це зменшує ймовірність плутанини типів і пов'язаних з нею проблем безпеки.

-Мови зі слабкою типізацією: Python та JavaScript пропонують більшу гнучкість, але можуть містити малопомітні помилки та вразливості, якщо помилки, пов'язані з типом, не контролювати ретельно.

Паралелізм

-Rust: Модель власності та гарантії паралельності Rust роблять її особливо придатною для написання безпечного паралельного коду, зменшуючи ризик виникнення гоночних станів та тупикових ситуацій.

-Java: Модель паралелізму Java, хоч і потужна, але вимагає ретельного управління, щоб уникнути таких проблем, як тупики та гонки.

Веб-безпека

-JavaScript: Як основна мова для клієнтської веб-розробки, JavaScript часто стає мішенню для зловмисників. Серед найпоширеніших вразливостей - XSS та підробка міжсайтових запитів (CSRF). Сучасні фреймворки та бібліотеки, такі як React та Angular, надають функції для зменшення цих ризиків.

-Внутрішні мови: Мови, що використовуються на стороні сервера, такі як Python, Java і Node.js (JavaScript), повинні ретельно обробляти валідацію, автентифікацію та авторизацію вводу, щоб уникнути поширених веб-уразливостей, таких як SQL-ін'єкції та CSRF.

Відповідність стандартам

Дотримання стандартів є важливим для забезпечення якості, безпеки та

інтероперабельності програмного забезпечення. Кілька організацій та органів зі стандартизації визначають настанови та вимоги до розробки програмного забезпечення.

Стандарти ISO/IEC

-ISO/IEC 27001: Цей стандарт забезпечує основу для управління інформаційною безпекою та визначає вимоги до створення, впровадження, підтримки та постійного вдосконалення системи управління інформаційною безпекою (СУІБ).

-ISO/IEC 27002: Цей стандарт містить настанови щодо стандартів інформаційної безпеки організації та практик управління інформаційною безпекою, включаючи вибір, впровадження та управління засобами контролю з урахуванням середовища (середовищ) ризику інформаційної безпеки організації.

Стандарти безпечного кодування

Стандарти безпечного кодування CERT: Підрозділ CERT Інституту програмної інженерії (SEI) розробляє стандарти безпечного кодування для декількох мов програмування, включаючи C, C++, Java та Perl. Ці стандарти допомагають розробникам уникати поширених пасток безпеки.

OWASP (Open Web Application Security Project): OWASP надає рекомендації та найкращі практики щодо безпеки веб-додатків, в тому числі OWASP Top Ten, в якому перераховані найбільш критичні ризики безпеки веб-додатків.

Галузеві стандарти

HIPAA (Health Insurance Portability and Accountability Act): У галузі охорони здоров'я HIPAA встановлює стандарти захисту конфіденційної інформації про пацієнтів. Програмне забезпечення, яке обробляє електронну захищену медичну інформацію (ePHI), має відповідати вимогам HIPAA.

Стандарт безпеки даних індустрії платіжних карток (PCI DSS): Програмне забезпечення, яке обробляє транзакції з платіжними картками, повинно відповідати вимогам PCI DSS, щоб забезпечити безпечну обробку, зберігання та передачу даних власників карток.

Найкращі практики для безпечної розробки програмного забезпечення

Дотримання найкращих практик розробки програмного забезпечення має вирішальне значення для зменшення ризиків безпеки та забезпечення відповідності стандартам. Деякі ключові практики включають

Перегляд коду та тестування

Рецензування колегами: Регулярні перевірки коду колегами можуть виявити потенційні проблеми з безпекою та забезпечити дотримання стандартів кодування.

Статичний і динамічний аналіз: Інструменти статичного аналізу (наприклад, SonarQube) та інструменти динамічного аналізу (наприклад, OWASP ZAP) можуть виявити вразливості безпеки та проблеми з продуктивністю.

Безпечний життєвий цикл розробки (SDL)

-Моделювання загроз: Виявлення потенційних загроз і розробка програмного забезпечення для їх усунення - це проактивний підхід до безпеки.

-Навчання з безпеки: Регулярне навчання розробників безпечним методам кодування та останнім загрозам безпеки є вкрай важливим.

Управління залежностями

-Сторонні бібліотеки: Використання сторонніх бібліотек може призвести до появи вразливостей, якщо ними не керувати належним чином. Такі інструменти, як Dependabot та Snyk, можуть допомогти виявити та виправити вразливі залежності.

-Регулярні оновлення: Оновлення програмного забезпечення та бібліотек найновішими патчами безпеки має вирішальне значення для усунення відомих вразливостей.

Висновок

Мови програмування відіграють центральну роль у розробці програмного забезпечення, впливаючи не лише на функціональність і продуктивність додатків, але й на їхню безпеку та відповідність стандартам. Розуміння сильних і слабких сторін різних мов, а також їхнього впливу на безпеку є важливим для прийняття обґрунтованих рішень при розробці програмного забезпечення.

Дотримання безпечних практик кодування, використання відповідних

інструментів і дотримання відповідних стандартів є фундаментальними кроками у створенні безпечного і надійного програмного забезпечення. Оскільки цифровий ландшафт продовжує розвиватися, для розробників і організацій залишатиметься критично важливим бути в курсі останніх подій у мовах програмування, загроз безпеці та вимог щодо відповідності стандартам.

Надаючи пріоритет безпеці та дотриманню стандартів з самого початку, розробники можуть створювати надійні додатки, які не лише відповідають функціональним вимогам, але й захищають конфіденційні дані та забезпечують довіру користувачів.

2.3 Продуктивність та маштабованість

Фактори продуктивності

Продуктивність мов програмування визначається декількома факторами, серед яких:

-Компіляція проти інтерпретації: Скомпільовані мови, такі як C та C++, зазвичай мають вищу продуктивність, ніж інтерпретовані мови, такі як Python, оскільки код перекладається безпосередньо на машинну мову, яку процесор може виконати швидше.

-Керування пам'яттю: Ефективне керування пам'яттю, чи то через ручне керування (як у C та C++), чи то через автоматичне збирання сміття (як у Java та Python), має значний вплив на продуктивність. Мови, які дозволяють тонке керування пам'яттю, можуть оптимізувати критичні до продуктивності програми, але ціною збільшення складності.

-Підтримка паралелізму: Здатність мови ефективно керувати паралельним виконанням завдань впливає на її продуктивність у багатопотоковому або розподіленому середовищі. Такі мови, як Go та Java, надають надійні механізми паралелізму, які підвищують продуктивність у сучасних середовищах паралельних обчислень.

-Стандартні бібліотеки та екосистема: Наявність оптимізованих бібліотек та фреймворків може значно покращити продуктивність мови. Наприклад, хоча Python є інтерпретованою мовою, вона може використовувати високооптимізовані бібліотеки, такі як NumPy, для задач, що вимагають високої продуктивності.

Фактори масштабованості

-Масштабованість - це здатність мови та середовища її виконання справлятися зі зростаючими навантаженнями. Ключові фактори масштабованості включають

-Паралельність і паралелізм: Мови, які надають потужну підтримку паралелізму, такі як Go та Java, можуть ефективніше масштабуватися у багатоядерних та розподілених системах.

-Асинхронне програмування: Асинхронні моделі програмування, такі як Node.js (JavaScript) та asyncio в Python, дозволяють додаткам обробляти більше паралельних операцій без блокування, покращуючи масштабованість.

-Підтримка розподілених обчислень: Мови та екосистеми, які сприяють розподіленим обчисленням (наприклад, Java з її надійною JVM та фреймворками, такими як Hadoop), краще підходять для масштабованих додатків.

-Мікросервіси та контейнеризація: Підтримка архітектури мікросервісів та контейнеризації (наприклад, Docker, Kubernetes) може покращити масштабованість додатків, дозволяючи різним сервісам масштабуватися незалежно.

Аналіз конкретних мов

C та C++

Продуктивність:

-Висока продуктивність: C та C++ відомі своєю високою продуктивністю завдяки прямій компіляції в машинний код та ефективному управлінню пам'яттю.

-Ручне управління пам'яттю: Пропонують тонкий контроль над пам'яттю, що дозволяє оптимізувати під конкретне обладнання та вимоги до продуктивності. Однак це може призвести до складнощів, таких як витоки пам'яті та переповнення буферів.

Масштабованість:

-Обмежена вбудована підтримка паралелізму: Хоча мови C та C++ підтримують багатопотоковість, їм бракує вбудованих абстракцій для

паралельного програмування, що ускладнює написання масштабованих паралельних додатків без сторонніх бібліотек.

-Мікросервіси та розподілені системи: Вони менш широко використовуються в сучасних архітектурах мікросервісів і розподілених системах, ніж такі мови, як Java і Go, які пропонують більш готову підтримку цих парадигм.

Java

Продуктивність:

-Оптимізація JVM: Продуктивність Java підвищується завдяки компіляції just-in-time (JIT) та іншим оптимізаціям, що надаються віртуальною машиною Java (JVM).

-Збір сміття: Автоматичне збирання сміття спрощує керування пам'яттю, але може спричинити накладні витрати, які можуть вплинути на продуктивність чутливих до затримок додатків.

Масштабованість:

-Надійна підтримка паралелізму: Java забезпечує надійну підтримку паралелізму через пакет java.util.concurrent та фреймворк Fork/Join, що робить її добре придатною для масштабованих багатопотокових додатків.

-Рішення корпоративного рівня: Java є найкращою мовою для великомасштабних корпоративних додатків завдяки своїй масштабованості, багатим бібліотекам та фреймворкам, таким як Spring та Hibernate.

-Мікросервіси та хмара: Java широко використовується в архітектурах мікросервісів, підтримується такими фреймворками, як Spring Boot, і добре інтегрується з хмарними платформами та технологіями контейнеризації.

Python

Продуктивність:

-Інтерпретована мова: Python є інтерпретованою мовою, що зазвичай призводить до повільнішого виконання порівняно з скомпільованими мовами, такими як C++ або Java. Однак продуктивність Python можна значно покращити, використовуючи компіляцію just-in-time (наприклад, PyPy) та оптимізовані бібліотеки (наприклад, NumPy, SciPy).

-Динамічний набір тексту: Динамічна типізація Python може спричинити накладні витрати під час виконання, які можуть вплинути на продуктивність у додатках з інтенсивними обчисленнями.

Масштабованість:

-Асинхронне програмування: Бібліотека asyncio у Python дозволяє асинхронне програмування, що може покращити масштабованість програм, прив'язаних до вводу/виводу.

-Проблеми паралелізму: Глобальне блокування інтерпретатора (GIL) в CPython може бути вузьким місцем для прив'язаних до процесора багатопотокових додатків, обмежуючи масштабованість Python в деяких сценаріях. Обхідні шляхи включають багатопроцесорну обробку або використання альтернативних реалізацій, таких як Jython або IronPython.

-Мікросервіси та контейнери: Python популярний в архітектурі мікросервісів, підтримується такими фреймворками, як Flask та Django, і добре інтегрується з технологіями контейнеризації.

Go

Продуктивність:

-Скомпільована мова: Go - це компільована мова, що пропонує високу продуктивність, порівнянну з C та C++.

-Ефективний паралелізм: Легкі підпрограми Go та ефективна модель паралелізму дозволяють створювати високопродуктивні паралельні додатки.

Масштабованість:

-Паралельність і паралелізм: Вбудована підтримка паралелізму через підпрограми та канали робить Go дуже масштабованою для багатопотокових та розподілених додатків.

-Мікросервіси: Go добре підходить для архітектур мікросервісів, завдяки своїй простоті, продуктивності та надійній стандартній бібліотеці. Він легко інтегрується з інструментами контейнеризації, такими як Docker, та платформами оркестрування, такими як Kubernetes.

Порівняльний аналіз

Продуктивність:

-Висока продуктивність: C, C++ та Go лідирують за продуктивністю

завдяки своїй скомпільованій природі та ефективному управлінню пам'яттю.

-Помірна продуктивність: Java забезпечує хорошу продуктивність завдяки оптимізації JVM та автоматизованому управлінню пам'яттю, що робить її збалансованим вибором для багатьох додатків.

-Нижча продуктивність: Python, як інтерпретована мова, зазвичай відстає в продуктивності, але компенсує це великими бібліотеками та простотою використання для швидкої розробки.

Масштабованість:

-Висока масштабованість: Go та Java мають відмінні показники масштабованості завдяки потужній підтримці паралелізму, надійним екосистемам та придатності до архітектури мікросервісів.

-Помірно масштабовані: Python може бути масштабованим, особливо для додатків з прив'язкою до вводу/виводу, що використовують асинхронне програмування, але стикається з проблемами при виконанні завдань з прив'язкою до процесора через GIL.

-Проблеми з масштабуванням: C та C++ можуть досягати масштабованості, але часто вимагають значних зусиль для управління паралельними та розподіленими обчисленнями, що робить їх менш ідеальними для сучасних масштабованих архітектур.

Висновок

Продуктивність і масштабованість мови програмування є критично важливими факторами при розробці програмного забезпечення, що впливають на ефективність, швидкість реагування і потенціал зростання додатків. C і C++ пропонують неперевершену продуктивність і тонкий контроль, що робить їх придатними для критично важливих до продуктивності додатків, але менш ідеальними для масштабованих паралельних систем без значних накладних витрат. Java пропонує збалансоване поєднання продуктивності та масштабованості, підтримуване надійною екосистемою, що робить її кращим вибором для корпоративних та великомасштабних додатків. Python, незважаючи на нижчу початкову продуктивність, відрізняється швидкою розробкою та масштабованістю для завдань, пов'язаних з вводом/виводом, завдяки широким бібліотекам та фреймворкам. Go має високу продуктивність і

вбудовану підтримку паралелізму, що робить її чудовим вибором для масштабованих, паралельних і розподілених систем.

Вибір правильної мови програмування залежить від конкретних вимог програми, включаючи вимоги до продуктивності, масштабованості, швидкості розробки, а також існуючої екосистеми та інструментів. Розуміння цих факторів допомагає розробникам і організаціям приймати обґрунтовані рішення для оптимізації своїх програмних систем для поточних і майбутніх потреб.

РОЗДІЛ 3.

РОЗРОБКА ТА ТЕСТУВАННЯ МОДЕЛІ.

3.1 Мова програмування Java

Java - одна з найпоширеніших мов програмування у світі, відома своєю універсальністю, платформонезалежністю та надійністю. З моменту свого створення у 1995 році Джеймсом Гослінгом та його командою в компанії Sun Microsystems (зараз є частиною корпорації Oracle), Java перетворилася з нішевої мови для інтерактивного телебачення на потужну мову, що використовується в широкому спектрі обчислювальних платформ, від вбудованих пристроїв до корпоративних серверів і суперкомп'ютерів. Така повсюдність значною мірою пов'язана з основними особливостями Java, які включають простоту, об'єктно-орієнтований дизайн, безпеку та крос-платформні можливості. .

Основна філософія Java

Розвиток Java керувався кількома ключовими принципами, які визначили її еволюцію:

1. Простота, об'єктно-орієнтованість і звичність: Java була розроблена, щоб бути простою у використанні та доступною для розробників. Її синтаксис чистий і легко читається, особливо для тих, хто знайомий з C або C++. Об'єктно-орієнтований підхід допомагає організувати складне програмне забезпечення в керовані, багаторазові та модульні частини.

2. Надійність та безпека: Java робить сильний акцент на ранній перевірці помилок та перевірці під час виконання. Це робить мову надійною та менш схильною до збоїв. Функції безпеки вбудовані в мову та систему виконання, що дозволяє створювати системи, захищені від несанкціонованого доступу та вірусів.

3. Незалежність від платформи: Однією з ключових переваг Java є її незалежність від платформи, як на рівні коду, так і на рівні двійкового коду. Код Java компілюється у байт-код, який може працювати на будь-якому пристрої з віртуальною машиною Java (JVM), що робить його дуже

портативним.

4. Висока продуктивність: Хоча Java є інтерпретованою мовою, такі досягнення, як компіляція Just-In-Time (JIT), значно підвищили її продуктивність, зробивши її конкурентоспроможною з мовами, що компілюються нативно, такими як C++.

5. Багатопотоковість та інтерактивність: Java підтримує багатопоточність на рівні мови, що дозволяє легко писати програми, які можуть виконувати кілька завдань одночасно.

6. Динамічна та розширювана: Java створена для адаптації до мінливого середовища. Класи в Java зв'язуються (завантажуються) лише тоді, коли це необхідно, що дозволяє оновлювати та обслуговувати їх, не впливаючи на загальну систему.

Використання Java в розробці

Універсальність Java означає, що її можна знайти в широкому діапазоні областей розробки:

1. Корпоративні додатки: Java - це мова вибору для створення великомасштабних корпоративних додатків. Такі технології, як Java EE (Enterprise Edition), забезпечують надійну платформу для створення розподілених, масштабованих і безпечних корпоративних рішень. Java EE включає в себе ряд API і сервісів, таких як сервлети, JavaServer Pages (JSP), Enterprise JavaBeans (EJB) та інші, які спрощують розробку великомасштабних додатків.

2. Мобільні додатки: Java є офіційною мовою розробки для Android. Додатки для Android переважно написані на Java, що робить її однією з найважливіших мов для розробки мобільних додатків. Android SDK надає широкі бібліотеки та інструменти для розробки насичених та інтерактивних додатків.

3. Веб-додатки: Java широко використовується для розробки динамічних веб-додатків. Такі фреймворки, як Spring, Struts і Hibernate, зробили революцію в розробці веб-додатків, надавши надійні інструменти для створення та управління веб-сервісами і додатками.

4. Наукові додатки: Багато наукових програм, особливо тих, що

потребують обробки великих обсягів даних, написані на Java. Її стабільність, масштабованість і висока продуктивність роблять її чудовим вибором для таких додатків. Такі бібліотеки, як Apache Commons Math і JFreeChart, широко використовуються в наукових обчисленнях.

5. Ігри: Хоча мова Java не так широко використовується у світі розробки ігор, як C++, її використовують для розробки ігор, особливо на мобільних платформах. Здатність Java обробляти високопродуктивну графіку та її великі бібліотеки, такі як Lightweight Java Game Library (LWJGL), роблять її придатною для розробки ігор.

6. Вбудовані системи: Java також використовується у вбудованих системах завдяки своїй портативності та ефективності. Java ME (Micro Edition) забезпечує надійну платформу для створення додатків для пристроїв з обмеженими ресурсами.

Переваги використання Java

Широке розповсюдження Java можна пояснити багатьма перевагами, які вона пропонує:

1. Незалежність від платформи: Можливість Java «написати один раз, запустити будь-де» є однією з її ключових переваг. JVM дозволяє запускати Java-додатки на будь-якому пристрої або операційній системі, що підтримує JVM, усуваючи необхідність перекомпіляції для різних платформ.

2. Об'єктно-орієнтована: Об'єктно-орієнтована природа Java полегшує модульну, гнучку та розширювану розробку програмного забезпечення. Об'єктно-орієнтоване програмування (ООП) сприяє кращій організації коду, повторному використанню програмних компонентів і чіткій структурі складних додатків.

3. Надійність: Java розроблена для усунення ситуацій, схильних до помилок, завдяки перевірці помилок під час компіляції та виконання. Такі функції, як автоматичне збирання сміття, обробка винятків та використання потужних механізмів перевірки типів, сприяють підвищенню надійності Java.

4. Безпека: Безпека є першочерговим завданням для Java. Мова забезпечує безпечне середовище за допомогою JVM, яка керує виконанням байт-коду та забезпечує рівень абстракції між додатком та машиною. Java

також надає API для різних функцій безпеки, таких як криптографія, контроль доступу та автентифікація.

5. Широкий набір API: Java постачається з повним набором API (інтерфейсів прикладного програмування), що охоплює все: від структур даних, мереж та утиліт до розбору XML, підключення до баз даних (через JDBC) та багато іншого. Ця велика стандартна бібліотека підтримує швидку розробку, надаючи попередньо створену функціональність.

6. Підтримка спільноти та документація: Java має велику спільноту розробників та обширну документацію. Ця потужна підтримка спільноти означає, що розробники можуть легко знайти допомогу, навчальні посібники та ресурси, які можуть бути критично важливими для усунення несправностей та навчання.

7. Продуктивність: Незважаючи на те, що Java є інтерпретованою мовою, її продуктивність значно покращилася з роками. Такі технології, як компіляція Just-In-Time (JIT) та вдосконалення JVM скоротили розрив у продуктивності між Java та мовами, що компілюються нативно.

8. Багатопоточність: Java була розроблена з урахуванням багатопоточності з самого початку. Мова надає високорівневі утиліти паралелізму та всеосяжний фреймворк синхронізації, що дозволяє легко створювати додатки, які можуть виконувати декілька завдань одночасно.

9. Легка у вивченні та використанні: Синтаксис мови Java простий і схожий на інші популярні мови, такі як C та C++, що полегшує її вивчення для розробників. Її чітка та стисла документація також допомагає процесу навчання.

10. Масштабованість: Додатки на Java мають високу масштабованість, що особливо важливо для додатків корпоративного рівня, які повинні справлятися зі зростаючими робочими навантаженнями. Такі технології, як Java EE, забезпечують надійні механізми для створення масштабованих додатків.

11. Інструменти та середовище розробки: Java підтримується багатим набором інструментів розробки та інтегрованих середовищ розробки (IDE), таких як IntelliJ IDEA, Eclipse та NetBeans. Ці інструменти надають потужні

функції, такі як завершення коду, рефакторинг, налагодження та багато іншого для спрощення процесу розробки.

Фреймворки та екосистема Java

Екосистема навколо Java дуже широка, з численними фреймворками та інструментами, які розширюють її можливості:

1. Spring Framework: Один з найпопулярніших фреймворків Java, Spring спрощує розробку корпоративних додатків, надаючи комплексну підтримку інфраструктури. Він включає в себе модулі для ін'єкції залежностей, аспектно-орієнтованого програмування, управління транзакціями та багато іншого, що дозволяє розробникам створювати надійні, підтримувані додатки.

2. Hibernate: Потужний фреймворк об'єктно-реляційного відображення (ORM), Hibernate спрощує взаємодію з базами даних у додатках на Java. Він абстрагує основні взаємодії з базами даних, дозволяючи розробникам працювати з високорівневими об'єктними моделями замість низькорівневого SQL.

3. Apache Struts: Надійний фреймворк для розробки веб-додатків, Struts розширює Java Servlet API і використовує архітектуру Model-View-Controller (MVC) для розділення рівнів логіки, представлення і даних, що спрощує розробку і підтримку.

4. JavaFX: JavaFX використовується для створення багатих інтернет-додатків із сучасним користувацьким інтерфейсом. Вона надає набір графічних і мультимедійних пакетів, які дозволяють розробникам проектувати, створювати, тестувати і розгортати багаті клієнтські програми.

5. JUnit: Важливий інструмент для розробки, орієнтованої на тестування, JUnit - це фреймворк для тестування, який дозволяє розробникам писати і виконувати повторювані тести. Він є невід'ємною частиною процесу розробки Java, забезпечуючи надійність та стійкість коду завдяки ретельному тестуванню.

6. Apache Maven та Gradle: Це потужні інструменти автоматизації збірки, що використовуються переважно для проектів на Java. Вони керують залежностями проекту, процесами збірки тощо, допомагаючи підтримувати узгодженість збірок у різних середовищах

Java в сучасному ландшафті розробки

У сучасному технологічному ландшафті, що швидко розвивається, Java залишається критично важливою мовою завдяки своїй адаптивності та потужному набору функцій. Декілька сучасних тенденцій та практик демонструють постійну актуальність Java:

1. Архітектура мікросервісів: Java з її надійними фреймворками, такими як Spring Boot, добре підходить для побудови архітектури мікросервісів. Мікросервіси розбивають додатки на менші, більш керовані частини, кожна з яких працює незалежно. Така архітектура покращує масштабованість, гнучкість та ремонтопридатність.

2. Хмарні обчислення: Java широко використовується у хмарних обчисленнях. Основні постачальники хмарних послуг, такі як AWS, Google Cloud Platform та Microsoft Azure, підтримують Java-додатки. Масштабованість і надійність Java роблять її ідеальною для хмарних додатків і сервісів.

3. Великі дані: У сфері великих даних Java часто використовується для розробки масштабованих, високопродуктивних додатків. Такі фреймворки, як Apache Hadoop та Apache Spark, в основному написані на Java, що демонструє здатність мови справлятися з завданнями обробки великих обсягів даних.

4. Інтернет речей (IoT): Портативність Java робить її природною для розробки IoT. Java ME і Java SE використовуються для розробки додатків для пристроїв IoT, забезпечуючи масштабованість і надійність, необхідні для цих додатків.

5. Штучний інтелект (AI) і машинне навчання (ML): Хоча такі мови, як Python, частіше асоціюються зі штучним інтелектом і машинним навчанням, Java також відіграє важливу роль. Такі бібліотеки, як Weka, Deeplearning4j і MOA, використовуються для створення додатків ШІ і МН на Java.

3.2 Архітектура моделі

Компоненти та їх взаємодія

MessengerApp JFrame

-Головне вікно програми, яке містить усі компоненти інтерфейсу користувача.

-Використовує BorderLayout для розташування компонентів.

Панель пошуку (Search Panel):

-searchField: Текстове поле для введення пошукових запитів.

-searchButton: Кнопка для ініціювання пошуку користувачів.

Панель списку користувачів (User List Panel):

-userList: Список користувачів, який відображається за допомогою JScrollPane.

-userListModel: Модель списку, що зберігає дані користувачів.

Панель області чату (Chat Area Panel):

-chatArea: Текстова область для відображення повідомлень чату, що також використовується з JScrollPane.

Панель повідомлень (Message Panel):

-messageField: Текстове поле для введення повідомлень.

-sendButton: Кнопка для відправки повідомлень.

Структури даних (Data Structures):

-users: HashMap, що зберігає інформацію про користувачів.

-hats: HashMap, що зберігає історії чатів.

Функціональні потоки

Пошук користувача (Search User):

-Вхідні дані: Користувач вводить пошуковий запит у searchField і натискає searchButton.

-Процес: Метод searchUser() фільтрує користувачів та оновлює userListModel.

-Вихідні дані: Відповідні користувачі відображаються у userList.

Вибір користувача (Select User):

-Вхідні дані: Користувач вибирає користувача зі списку userList.

-Процес: Метод selectUser(User user) оновлює currentUser і відображає історію чату в chatArea.

-Вихідні дані: Історія чату відображається у chatArea.

Відправка повідомлення (Send Message):

-Вхідні дані: Користувач вводить повідомлення у messageField і натискає sendButton.

-Процес: Метод sendMessage() відправляє повідомлення, оновлює історію чату та очищає messageField.

-Вихідні дані: Повідомлення додається до chatArea і messageField очищується.

Спрощений потік даних

Взаємодія користувача:

-Пошук користувачів, вибір користувача, відправка повідомлень.

Компоненти інтерфейсу користувача:

-Панелі для пошуку, списку користувачів, області чату та введення повідомлень.

3.3 Побудова та тестування програми

Ця програма є простим чатом, який можна використовувати для надсилання текстових повідомлень між користувачами.

-спілкування з друзями та колегами:

Пошук користувачів: Користувач може використовувати поле пошуку, щоб знайти інших користувачів за іменем або номером телефону.

-Вибір користувачів: Знайшовши користувача, користувач може вибрати його зі списку користувачів.

- Обмін повідомленнями: Після вибору користувача користувач може переглянути історію повідомлень з цим користувачем і відправити нові повідомлення за допомогою текстового поля.

Використання додатку:

Пошук користувача:

-Введіть ім'я або номер телефону користувача в поле пошуку та натисніть кнопку "Search", щоб знайти його у списку.

-Вибір користувача: Оберіть користувача зі списку, натиснувши на його ім'я.

-Обмін повідомленнями: Введіть текст повідомлення в поле для введення тексту та натисніть кнопку "Send", щоб надіслати його обраному користувачеві.

-Перегляд історії повідомлень: Історія повідомлень з обраним

користувачем буде відображена у центральній частині вікна.

-Вихід: Для виходу з програми просто закрийте вікно додатку.

Тестування:

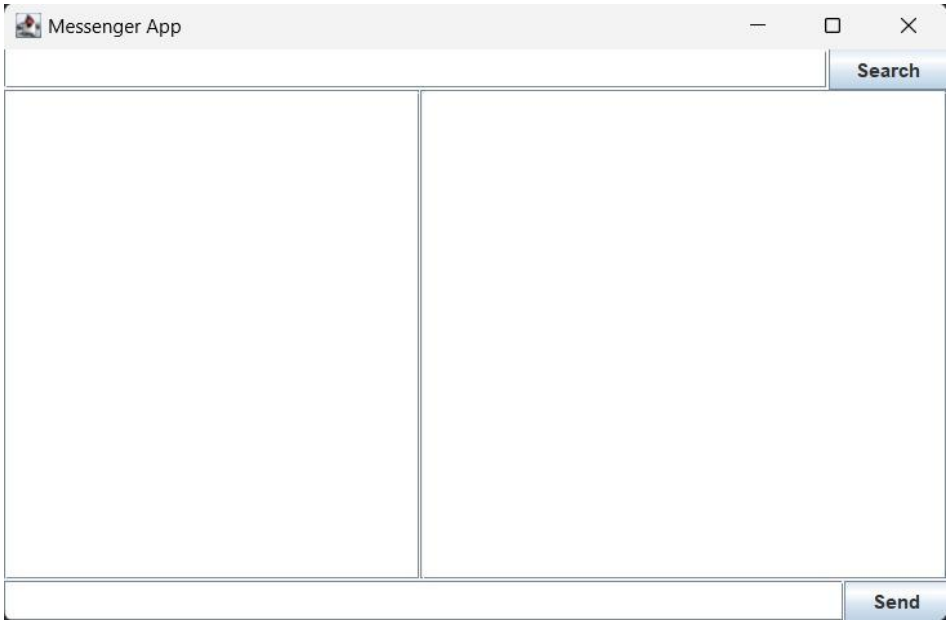

Рисунок 3.1 - Графічний інтерфейс програми

На рисунку зображено інтерфейс програми для обміну повідомленнями, яка називається “Messenger App”. Вікно має стандартну панель заголовка з кнопками згортання, розгортання та закриття у верхньому правому куті. У верхній правій частині всередині програми є рядок пошуку. Основна область програми поділена на дві вертикальні секції. У нижній частині правої секції є поле введення з кнопкою “Надіслати” поруч, для набору та відправлення повідомлень

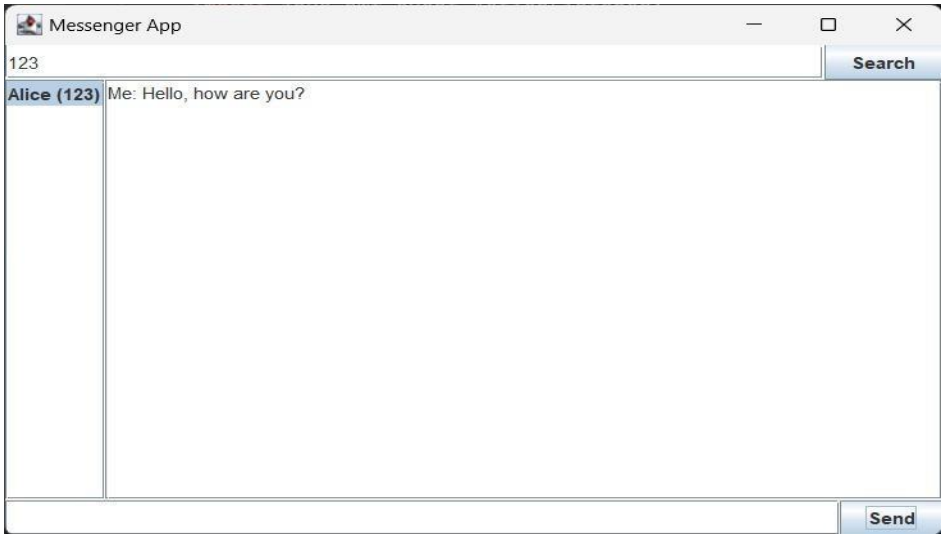

Рисунок 3.2 - інтерфейс доставки повідомленнь користувачеві

На рисунку зображено інтерфейс програми для обміну повідомленнямина екрані комп’ютера. Зліва є рядок пошуку з введеним номером “123” та кнопкою “Пошук” поруч. Нижче цієї області є список з одним видимим контактом на ім’я “Alice (123)”, виділеним синім кольором, що вказує на активний чат. У головній області чату показано одне повідомлення для Alice, яке говорить: “Я: Привіт, як справи?”. І так далі можна писати в рядок для пошуку та знаходити інших зареєстрованих користувачів за присвоєним їм кодом та починати з ними діалог. Але створювати групи між великою кількістю людей та відправляти фотографії\відео як наприкал в месенджері “Telegram” або інших популярних додатках, неможливо.

Висновки за розділом 3:

В даному розділі я розповів чому обрав для створення додатку саме мову програмування Java. Навів плюси та мінуси даної мови. Також показав та розповів про архітектуру моделі мого застосунку. Потім розповів для чого потрібна моя програма та як нею користуватись, провів тестування додатку після чого помилок в роботі не виявив.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було створено та вивчено модель застосунку месенджеру мовою програмування Java. Ця робота була розділена та три основні етапи, такі як: аналіз вимог до месенджерів, вибір технологій програмування, розробка та тестування моделі.

У першому розділі було висвітлено загальну історію розвитку месенджерів. Було описано початкові етапи становлення месенджерів, зазначено, як один месенджер поступово заміняв інший, більш технологічно розвинений. Проведено аналіз популярних месенджерів, що дозволило дізнатися про їхні основні характеристики та відмінності.

У другому розділі здійснено аналіз та порівняння різних мов програмування. Обговорено їхні переваги та недоліки, визначено, для яких завдань вони найкраще підходять, а для яких – ні. Особлива увага приділена питанням безпеки та відповідності мов програмування сучасним стандартам. Розглянуто сильні та слабкі сторони різних мов програмування, а також їхню ефективність у різних контекстах.

У третьому розділі обґрунтовано вибір мови програмування Java для створення додатку. Детально розглянуто всі можливі переваги та недоліки цієї мови. Далі висвітлено призначення розробленого додатку та способи його використання. Завершується розділ описом процесу тестування роботи додатку, що дозволило оцінити його функціональність та надійність.

Мова програмування Java була обрана для створення додатку з огляду на її численні переваги, зокрема стабільність, надійність та широкі можливості для реалізації функціоналу. Опис призначення додатку та способів його використання підкреслив його корисність та практичну цінність. Процес тестування підтвердив працездатність та ефективність створеного додатку.

Таким чином, проведене дослідження не лише дозволило зрозуміти еволюцію

месенджерів та вибрати оптимальну мову програмування, але й забезпечило успішне створення функціонального додатку, що відповідає сучасним вимогам користувачів та стандартам безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Блох, Джошуа. "Effective Java". Нью-Йорк: Addison-Wesley, 2018. ст. 37-52.

2. Сьєрра, Кеті, Берт Бейтс. "Head First Java". Себастополь: O'Reilly Media, 2005. ст. 553-582.

3. Шильдт, Герберт. "Java: A Beginner's Guide". Нью-Йорк: McGraw-Hill Education, 2018. ст. 741-778.

4. Ґетц, Браян, Тім Пейрлс, Джошуа Блох, Джозеф Бовбер, Девід Холмс, Даг Лі. "Java Concurrency in Practice". Нью-Йорк: Addison-Wesley, 2006. ст. 123-140.

5. Нафталін, Моріс, Філіп Вадлер. "Java Generics and Collections". Нью-Йорк: O'Reilly Media, 2006. ст. 63-84.

6. Оукс, Скотт. "Java Performance: The Definitive Guide". Нью-Йорк: O'Reilly Media, 2014. ст. 23-44.

7. Шильдт, Герберт. "Java: The Complete Reference". Нью-Йорк: McGraw-Hill Education, 2018. ст. 177-216.

8. Тідам, Алекс. "Java EE 8: Only What's New". Нью-Йорк: O'Reilly Media, 2018.

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **бакалавр**
Галузь знань: **12 – Інформаційні технології**
Спеціальність: **123 – Комп'ютерна інженерія.**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної
та прикладної системотехніки
д.т.н., проф. Шматков С. І.
«21» грудня 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Коваленко Іван Владиславович

1. Тема роботи **«Модель застосунку-мессенджеру на мові програмування Java»**
керівник роботи Мірошник Марина Анатоліївна д.т.н, професор кафедри ТПС
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «03» травня 2024року № 4101-5/909

2.Строк подання студентом роботи 31 травня 2024року

3. Перелік питань, які потрібно розробити:

1) Дослідження відповідної літератури для оформлення проекту.
   Розробка структури додатку.
2) Проектування інтерфейсу додатку.
3) Тестування додатку на предмет помилок або некоректної роботи.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Аналіз научної літератури	21.12.2023 – 05.01.2024
2	Розробка скелету програми.	6.03.2024 - 14.03.2024
3	Реалізація розробленого плану.	16.03.2024 - 6.04.2024
4	Тестування та апробація комп'ютерної моделі.	7.04.2024 - 15.04.2024
5	Підготовка наукової статті	16.04.2024 - 30.04.2024
6	Розробка пояснювальної записки.	31.04.2024 - 27.05.2024
7	Представлення кваліфікаційної роботи керівнику та рецензенту	16.05.2024 – 21.05.2024
8	Оформлення супроводжувальної документації	22.05.2024 – 27.05.2024
9	Оформлення звіту за результатами переддипломної практики	22.05.2024 – 05.06.2024

5. Дата видачі завдання 21.12.2023

Студент І. В. Коваленко
ініціали, прізвище
підпис

Керівник роботи М. А. Мірошник
ініціали, прізвище
підпис

Додаток Б

Технічне завдання
на розробку програмного вироби
«Модель застосунку-мессенджеру на мові програмуванняJava»

Назва розділу	Назва і зміст підрозділу
1. Введення	1.1. Назва програмного виробу – Модель застосунку-мессенджеру на мові програмуванняJava. 1.2. Галузь застосування – 12 – Інформаційні технології.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – Комп'ютерна інженерія. 2.2. Завдання на кваліфікаційну роботу бакалавра, затверджено наказом ХНУ імені В. Н. Каразіна № 4101-5/909 від 03.05.2024 р. (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3. Призначення розробки	3.1. Мета розробки програмного вироби – допомога при виборі архітектурного підходу при створенні IT-проєкту за рахунок використання зібраних параметрів при тестуванні. 3.2. Призначення програмного вироби для автоматизації вибору архітектурного підходу за параметрами. 3.3. Початкові дані для розробки: лист задач, їх характеристики.
4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: 1) представляти з себе програмну реалізацію 2) додати три додатки які використовують три архітектурних підходи для демонстрації особливостей впровадження, і надання можливості особистого тестування 3) надавати зібрані дані при тестуванні 4) надавати додаток з візуальним інтерфейсом який буде зберігати в собі всі дані та містити формули для підрахунку підходящого архітектурного підходу за наданими параметрами та умовами. 4.2. Вимоги до надійності: Можна обрати архітектурний підхід за ваговими коефіцієнтами бажаних характеристик. Розроблені моделі мають однакові відповіді при однакових запитах. 4.3. Вимоги до умов експлуатації офісні приміщення. 4.4. Вимоги до складу і параметрів технічних засобів Персональний комп'ютер у повній комплектації (ноутбук) 4.5. Вимоги до інформаційної та програмної сумісності забезпечити сумісність з усіма обчислювальними засобами. 4.6. Вимоги до маркування та упаковки відсутні. 4.7. Вимоги до транспортування і зберігання відсутні. 4.8. Спеціальні вимоги не пред'являються.
5. Вимоги до програмної	Програмною документацією до виробу «Модель застосунку-мессенджеру на мові програмуванняJava» вважати:

<table>
<tr><td>документації .</td><td colspan="3">1) Справжнє Технічне завдання на розробку програмного вироби (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Програму і методику випробувань розробленого програмного вироби (представити у вигляді Додатку В до пояснювальної записки до кваліфікаційної роботи).
3) Опис програмного виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи).</td></tr>
<tr><td>6. Техніко-економічні показники</td><td colspan="3">Вимоги до розрахунку техніко-економічних показників не потрібні.</td></tr>
<tr><td rowspan="9">7. Стадії та етапи розробки</td><td>1 етап</td><td>Аналіз научної літератури</td><td>21.12.2023 – 05.01.2024</td></tr>
<tr><td>2 етап</td><td>Розробка скелету програми.</td><td>6.03.2024 - 14.03.2024</td></tr>
<tr><td>3 етап</td><td>Реалізація розробленого плану.</td><td>16.03.2024 - 6.04.2024</td></tr>
<tr><td>4 етап</td><td>Тестування та апробація комп’ютерної моделі.</td><td>7.04.2024 - 15.04.2024</td></tr>
<tr><td>5 етап</td><td>Підготовка наукової статті</td><td>16.04.2024 - 30.04.2024</td></tr>
<tr><td>6 етап</td><td>Розробка пояснювальної записки.</td><td>31.04.2024 - 27.05.2024</td></tr>
<tr><td>7 етап</td><td>Представлення кваліфікаційної роботи керівнику та рецензенту</td><td>16.05.2024 – 21.05.2024</td></tr>
<tr><td>8 етап</td><td>Оформлення супроводжувальної документації</td><td>22.05.2024 – 27.05.2024</td></tr>
<tr><td>9 етап</td><td>Оформлення звіту за результатами переддипломної практики</td><td>22.05.2024 – 05.06.2024</td></tr>
<tr><td>8. Порядок контролю і приймання</td><td colspan="3">1) Перевірку ходу розробки програмного виробу керівнику робіт виконувати раз в 2 тижні.
2) Випробування програмного продукту провести відповідно до програми та методики випробувань на базі комп’ютерного класу.
3) Захист розробленої моделі провести на засіданні Атестаційної комісії.
4) Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику.</td></tr>
</table>

Виконавець

студент групи КІ-41

Замовник

д.т.н, професор кафедри ТПС

І. В. Коваленко М. А. Мірошник _____

_____

Додаток В

Програма і методика випробувань програмного вироби «Модель застосунку-мессенджеру на мові програмування Java»

1 Об'єкт випробувань

1.1 Найменування випробуваного програмного вироби «Модель застосунку-мессенджеру на мові програмуванняJava».

1.2 Область його застосування **Комп'ютерна інженерія.**.

2. Мета випробувань

Метою кваліфікаційної роботи є дослідження та порівняння різних архітектурних підходів написання програмного забезпечення.

Загальні положення

2.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2.2 Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

2.3 Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї Програми і методики випробувань.

2.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

3. Вимоги до програми або програмного виробу

4.1. Вимоги до функціональних характеристик:

1) представляти з себе програмну реалізацію

2) додати три додатки які використовують три архітектурних підходи для демонстрації особливостей впровадження, і надання можливості особистого тестування

3) надавати зібрані дані при тестуванні

4) надавати додаток з візуальним інтерфейсом який буде зберігати в собі всі дані та містити формули для підрахунку балів архітектурного підходу за наданими параметрами та умовами.

4.2. Вимоги до надійності:

Можна обрати архітектурний підхід за ваговими коефіцієнтами бажаних характеристик. Розроблені моделі мають однакові відповіді при однакових запитах.

4.3. Вимоги до умов експлуатації офісні приміщення.

4.4. Вимоги до складу і параметрів технічних засобів Персональний комп’ютер у повній комплектації (ноутбук)

4.5. Вимоги до інформаційної та програмної сумісності забезпечити сумісність з усіма обчислювальними засобами.

4.6. Вимоги до маркування та упаковки відсутні.

4.7. Вимоги до транспортування і зберігання відсутні.

4.8. Спеціальні вимоги відсутні.

4. Вимоги до програмної документації

Склад програмної документації, що подається на випробування, включає:

1) Технічне завдання на розробку програмного вироби (представлено в Додатку Б до пояснювальної записки до кваліфікаційної роботи).

2) Ця Програма і методика випробувань розробленого програмного вироби (представлена в Додатку В до пояснювальної записки до кваліфікаційної роботи).

3) Опис програмного вироби (представлено в розділі 3 пояснювальної записки до кваліфікаційної роботи).

5. Засоби і порядок випробувань

5.1 Засоби випробувань

Випробування проводяться на технічних засобах, яких персональний комп’ютер, ноутбук.

Випробування проводяться з використанням програмних засобів, яких EXCEL, командний рядок, Kubernetes, Docker, Postman.

5.2 Порядок проведення випробувань

5.2.1. Перевірка програмної документації.

Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в технічному завданні документації.

5.2.2. Перевірка якості програмної документації.

Перевірку здійснювати за критерієм відповідності вимогам єдиної системи програмної документації (ЄСПД).

5.2.3. Перевірка виконання програми.

Тест 1: Перевірка відповідей розроблених моделей при однакових запитах. По вмісту запиту та відповіді робиться висновок про правильно працездатність моделей (рисунки В.1, В.2).

Рисунок В.1 - графічний інтерфейс програми

Рисунок В.2 - інтерфейс доставки повідомленнь користувачеві

Висновок: Перевірка відповідей розроблених моделей при однакових запитах виконано успішно, всі відповіді однакові, при однакових запитах.
Тест 2: Перевірка обрання архітектурнрнр підходу за ваговими коефіцієнтами, з максимальними коефіцієнтами простоти впровадження та медіани затримки.
Висновки: при вдалому виконанні всіх 3 тестів випробування розробленого додатку вважаються успішними.

Виконавець
студент групи КІ-41
Коваленко І. В.

Додаток Г

Лістинг коду:

MessengerApp.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;
import java.util.Map;

public class MessengerApp extends JFrame {
private Map<String, User> users;
private Map<String, Chat> chats;

private JTextField searchField;
private JTextArea chatArea;
private JTextField messageField;
private JList<User> userList;
private DefaultListModel<User> userListModel;
private User currentUser;

public MessengerApp() {
users = new HashMap<>();
chats = new HashMap<>();

// Sample users
users.put("123", new User("Alice", "123"));
users.put("456", new User("Bob", "456"));
users.put("789", new User("Charlie", "789"));

setTitle("Messenger App");
setSize(600, 400);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLayout(new BorderLayout());

// Search Panel
JPanel searchPanel = new JPanel();
searchPanel.setLayout(new BorderLayout());
searchField = new JTextField();
JButton searchButton = new JButton("Search");
searchPanel.add(searchField, BorderLayout.CENTER);
searchPanel.add(searchButton, BorderLayout.EAST);

// User List
userListModel = new DefaultListModel<>();
userList = new JList<>(userListModel);
userList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

JScrollPane userScrollPane = new JScrollPane(userList);
```

```
// Chat Area
chatArea = new JTextArea();
chatArea.setEditable(false);
JScrollPane chatScrollPane = new JScrollPane(chatArea);

// Message Panel
JPanel messagePanel = new JPanel();
messagePanel.setLayout(new BorderLayout());
messageField = new JTextField();
JButton sendButton = new JButton("Send");
messagePanel.add(messageField, BorderLayout.CENTER);
messagePanel.add(sendButton, BorderLayout.EAST);

add(searchPanel, BorderLayout.NORTH);
add(userScrollPane, BorderLayout.WEST);
add(chatScrollPane, BorderLayout.CENTER);
add(messagePanel, BorderLayout.SOUTH);

// Action Listeners
searchButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
searchUser();
}
});

sendButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
sendMessage();
}
});

userList.addListSelectionListener(e -> {
if (!e.getValueIsAdjusting()) {
selectUser(userList.getSelectedValue());
}
});
}

private void searchUser() {
String searchTerm = searchField.getText();
userListModel.clear();
for (User user : users.values()) {
if (user.getUsername().contains(searchTerm) ||
user.getPhoneNumber().contains(searchTerm)) {
userListModel.addElement(user);
}
```

```
}
}

private void selectUser(User user) {
currentUser = user;
if (user != null) {
chatArea.setText("");
String chatKey = getChatKey(currentUser);
if (chats.containsKey(chatKey)) {
for (String message : chats.get(chatKey).getMessages()) {
chatArea.append(message + "\n");
}
}
}
}
private String getChatKey(User user) {
return "chat_" + user.getPhoneNumber(); // Simplified key
}

public static void main(String[] args) {
SwingUtilities.invokeLater(() -> {
MessengerApp app = new MessengerApp();
app.setVisible(true);
});
}
}


private void sendMessage() {
if (currentUser != null) {
String message = messageField.getText();
if (!message.isEmpty()) {
String chatKey = getChatKey(currentUser);
chats.putIfAbsent(chatKey, new Chat(currentUser, currentUser)); // Simplified
for single user chat
chats.get(chatKey).addMessage("Me: " + message);
chatArea.append("Me: " + message + "\n");
messageField.setText("");
}
}
}
```

Chat.java

```
import java.util.ArrayList;
import java.util.List;

class Chat {
private User user1;
private User user2;
private List<String> messages;

public Chat(User user1, User user2) {
this.user1 = user1;
this.user2 = user2;
this.messages = new ArrayList<>();
}

public void addMessage(String message) {
messages.add(message);
}

public List<String> getMessages() {
return messages;
}

public User getUser1() {
return user1;
}

public User getUser2() {
return user2;
}

@Override
public String toString() {
return user1.getUsername() + " and " + user2.getUsername();
}
}
```

User.java

```
class User {
private String username;
private String phoneNumber;

public User(String username, String phoneNumber) {
this.username = username;
this.phoneNumber = phoneNumber;
}
```

```
public String getUsername() {
return username;
}

public String getPhoneNumber() {
return phoneNumber;
}

@Override
public String toString() {
return username + " (" + phoneNumber + ")";
}
}
```