НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ В.П. Тарасенко
(підпис)
"___"_____2019р.

Дипломний проект

освітньо-кваліфікаційного рівня "Бакалавр"

з напряму підготовки 6.050102 "Комп'ютерна інженерія"

на тему: «ШАБЛОНІЗАТОР JAVA ДЛЯ АВТОМАТИЧНОЇ РОЗРОБКИ ГРАФІЧНОГО КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ»

Виконав: студент 4 курсу, групи КВ-53
Мандрік Марія Владиславівна _____
(підпис)

Керівник асистент каф. СПіСКС Радченко К.О. _____
(підпис)

Консультант з нормоконтролю доц., к.т.н. Клятченко Я.М. _____
(підпис)

Рецензент доц., к.т.н. Марковський О.П. _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет прикладної математики

Кафедра системного програмування та спеціалізованих комп’ютерних систем

Освітньо-кваліфікаційний рівень “Бакалавр”

Напрям підготовки 6.050102 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ В.П. Тарасенко
“___” _____ 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Мандрік Марії Владиславівні

1. Тема проекту «ШАБЛОНІЗОТОР JAVA ДЛЯ АВТОМАТИЧНОЇ РОЗРОБКИ ГРАФІЧНОГО КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ»,
керівник проекту Радченко Костянтин Олександрович, асистент каф. СПіСКС,
затверджені наказом по університету від “22” травня 2019 року № 1330-С

2. Строк подання студентом проекту: “13” травня 2019 р.

3. Вихідні дані для дипломного проектування: див. Технічне завдання.

4. Перелік задач, які потрібно вирішити:

- розробити набір шаблонів графічних компонентів (кнопки, прапорці, перемикачі, елементи меню тощо);
- забезпечити підтримку роботи з зовнішніми графічними ресурсами;
- забезпечити підтримку багатомовного інтерфейсу;
- забезпечити збереження стану графічного інтерфейсу при перезапуску програми;
- забезпечити підтримка шаблону MVC (Model-View-Controller);
- виконати тестування програми.

5. Перелік обов’язкового ілюстративного матеріалу:
- цикл роботи системи (креслення);
- діаграма взаємодії компонентів системи при реалізації шаблону MVC;
- взаємозв'язок між класами системи (креслення).

6. Консультанти:

Питання	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доц., к.т.н.		

7. Дата видачі завдання: “___” _____ 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів	Примітка
1.	Вивчення літератури за тематикою проекту	15.04.2019	
2.	Розробка та узгодження технічного завдання	30.04.2019	
3.	Аналіз існуючих рішень	05.05.2019	
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019	
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019	
6.	Підготовка графічної частини дипломного проекту	20.05.2019	
7.	Оформлення документації дипломного проекту	25.05.2019	
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019	

Студент _____ Мандрік М.В.

Керівник проекту _____ Радченко К.О.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (___ с., ___ рис., ___ табл., ___ додатки).

Об’єкт розробки – бібліотека автоматизації розробки графічного інтерфейсу для JVM-мов. Метою проекту є полегшення та пришвидшення розробки програмних компонентів з графічним інтерфейсом користувача шляхом розробки та включення у окрему бібліотеку основних шаблонів.

Передбачається використання розробленого комплексу при написанні програм мовами, що виконуються на Java Virtual Machine (JVM).

Розроблена бібліотека підтримує наступні елементи: - набір шаблонів графічних компонентів (кнопки, прапорці, перемикачі, елементи меню тощо)

- підтримка роботи з зовнішніми графічними ресурсами;

- підтримка багатомовного інтерфесу;

- збереження стану графічного інтерфейсу при перезапуску програми;

- підтримка шаблону MVC (Model-View-Controller).

Розроблений програмний комплекс являє собою високорівневу обгортку над стандартною графічною бібліотекою Swing.

Ключові слова: графічний інтерфейс користувача, графічна бібліотека, віджет, Kotlin, Swing.

ABSTRACT

Qualification work includes explanation note (___ p., ___ pic., ___ tab., ___ additions).

Object of development - JVM language graphing automation library. The purpose of the project is to facilitate and accelerate the development of software components with a graphical user interface by developing and including in a separate library of the main templates.

The use of the developed complex is assumed when writing programs in the languages executed on Java Virtual Machine (JVM).

The developed library supports the following elements: - a set of graphic component templates (buttons, check boxes, switches, menu items, etc.)

- support for working with external graphical resources;

- support for multilingual interfaces;

- preserving the state of the graphical interface when restarting the program;

- support for the MVC Template (Model-View-Controller).

The developed software package is a high-level wrapper over the standard Swing graphics library.

Keywords: Graphical User Interface, Graphic Library, Widget, Kotlin, Swing.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.002 ТЗ	Шаблонізатор Java	4		
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Технічне задання			
	А4	ІАЛЦ. 045490.003 ТП	Шаблонізатор Java	3		
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ. 045490.004 ПЗ	Шаблонізатор Java	55		
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розробив		Мандрік М.В.			Шаблонізатор Java для автоматичної розробки графічного користувацького інтерфейсу Опис альбому	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Радченко К.О.					
Н. контроль		Клятченко Я.М.				НТУУ "КПІ ім. І. Сікорського", ФПМ, КВ-53	
Затвердив		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ. 045490.005 Д1	Шаблонізатор Java	1		
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Цикл роботи системи.			
			Схема зав’язків класів			
			бібліотеки JavaFX			
	А4	ІАЛЦ. 045490.006 Д2	Шаблонізатор Java	1		
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Схема зав’язків класів			
			розробленої бібліотеки			
	А4	ІАЛЦ. 045490.007 Д3	Шаблонізатор Java	1		
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Схема зав’язків класів			
			бібліотеки Swing			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ. 045490.008 Д4	Шаблонізатор Java	1		
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Схема роботи з			
			налаштуваннями			
			бібліотеки UML-діаграма			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.001 ОА	Арк. 4

ЗМІСТ

стор.

					ІАЛЦ.045490.002 ТЗ		
Зм.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Мандрік М.В.			Шаблонізатор Java для автоматичної розробки графічного користувацького інтерфейсу Технічне завдання	Літ.	Аркуш 1 / Аркушів 4
Перевір.		Радченко К.О.					
Н. контр.		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ КВ-53	
Затв.		Тарасенко В.П.					

1.НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування роботи – «Шаблонізатор Java для автоматичної розробки графічного користувацького інтерфейсу». Область застосування: розробка програмних компонентів з графічним інтерфейсом користувача.

2.ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені І. Сікорського».

3.ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є розробка бібліотеки графічних компонентів для мов JVM. Даний проект призначений спростити та пришвидшити процес розробки програм, що мають графічний інтерфейс користувача.

4. ДЖЕРЕЛА РОБОТИ

Джерелами роботи є дослідні статті та книги на тему розробки графічного інтерфейсу, статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги програмного комплексу:

- Надає набір шаблонів графічних компонентів, що можуть буть використані при подальшій розробці програм.
- Забезпечує підтримку багатомовного інтерфейсу.
- Забезпечує підтримку всіх мов, що виконуються на JVM.
- Надає можливість серіалізації графічного інтерфейсу.

5.2 Вимоги до програмного забезпечення:

- Операційна система Windows 10, Linux.
- Java Development Kit (JDK) 8.

5.3 Вимоги до апаратного забезпечення:

- Процесор MediaTek, Snapdragon, Kirin, Intel Atom;
- Оперативна пам’ять: 2 Гб.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.004 ПЗ	Шаблонізатор Java	55		
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Пояснювальна записка			
	А1	ІАЛЦ. 045490.005 Д1	Шаблонізатор Java	1		
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Цикл роботи системи.			
			Цикл роботи системи.			
			Схема зав’язків класів			
			бібліотеки JavaFX			
			Цикл роботи системи.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП		
Розробив		Мандрік М.В			Шаблонізатор Java для автоматичної розробки графічного користувацького Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Радченко К.О.					
Н. контроль		Клятченко Я.М.				КПІ ім Ігоря Сікорського, ФПМ, КВ-53	
Затвердив		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А1	ІАЛЦ. 045490.006 Д2	Шаблонізатор Java	1		
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Схема зав’язків класів			
			розробленої бібліотеки			
	А1	ІАЛЦ. 045490.007 Д3	Шаблонізатор Java	1		
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Схема зав’язків класів			
			бібліотеки Swing			
	А1	ІАЛЦ. 045490.008 Д4	Шаблонізатор Java	1	А	
			для автоматичної			
			розробки графічного			
			користувацького			
			інтерфейсу.			
			Схема роботи з			
			налаштуваннями			
			бібліотеки UML-діаграма			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			

ЗМІСТ

стор.

ДОДАТКИ

Додаток 1. Копії графічного матеріалу

ІАЛЦ.467200.005 Д1. Схема зав’язків класів бібліотеки JavaFx.

ІАЛЦ.467200.006 Д2. Схема зав’язків класів розробленої бібліотеки.

ІАЛЦ.467200.007 Д3. Схема зав’язків класів бібліотеки Swing.

ІАЛЦ.467200.008 Д4. Схема роботи з налаштуваннями бібліотеки UML-діаграма.

Додаток 2. Лістинг програми

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

GEOS – Graphic Environment Operating System — Операційна Система з Графічним Середовищем.

ГІК – графічний інтерфейс користувача.

JVM – Java Virtual Machine, набір структур даних та комп'ютерних програм, що виконують інші комп'ютерні програми та використовуючи модель віртуальної машини. JVM, як правило, використовує бінврний код Java, який генерується з вихідних кодів мови програмування Java. Віртуальна машина також використовується для виконання коду, згенерованого за допомогою інших мов програмування.

PARC – Xerox Palo Alto Research Center, науково-дослідницький центр, заснований, що був заснований у 1970 році.

Вікно – візуально відокремлена частина екрана у якій розташований інтерфейс користувача. Як правило вікна для програм мають прямокутну або квадратну форму. У вікні відображається ввід вивід даних і процесів, що відбуваються у програмі [3].

Інтерфейс – засіб взаємодії користувача з програмою або іншою інформаційною системою. Сукупність засобів для обробки та відбиття інформації [3].

ОС – операційна системи, базовий комплекс програм, що застосовується для управління апаратною складовою комп'ютерної системи або віртуальної машини. Операційна система керує обчислювальним процесом і забезпечує взаємодію комп’ютерної системи з користувачем. Як правило, до складу операційної системи входить та базовий набор прикладних програм.

JDK – Java Development Kit, комплект, що безкоштовно розповсюджується компанією Oracle для розробники програм мовою Java. Цей комплект включає в себе компілятор Java, стандартні бібліотеки

класів Java, приклади створених програм, документацію, різноманітні утиліти і виконавчу систему Java [4].

Фреймворк – інфраструктура програмних рішень для полегшення процесу розробку складних систем. Дану інфраструктуру можна вважати комплексною бібліотекою, яка має певні обмеження, що задають правила написання коду та створення структури проекту [3].

AWT – Abstract Window Toolkit.

SWT – Standard Widget Toolkit.

«Hotdog Stand» – схема використання кольорів у візуальній продукції яка спрощую сприйняття графічної інформації для людей з порушенням сприйняття кольорів. Основою цієї схеми є використання трьох кольорів: жовтого, червоного та чорного.

API – прикладний програмний інтерфейс, сукупність засобів та правил, що надають можливість взаємодіяти між собою окремим складникам програмного забезпечення або програмним та апаратним засобам забезпечення.

ВСТУП

Щодня на ринок програмного забезпечення випускається багато нових програм або їх оновлених версій. У сучасному світі кожна програма повинна мати інтерфейс взаємодії з користувачем. Складно уявити програму, вікно якої не має кнопки для закриття програми, кнопок для зміни параметрів роботи програми, та можливості зміни розміру вікна для комфортного використання програми. Як правило, розробники обирають використання саме графічного користувацького інтерфейсу для свого майбутнього проекту. Це обумовлено його простотою та зрозумілістю для користувача.

До сучасних графічних інтерфейсів існує багато вимог: інтерфейс повинен бути інтуїтивно зрозумілим для користувачів, мати необхідний набір інструментів для зручного використання елементів програми, повинен бути простим для написання, має забезпечувати мінімальний час відклику.

Використання шаблонів для створення елементів ГІК значно пришвидшує та спрощує розробку зручної для користувача програми. На сьогоднішній день існує багато бібліотек-шаблогізаторів серед яких можна побачити не мало таких, які спеціалізуються на графічних інтерфейсах. Їх різноманіття обумовлено тим, що кожний шаблонізатор графічного інтерфейсу створений лише для однієї мови програмування або одного сімейства мов. Також шаблонізатори графічного інтерфейсу відрізняються наборами інструментів, заданими шаблонами та реалізацією їх з програмного боку. Отже, зручним шаблонізатором виступає бібліотека, яка спрощує реалізацію великої кількості елементів графічного інтерфейсу, тобто зводить її до одного або лише декількох рядків коду.

Мови JVM використовуються для рішення різноманітних типів задач. В результаті цього виникає необхідність швидко реалізовувати зручний користувацький інтерфейс.

У родини мов JVM, до якої належать такі мови як Scala, Kotlin, Clojure, Jython, JRuby та інші, є можливість побудувати графічний інтерфейс для користувача. Недоліком написання графічного інтерфейсу вручну є великий об’єм коду для описання кожного з елементів та повтор коду для описання схожих елементів, що протирічить принципу «не повторюй сам себе». Для рішення цієї проблеми зручно використовувати бібліотеки з графічними інтерфейсами.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ, ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Користувацькі інтерфейси

Користувацький інтерфейс покликаний щоб забезпечити передачу інформації між користувачем-людиною і комп'ютерною системою. Будь-який користувацький інтерфейс складається з сукупності засобів та методів «спілкування» з користувачем [1]. До таких засобів та методів належать засоби виводу інформації з системи до користувача, засоби вводу інформації, команди, які користувач віддає у комп'ютерну систему, набір правил, який описують алгоритм дій користувача для досягнення необхідної мети при взаємодії з комп'ютерною програмою [2].

Для задоволення потреб різних користувачів було розроблено декілька видів користувацьких інтерфейсів. Існують такі типи користувацьких інтерфейсів:

- Візуальний
  - Текстовий
  - Графічний
- Тактильний
- Жестовий
- Голосовий
- Матеріальний

1.2. Графічний користувацький інтерфейс

Особливістю графічного інтерфейсу користувача (ГІК) є розміщення таких елементів інтерфейсу як меню, значки, іконки, кнопки, списки и т. д. на дисплеї пристрою, що використовується для взаємодії користувача та комп'ютерної системи. Всі елементи у такій системі візуалізуються за

допомогою графічних зображень.

Історично склалося так, що основним конкурентом ГІК був тип організації роботи з користувачем за допомогою командного рядка. Перевагою ГІК над інтерфейсом командного рядка є вільний доступ за допомогою миші, клавіатури, тачпаду або джойстика до будь-якого об’єкту на екрані пристрою та можливість маніпулювати ними. ГІК простий у користуванні та зрозумілий для користувача завдяки використання простих та зрозумілих метафор у дизайні елементів інтерфейсу.

Яскравим приклади ГІК слугують інтерфейси веб-сайтів, додатків для комп'ютерів і мобільних пристроїв, ОС, зокрема Windows і MacOS.

1.3. Історія

Історія розвитку ГІК почалася ще в середині двадцятого століття з наукових досліджень у університетах різних країн. В 1960-і роки Дагом Енгельбартом в науково-дослідному інституті Стенфорда в рамках дослідження був винайдений ГІК. Довгий час ГІК залишався виключно науковою розробкою і не потрапляв у комерційну сферу програмного забезпечення [5].

Перший ГІК був розроблений PARC ще в далеких 70-х. Ця розробка нову хвилю ері інновацій у комп’ютерній графіці та створила нові дисципліни серед яких і дизайн комп’ютерних програм та інтерфейсів [6].

Першим персональним комп’ютером у якому був використаний тоді ще нових для користувачів та розробників графічний інтерфейс був Xerox Alto. Він був створений в 1973 році. Цей комп’ютер не був комерційним продуктом і призначався переважно для наукових досліджень та розробок в університетах, інститутах та на виробництві.

1.4. Приклади операційних систем, що використовують ГІК

1.4.1. Xerox 8010 Star

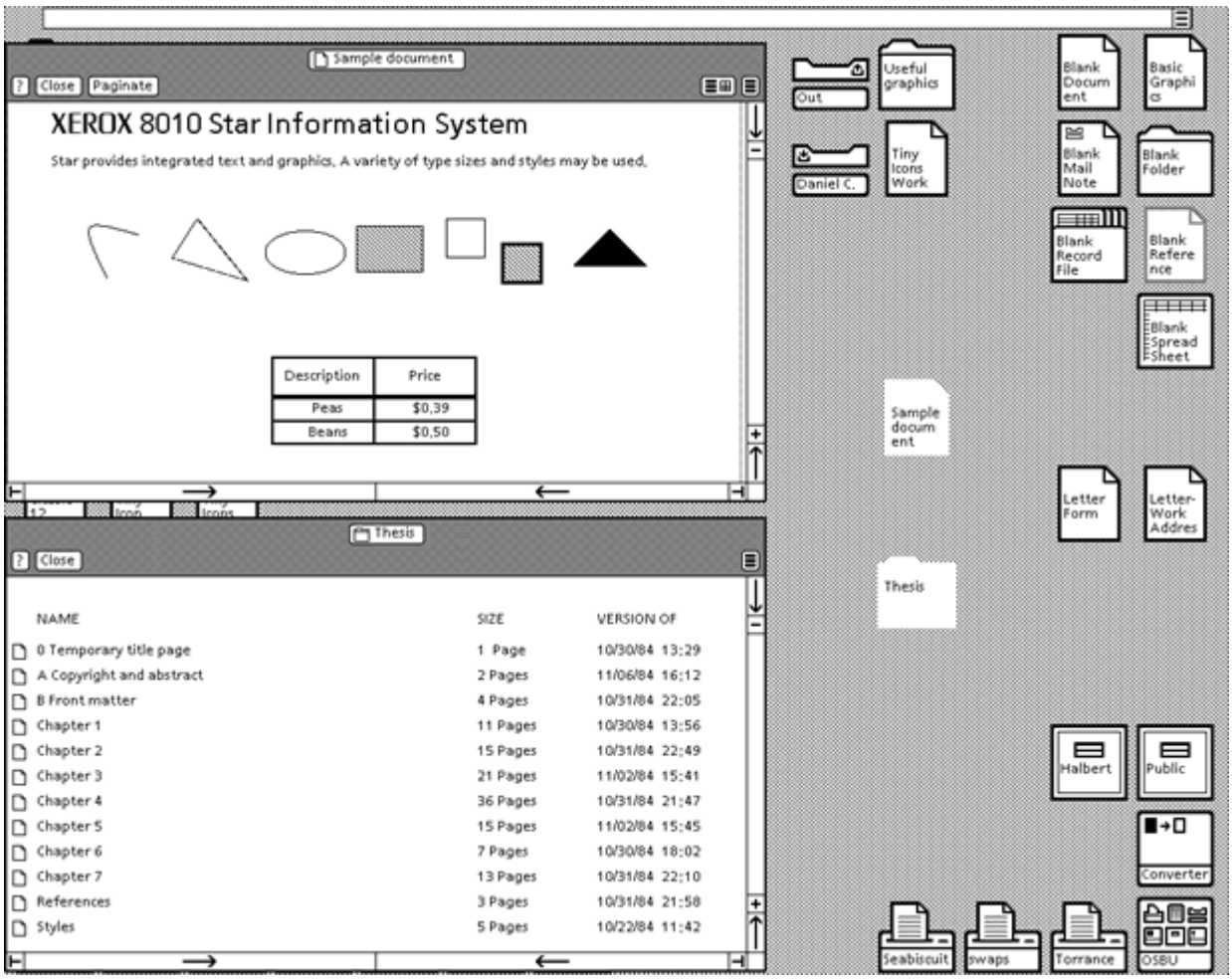


Рис. 1. ГІК Xerox 8010 Star

Xerox 8010 Star став першою системою, що була інтегрованим персональним комп’ютером. Приклад роботи Xerox 8010 Star наведений на рисунку 1. Ця система була розроблена у 1981 році. Вона мала програмні додатки і графічний інтерфейс. Комп’ютер був відомий як «The Xerox Star». Пізніше він був перейменований у «ViewPoint», а згодом – в «GlobalView» [6] .

1.4.2. Apple Lisa Office System 1

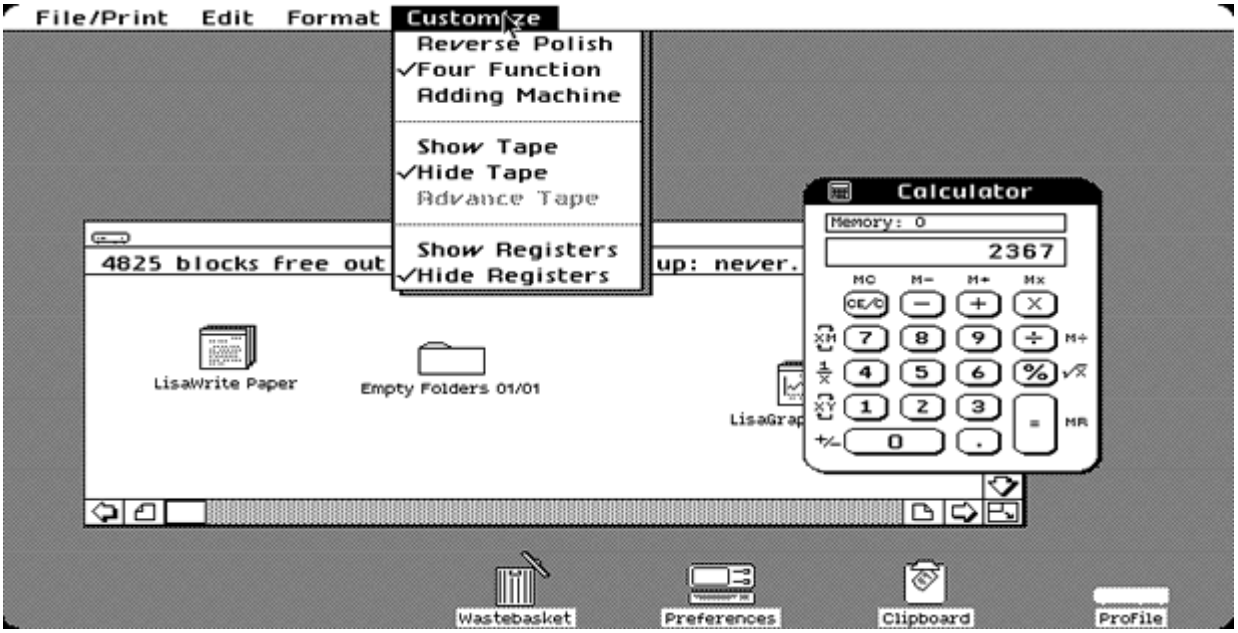


Рис. 2. ГІК Apple Lisa Office System 1

Система Apple Lisa Office System 1 була розроблена у 1983 році. Приклад роботи системи Apple Lisa Office System 1 наведений на рисунку 2. Розробниками цієї системи є компанія Apple, яка створила її спеціально для роботи з документами. Apple Lisa Office System 1 не витримала конкуренцію з системою Apple Macintosh на ринку програмного забезпечення. Причиною цього стала більш доступну ціну другої з систем. З часом були випущенні версії системи Lisa OS до Lisa OS 2 в 1983 году та Lisa OS 7/7 3.1 в 1984. Єдиними змінами в них у них була зміна лише самої системи роботи в той час як інтерфейс залишався незмінним. На той час він був одним із найзручніших для використання, дизайн його був сучасним та відповідав тогочасним потребам [7].

1.4.3. VisiCorp Visi On

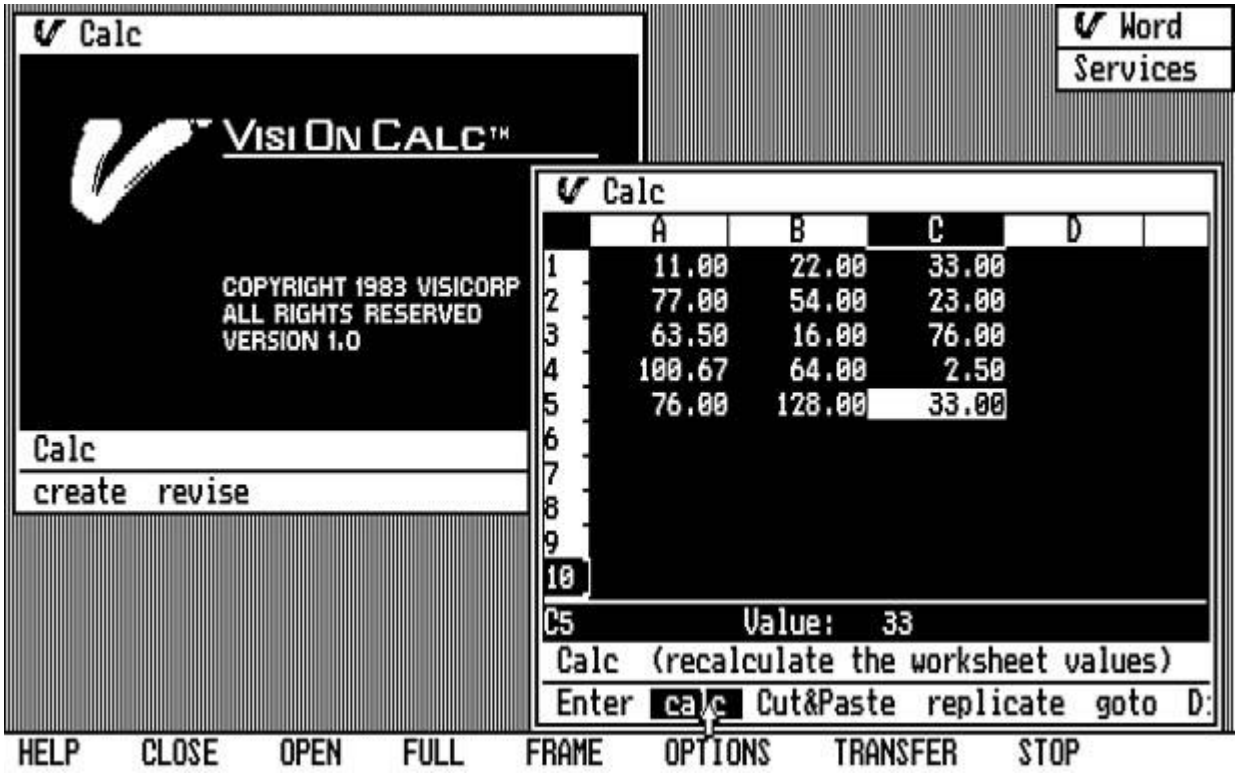


Рис . 3. ГІК VisiCorp Visi On.

VisiCorp Visi On був розроблено у 1984. Приклад роботи системи VisiCorp Visi On наведений на рисунку 3. Він був першим інтерфейсом, який розробили безпосередньо для IBM PC. Цільовою аудиторією для цієї системи були великі корпорації з великим капіталом. Як результат, за законами ринкової економіки, вартість продукції для виробництва була також великою. З засобів маніпуляції ця система мала лише мишку. В систему також був вбудований інсталятор та довідкова система. Однак дизайн ще був далекий від сучасного, у ньому були відсутні іконки та ще деякі елементи сучасного інтерфейсу [8].

1.4.4. Mac OS System 1.0

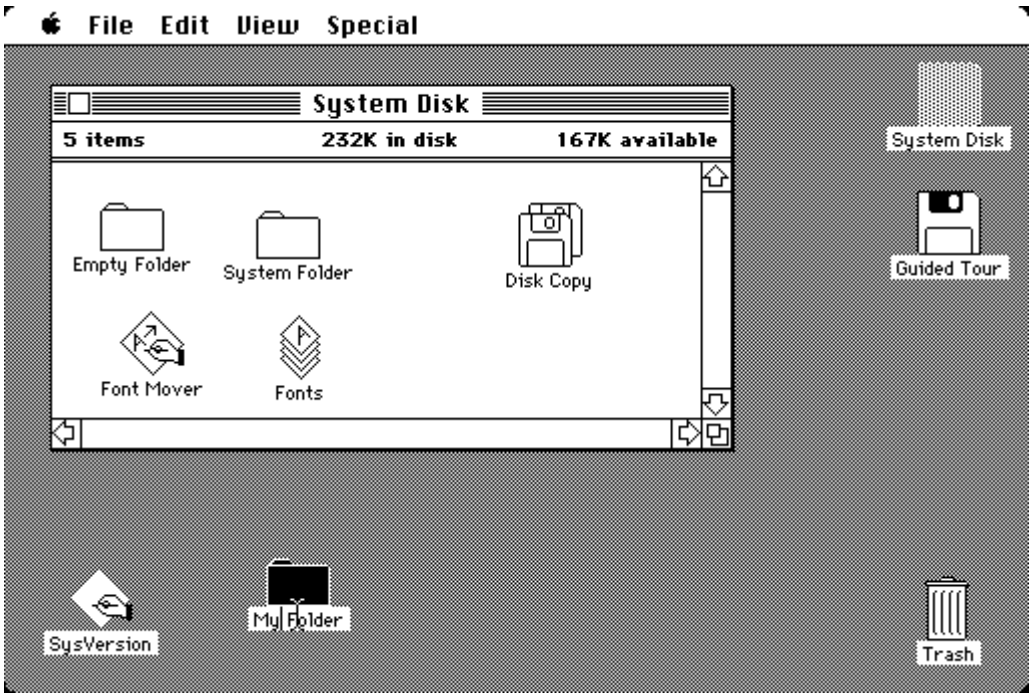


Рис. 4. ГІК Mac OS System 1.0.

Mac OS System 1.0 була випущена на ринок програмного забезпечення у 1984 році. Приклад роботи системи Mac OS System 1.0 наведений на рисунку 4. Ця система стала першою, яку було створено спеціально для комп'ютерів фірми Macintosh, при її розробці були враховані особливості апаратного забезпечення комп'ютерів цієї марки. Mac OS System 1.0 також включала в себе деякі елементи сучасних операційних систем. Так, наприклад, вона вже мала іконки як і ОС, якими зараз ми користуємось. Основою цієї системи став віконний принцип, який і лежить у основу усіх сучасних ОС. Маніпулювати положенням вікон на екрані можна було за допомогою миші. Для переміщення файлів та папок на нове місце користувач мав копіювати, що і створювало ще один екземпляр вибраного файлу, але вже на новому місці [7].

1.4.5. Graphic Environment Operating System

В 1986 році була розроблена ОС GEOS. GEOS є розробкою компанії, що займається створення програмного забезпечення, Berkeley Softworks. Пізніше ця компанія була перейменована в GeoWorks. Система GEOS була розроблена для одних з перших персональних комп'ютерів, комп'ютерів типу Commodore 64. До складу цієї ОС входили програма для малювання geoPaint та текстовий редактор geoWrite, що стало її відмінною рисою від всіх інших тогочасних ОС. geoPaint та geoWrite стали тогочасними аналогами програм Paint та Word і цим самим вивели GEOS на новий вищий рівень у порівнянні з іншими існуючими тоді аналогами [9].

1.4.6. Windows 3.1

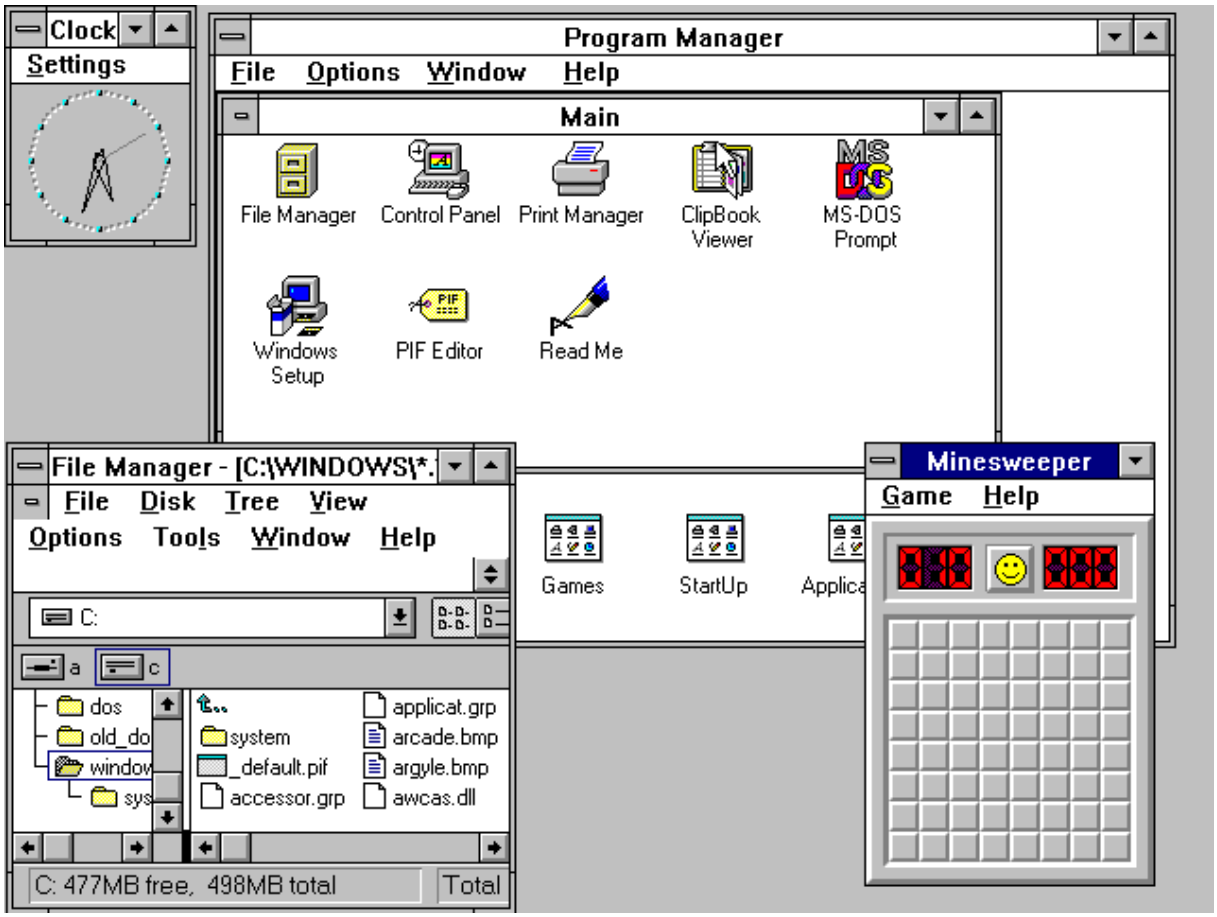


Рис. 5. ГІК Windows 3.1.

У Windows 3.1, який був створений у 1992 році, були включені шрифти типу TrueType. Саме цю поставило систему Windows на місце основної видавничої платформи. Раніше такі шрифти були доступні лише у одній версії Windows, і то з використанням програмного забезпеченням від іншої компанії, що було не зручним для користувачів. «Hotdog Stand» також ввійшов до цієї версії системи. «Hotdog Stand» мав яскраві відтінки різних кольорів, але найбільша увага приділялася відтінкам чорного, жовтого та червоного. Саме така схема була створена щоб полегшити сприйняття текстової та графічної інформації для людей, які мають порушення у сприйнятті кольорів [10]. Приклад роботи цієї системи Windows 3.1 наведений на рисунку 5.

1.4.7. Windows 95

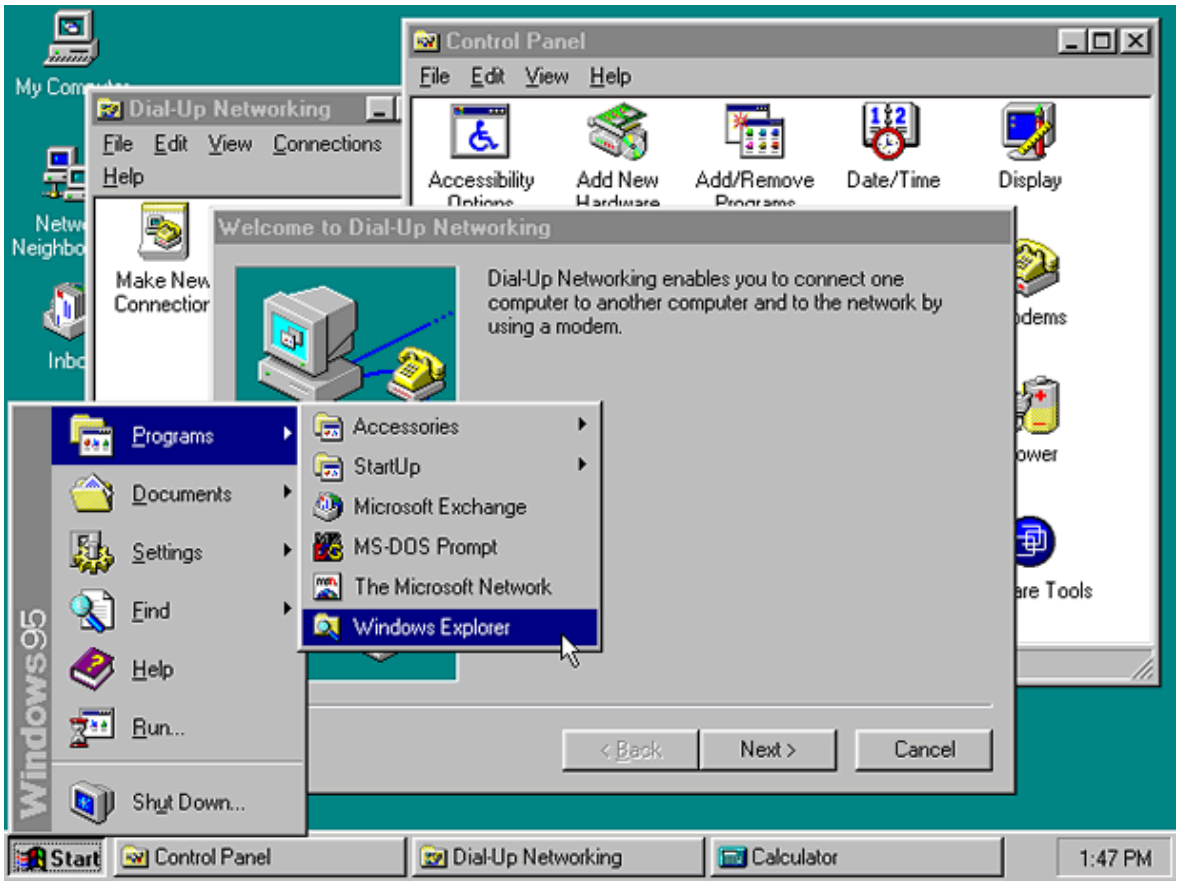


Рис. 6. ГІК Windows 95

У Windows 95, який було випущено у 1995 році, було повність змінено інтерфейс користувача. Приклад роботи системи Windows 95 наведений на рисунку 6. Це була перша версія Windows де у кутку кожного вікна з'явилась з хрестиком для закриття. Також були додані елементи керування серед яких доступно, недоступно, вибрано, відмічене та інші, а також були додані різні стани іконок. Відмінною рисою цієї версії Windows стала поява кнопки «Пуск». Ця кнопка давала доступ до вимикання комп'ютера, перезапуску системи, доступ до деяких програм, системи пошуку та довідки [10].

1.4.8. Windows 98

У версії Windows 98 1998 року випуску стиль іконок нагадує Windows 95. Основною відмінною рисою системи є можливість використовувати більше ніж 256 кольорів для відображення графічного інтерфейсу. У цій версії повністю був змінений Windows Explorer і уперше там з'явився «Active Desktop». Також у Windows 98 були присутні вже абсолютно всі риси, що притаманні й сучасним ОС з ГІК. До складу системи входили програми для роботи з різними типами файлів, зокрема з документами, таблицями, малюнками, презентаціями та інше [10].

1.5. Бібліотеки

Бібліотекою в програмуванні прийнято називати підмножину програм та об'єктів які використовують для розробки програмного забезпечення в цілому або деяких його елементів.

З боку ОС та прикладного програмного забезпечення бібліотеки можна поділити на динамічні та статистичні. Вперше термін «бібліотека програм» було використано Уілкс М., Уіллер Д., ГіллС. в якості одного з способів організації обчислень на комп'ютері [2].

1.5.1. Види бібліотек

Динамічною бібліотекою називають окремі файли, що передають програмі комплект виконуваних функцій. Ці функції завантажуються на етапі виконання при зверненні програми до ОС у момент заявки на виконання функції з бібліотеки. Якщо необхідна бібліотека вже завантажена в оперативну пам'ять, програма використовує завантажену копію бібліотеки. Такий підхід дозволяє зекономити час і пам'ять, оскільки декілька програм використовують одну копію бібліотеки, вже завантажену в пам'ять. Недоліками таких бібліотек є можливе руйнування API, конфлікт версій динамічних бібліотек, доступ однакових функцій за однаковими адресами до різних процесів однієї або різних програм.

Статичною бібліотекою називають файл у якому знаходиться вихідний код або об’єктний файл. Такий файл створений для вставки в програму на етапі компонування. Бібліотеки, що в розповсюджуються у вигляді вихідного коду в об’єктні файли перетворюються за допомогою компілятора. Потім об’єктні файли бібліотек та об’єктні файли програми з’єднує компонувач в один виконуваний файл.

Бібліотеки що розповсюджуються у вигляді об’єктних файлів зразу готові для компонування. Під час створення виконуваного файлу компонувач об’єдную об’єктні файли бібліотек і об’єктні файли програми у одне ціле [11].

Перевагою статичних бібліотек є те, що всі необхідні функції розміщенні в одному файлі. Саме через це для бібліотеки, що розроблена обран саме цей тип.

Недоліком статичних бібліотек є те, що файл, який використовується, займає багато місця на диску та в пам’яті. Також недоліком є те, що при знаходженні хоча б однієї помилки в бібліотеці необхідно виконувати повторну зборку програми.

1.5.2. Шаблонізатор

Шаблонізатором називають такий продукт програмного забезпечення, що дозволяє використовувати шаблони різних типів для генерації кінцевих сторінок програм та сайтів або їх елементів. Окреме представлення даних від виконуваного коду є метою, з якою зазвичай використовують шаблонізатори у процесі розробки програм. Також шаблонізатори часто використовують для забезпечення паралельної роботи програміста, що розробляє апаратну частину програми, та програміста-дизайнера, що розробляє графічну складову проекту, стиль та фінальний вигляд програмного продукту. Як правило використання шаблонізаторів підвищую читабельність коду [12].

У родині мов програмування JVM існують шаблонізатори для розробки різних інтерфейсів, серед них: Apache Velocity, FreeMarker, Histone, Thymeleaf.

1.6. Обґрунтування теми дипломного проекту

Темою дипломного проекту є створення бібліотеки для мов родини JVM з набором шаблонів для розробки графічного користувацького інтерфейсу. Створений шаблонізатор є високорівневою обгорткою над існуючими бібліотеками. Розроблена бібліотека є зручною для використання у невеликих проектах з простим графічним інтерфейсом, що складається зі стандартних елементів. При використанні цієї бібліотеки створення будь-якого графічного елементу зводиться до мінімальної кількості коду, що спрощує його написання.

Бібліотека включає в себе шаблони для розробки кнопок, різноманітних списків, календаря, рамок, іконок, радіо-кнопок, кнопок-перемикачів, вкладок та інших компонентів.

Так, як мови родини JVM активно використовуються у навчальних

та виробничих цілях, існує необхідність у швидкому та зручному створенні хоча б елементарних графічних інтерфейсів. Прості інтерфейси зазвичай використовуються у невеликих проектах або на проміжних етапах у створенні проектів коли немає необхідності у складності та унікальності інтерфейсу, як у фінальній версії продукту.

Перевагою розробленої бібліотеки є простота та наочність її використання. Також вона не потребує залучення багатьох ресурсів у проект, це дозволяє зайвий раз не ускладнювати та не загромаджувати проект. До переваг розробленої бібліотеки можна віднести також багатомовність: програма, що розробляється з використанням даної бібліотеки перекладає весь текст, що демонструється користувачу, на мову, що є основною для проекту на даний момент за наявності такої мови у словнику.

Недоліком бібліотеки є неможливість створювати елементи з різним зовнішнім виглядом, наприклад кнопки різної форми або вікна програми з різним дизайном. Такий недолік є побічним ефектом простоти, компактності та лаконічності розробленої бібліотеки.

2. РОЗРОБКА ЕЛЕМЕНТІВ БІБЛІОТЕКИ

2.1. Загальна структура системи

Розроблений програмний комплекс є системою, що складається з трьох основних пакетів:

- пакет settings;
- пакет gui;
- пакет handler.

2.1.1. Пакет settings

Пакет settings – пакет, що відповідає за збереження, встановлення користувацьких налаштувань в межах проекту та доступ до них. Пакет відповідає за реєстрацію компонентів графічного інтерфейсу в проекті. При реєстрації компонента в системі модуль проводить перевірку компонента на унікальність.

Також пакет відповідає за роботу з зовнішніми ресурсами, зокрема піктограмами та файлами, що відповідають за підтримку багатомовного інтерфейсу (словників).

У пакеті settings реалізовано UML-діаграма яка наведена у додатку «Схема роботи з налаштуваннями бібліотеки UML-діаграма».

2.1.2. Пакет handler

Пакет Handler забезпечує зручну роботу з налаштуваннями різних типів.

Класи пакету handler:

- BooleanPropertyHandler.kt – відповідає за роботу з параметрами, що являються логічними змінними.

- IntPropertyHandler.kt – відповідає за роботу з параметрами, що являються числовими змінними.
- PropertyHandler.kt – відповідає за абстрактний інтерфейс, що надає методи для роботи з налаштуваннями.
- StringPropertyHandler.kt – відповідає за роботу з параметрами, що являються строковими змінними.
- LanguagePropertyHandler.kt – забезпечує підтримку багатомовного інтерфейсу.

2.1.3. Пакет gui

Пакет gui представляє собою набір шаблонів графічних компонентів.

Класи gui:

- LButton.kt – шаблон, що відповідає за створення кнопки;
- LCalendar.kt – шаблон, що відповідає за створення календаря;
  - updatePanel – функція, що обновлює панель;
  - genDayBar – функція, що відповідає за створення заголовока календаря з іменами днів тижня;
  - genToolbar – функція, що відповідає за створення панелі інструментів. Ці інструменти забезпечують навігацію у календарі за місяцями та роками;
  - genView – функція, що відповідає за відображення поточного місяця у вигляді таблиці.
- LCheckBox.kt - шаблон, що відповідає за створення кнопки-перемикача;
- LComboBox.kt - шаблон, що відповідає за створення випадаючий список;
  - setItems - функція, що надає можливість замінити елементи випадаючого списку.

- LEmptyLabel.kt - шаблон, що відповідає за створення порожнього компонента;
- LFrame.kt - шаблон, що відповідає за створення рамки;
    - componentMoved – функція, що зберігає поточну позицію вікна програми;
    - componentResized - функція, що зберігає поточний розмір вікна програми;
    - restoreSize – функція, відтворює розміри та положення вікна на основі попередньо серіалізованих налаштувань, призначений для відтворення ГІК при запуску проекту;
- LIconButton.kt - шаблон, що відповідає за створення іконки;
- LLabel.kt – шаблон, що відповідає за створення текстового напису;
- LLangComboBox.kt - шаблон, що відповідає за створення випадаючого списку з мовами інтерфейсу;
- LRadioButton.kt - шаблон, що відповідає за створення радіо-кнопки;
- LStateCheckBox.kt - шаблон, що відповідає за створення статичної кнопки-перемикача;
- LStringComboBox.kt – шаблон для створення випадаючого списку з елементами строками;
- LTabPane.kt - шаблон, що відповідає за створення вкладок у вікні однієї програми;
    - addTab – функція, що відповідає за додання нових вкладок;
    - restoreSelected - функція, відтворює вкладки програми на основі попередньо серіалізованих налаштувань, призначений для відтворення ГІК при запуску проекту;
- LWrapper.kt - надає можливість доступу добудь-якого об’єкту з довільного місця у проекті.

2.2. Реалізація розроблених елементів

Кожен з шаблонів реалізує окремий елемент графічного інтерфейсу. Для розміщення кожного з шаблонів виділено у бібліотеці окремий файл. Шаблон за своєю суттю є класом, що зводить створення будь-якого з компонентів інтерфейсу передбачених бібліотекою до мінімальної кількості коду.

Шаблонізатор є високорівневою обгорткою над вже існуючими бібліотеками, як результат, створенні шаблонів задіяні вже існуючі бібліотеки JVM та їх функції.

2.2.1. Створення кнопки

Розроблена бібліотека дозволяє за шаблоном створити кнопку з будь-яким текстом у середині. У одному вікні можливо створити одну, дві або більше кнопок одночасно.

Для створення кнопки необхідно вказати, що створюється саме кнопка та її текст. Текст вказується у полі «name». Виклик кнопки з текстом «Example of button» виглядає наступним чином:

LButton(name = “Example of button”).

За замовченням всі кнопки створюються у один рядок. Щоб розмістити кнопки у декілька рядків або у колонку їх слід створювати в різних панелях.

2.2.2. Створення календаря

За допомогою розробленої бібліотеки можна створити календар. На панелі календаря є кнопки що дозволяють змінювати поточний місяць та рік. Кнопка «Today» виконую повернення до поточної дати.

Створення календаря відбувається за допомогою команди:

LCalendar().

При відкриття календаря автоматично виділена поточна дата.

Нижче наведений частина коду, з алгоритмом реалізації календаря:

```
init {
        Settings.register(id, this)
        if (load) {
            try {
                calendar.set(Calendar.YEAR, Settings.int["$id#year"])
                calendar.set(Calendar.MONTH,
    Settings.int["$id#month"])
                calendar.set(Calendar.DAY_OF_MONTH,
    Settings.int["$id#day"])
            } catch (ignored: NullPointerException) {
                calendar.timeInMillis = System.currentTimeMillis()
            }
        } else
            calendar.timeInMillis = System.currentTimeMillis()
        add(panel)
        panel.layout = BoxLayout(panel, BoxLayout.Y_AXIS)
        updatePanel()
    }
```

2.2.3. Створення кнопки-перемикача

Бібліотека дає можливість створювати кнопки-перемикачі. Кнопка такого типу має два положення: «вибрано» та «не вибрано». Положення, у яке встановлена будь-яка з створених кнопок не впливає на жодну іншу, таким чином обраною може буди одна, декілька, або усі запропоновані кнопки або може бути не обрана жодна з них.

Створити кнопку-перемикач з текстом «Example of CheckBox» можна за допомогою команди:

LCheckBox(name = “Example of CheckBox”).

При створенні кнопки-перемикача автоматично встановлюється стан «не вибрано». Стан кнопки можна змінити кліканням на неї мишкою.

При перезапуску програми кожна кнопка-перемикач автоматично встановлюється у положення «не вибрано».

2.2.4. Створення текстового напису

За допомогою бібліотеки можна створити текстовий напис у вікні програми. Також її можна вважати кнопкою без можливості взаємодії з нею.

Створити текстовий напис «Example of Label» можна за допомогою команди:

LLabel(name = “Example of Label”).

За замовченням всі текстові написи створюються підряд у один рядок. Щоб розмістити текстові написи у декілька рядків або у колонку їх слід створювати в різних панелях.

2.2.5. Створення випадаючого списку

Розроблена бібліотека має шаблон для створення випадаючого списку. Випадаючий список дає можливість вибрати один із вкладених у нього варіантів. За замовченням обраним вважається перший із запропонованих варіантів. Щоб змінити обраний варіант треба розгорнути список усіх варіантів, натиснути на нього мишкою та обрати будь-який інший варіант натисканням на нього мошкою. У список може бути вкладено один, два або більше елементів.

Створити випадаючий список з назвою «Example of ComboBox» та з трьома можливими значеннями «first element», «second element» та «third element» можна за допомогою команди:

LComboBox(name = “Example of ComboBox”, Arrays.asList(“first element”, “second element”, “third element”).toTypedArray()).

Нижче наведений частина коду, з алгоритмом реалізації випадаючого списку:

```
init {
        Settings.register(id, this)
        val realIndex = if (Settings.int.getOrDefault(id, index) >=
    values.size) 0 else index
        Settings.int[id] = realIndex
        selectedIndex = realIndex
        addActionListener {
           Settings.int[id] = selectedIndex
        }
     }
```

Елементами випадаючого списку можуть бути будь-якими об’єктами, ними може виступати текст, картинка або інша кнопка.

2.2.6. Створення меню з мовами

Бібліотека надає можливість створити випадаюче меню з можливістю вибору мови інтерфейсу програми. До можливих варіантів такого списку належать всі мови, словники яких додані до програми.

Створити випадаючий список з мовами інтерфейсу під назвою «Example of language ComboBox» можна за допомогою команди: LLangComboBox(name = “Example of language ComboBox”).

За замовченням обраною буде перша з завантажених мов. Змінити обрану мову можна таким чином: натиснути на випадаючий список, з запропонованих мов обрати необхідну і натиснути на неї. Таким чином

мова інтерфейсу проекту зміниться.

Реалізація алгоритму створення іконки виглядає наступним чином:

```
init {
        Settings.register(id, this)
        selectedItem = Settings.lang.getActiveLang()
        addActionListener {
           Settings.lang.setActiveLang(selectedItem as String)
        }
```

2.2.7. Створення радіо-кнопки

До шаблонів розробленої бібліотеки входить шаблон для створення радіо-кнопки. Радіо-кнопка – це кнопка, що має два положення: «вибрано» та «не вибрано». За замовченням створена кнопка знаходиться у положенні «не вибрано». При створених декількох радіо-кнопках можна встановити в положення «вибрано» жодну з них або лише одну. За замовченням при створенні кнопок жодна з них знаходиться у положенні «не вибрано».

Створити радіо-кнопку з назвою «Example of RadioButton» можна за допомогою команди:

```
LRadioButton(name = “Example of RadioButton”).
```

2.2.8. Створення статичної кнопки-перемикача

Розроблена бібліотека надає можливість створювати статичні кнопки-перемикачі. Їх особливість полягає у тому, що при перезапуску програми вини зберігають свій попередній стан. Тобто, якщо чек-бокс був встановлений у стан «вибрано», то при перезапуску він збереже цей стан. У всіх інших аспектах статичні кнопки-перемикачі не відрізняються від звичайних кнопок-перемикачів.

Створити статичну кнопку-перемикач з написом «Example of State

ChecBox» можна за допомогою команди:

LStateCheckBox(name = “Example of State ChecBox”) .

2.2.9. Створення іконки

За допомогою розробленої бібліотеки можна створити кнопку з іконкою, тобто малим зображенням. Вибрати іконку можна з тих, що вже додані до бібліотеки або завантажити у бібліотеку нові.

Створити кнопку з іконкою load.png можна за допомогою команди: LIconButton(“load.png”).

Реалізація алгоритму створення іконки наступним чином:

```
init {
        val file = Paths.get(Settings.resourceDir, path).toFile()
        val realName = name ?: file.nameWithoutExtension
            .map { if (it in 'A'..'Z') " ${it.toLowerCase()}" else "$it"}
            .joinToString("")
            .capitalize()
        val realId = id ?: "__ibtn__$realName"
        Settings.register(realId, this)
        if (action != null)
           addActionListener(action)
        if (file.exists()) {
           icon = ImageIcon(file.absolutePath)
           preferredSize = Dimension(icon.iconWidth, icon.iconHeight)
           maximumSize = preferredSize
           minimumSize = preferredSize
        } else
           text = Settings.lang[realName]
    }
```

2.2.10. Створення вікна

За допомогою бібліотеки можна створити вікно для програми. Вікно має кнопки «хрестик» для закриття вікна, кнопку для згортання вікна та кнопку для відкриття на весь екран. Поруч з кнопками для закриття, згортання вікна та розкриття його на весь екран відображається назва вікна. Перед закриттям вікна автоматично генерується уточнююче питання до користувача «закрити?».

За замовченням розмір створеного вікна обирається, так, щоб вмістити в себе всі об’єкти, що програма виводить на екран. Змінити розміри вікна можна потягнувши за стінки вікна або за його кути. Нижче наведений код з реалізацією алгоритму створення вікна:

```
init {
        Settings.register(id, this)
        if (panel != null)
           contentPane = panel
        isVisible = true
        defaultCloseOperation =
    WindowConstants.DO_NOTHING_ON_CLOSE
        addWindowListener(this)
        addComponentListener(this)
      }
```

За визнаення розмірів вікна та його положення на екрані відповідає наступна функція:

```
fun restoreSize() {
                val height =
            Settings.int.getOrDefault("$id#height", height)
                val width =
            Settings.int.getOrDefault("$id#width", width)
```

```
            val x = Settings.int.getOrDefault("$id#X", 100)
            val y = Settings.int.getOrDefault("$id#Y", 100)
            setBounds(x, y, width, height)
        }
```

Створити вікно з назвою «test» для програми можна за допомогою команди:

LFrame(name = “test”, JPanel()).

2.2.1. Створення текстового випадаючого списку

За допомогою розробленої бібліотеки можна створити випадаючий список, елементами якого будуть лише текстові поля.

Випадаючий список дає можливість вибрати один із вкладених у нього варіантів. За замовченням обраним відображається перший варіант зі списку. Обрати інший варінт можна розкривши список і клікнувши на будь-який фнший варіант. У список може бути вкладено один, два або більше елементів. Реалізація алгоритму створення текстового випадаючого списку виглядає наступним чином:

```
init {
        Settings.register(id, this)
        val realIndex = if (index >= values.size) 0 else index
        selectedIndex = Settings.int.getOrPut(id, realIndex)
        addActionListener {
            Settings.int[id] = selectedIndex
        }
    }
```

Створити текстовий випадаючий список з назвою «Example of StringComboBox» та з трьома можливими значеннями «first element», «second element» та «third element» можна за допомогою команди:

LStringComboBox(name = “Example of String ComboBox”, Arrays.asList(“first element”, “second element”, “third element”).toTypedArray()).

2.2.12. Створення вкладок

Розроблена бібліотека має інструменти для створення вкладок у вікні програми. Одночасно у вікні однієї програми може бути створена одна, дві або декілька вкладок. При відновленні роботи вікна програми, що було раніше закрито, всі вкладки, їх порядок розташування, а також поточна активна вкладка залишаються не змінними. При запуску програми відкриває їх у новому, щойно створеному вікні.

Реалізація алгоритму створення вкладки у вікні програми виглядає наступним чином:

```
init {
        Settings.register(id, this)
        if (id.isNotBlank())
            addChangeListener {
                Settings.int["$id#index"] = selectedIndex
            }
    }
```

Створити текстовий вкладку з назвою «Example of TabPane» можна за допомогою команди:

LTabPane (name = “Example of TabPane”).

2.3. Висновки до розділу

Розроблений проект складається з основних пакетів які розділені на пакети за принципом функцій, які вони реалізують. До складу проекту входить пакет з налаштуваннями, пакет, що відповідає за роботу зі змінними різних типів та набір шаблонів для створення компонентів інтерфейсу користувача. Взаємодію класів бібліотеки можна побачити у додатку «Схема зв’язків класів розробленої бібліотеки ».

Створена бібліотека надає можливість розробникам, що працюють з мовами родини JVM, мінімізувати час та зусилля, які витрачаються на створення зручних компонентів ГІК. Функції бібліотеки дозволяють звести створення кнопок, міток, вкладок, випадаючих списків, кнопок-перемикачів, радіо-кнопок, вікон програми коду, що займає багато місця, до лише одного рядка.

Розроблена бібліотека забезпечує багатомовність інтерфейсу та збереження налаштувань програми при закритті і подальшому відновленні вікна. Бібліотека може працювати як з логічними змінними, так і з числовими і строковими типами змінних.

3. ТЕСТУВАННЯ БІБЛІОТЕКИ

3.1. Приклади створення елементів ГІК

Проведемо серію створення кнопок та кнопок-іконок:

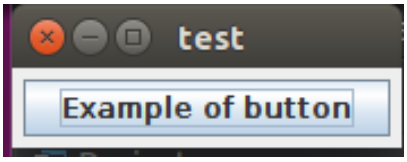


Рис. 7 Кнопка з текстом «Example of button»

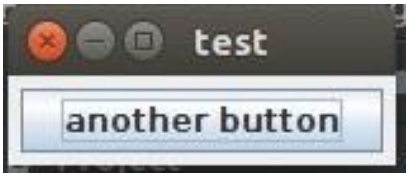


Рис. 8 Кнопка з текстом «another button»

Рис. 9 Кнопки «First button» та «Second button»

Рис. 11. Приклад кнопок-іконок

За допомогою бібліотеки можна кнопки прямокутної форми з текстом або іконкою всередині. Приклади створених кнопок наведені на рисунках 7-11. Для створення кнопок з текстом всередині необхідно використовувати функцію LButton, а для створення кнопок з іконкою – LiconButton.

Розмір кнопки визначається автоматично в залежності від об’єму тексу, що знаходиться всередині або розміру іконки. Також розмір кнопки може задаватися вручну програмістом. Кнопка по краях обмежена синьою лінією, а фоновий колір наповнення кнопки переливається від блакитного до синього створюючи тим самим ефект об’єму.

Проведемо серію тестів з створення та використання кнопок-перемикачів:

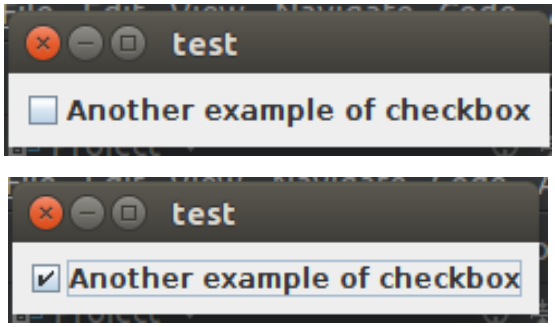


Рис. 12,13. Приклад роботи кнопки-перемикача

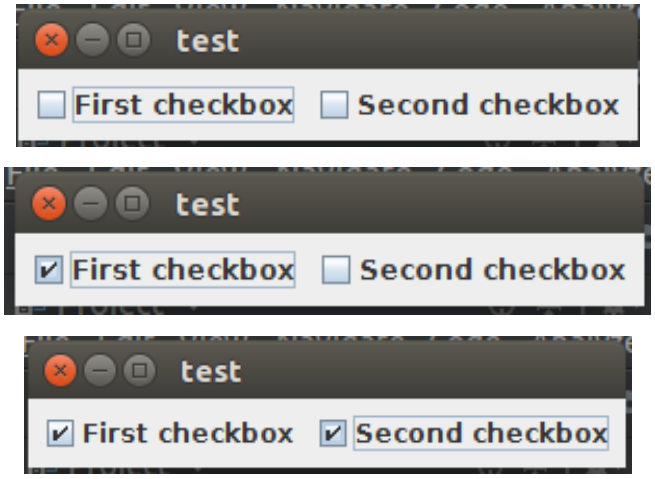


Рис. 14-16. Приклад одночасної роботи декількох кнопок-перемикачів

Рис. 17. Приклад створення статичної кнопки-перемикача

За допомогою бібліотеки можна створити кнопки-перемикачі значеннями яких можуть виступати як рядкові данні, так і чисельні або картинка (іконка). Приклади створення кнопок-перемикачів наведені на рисунках 12-17 . Кнопка-перемикач є елементом, що складається з поля у якому вказується текст або іконка кнопки-перемикача та з квадратика у якому відображається поточний стан. Пустий квадратик відповідаю стану «вибрано», а квадратик з галочкою у середині – стану «не вибрано».

Проведемо серію тестів з створення та використання текстового напису:

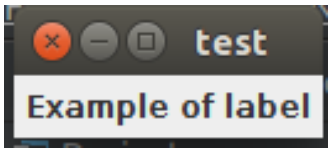


Рис. 18. Приклад створення текстового напису «Example of label»

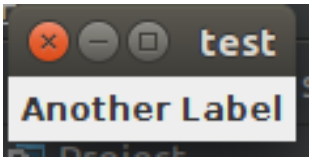


Рис. 19. Приклад створення текстового напису «Another Label»

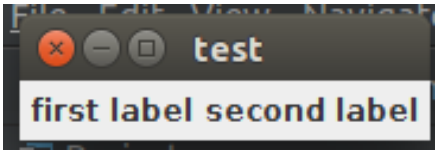


Рис. 20. Приклад двох поруч створених текстових написів у одному вікні

За допомогою бібліотеки можна створити текстовий напис. Приклади створених текстових написів наведені на рисунках 18-20. Текстовий напис складається лише з тексту, що і є її наповненням.

Візуально границі текстового напису не виділяються на фоновому тлі програми. Розмір текстового напису обирається відповідно до об’єму тексту, що знаходиться у ній або може бути заданим програмістом.

Проведемо серію тестів з створення та використання випадаючого списку та випадаючого списку з мовами інтерфейсу:

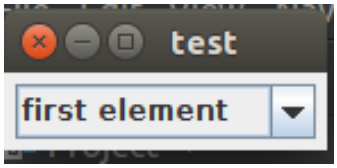


Рис. 21. Приклад випадаючого списку

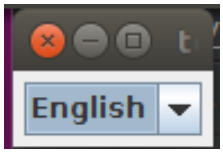


Рис. 22. Приклад випадаючого списку з вибором мови інтерфейсу

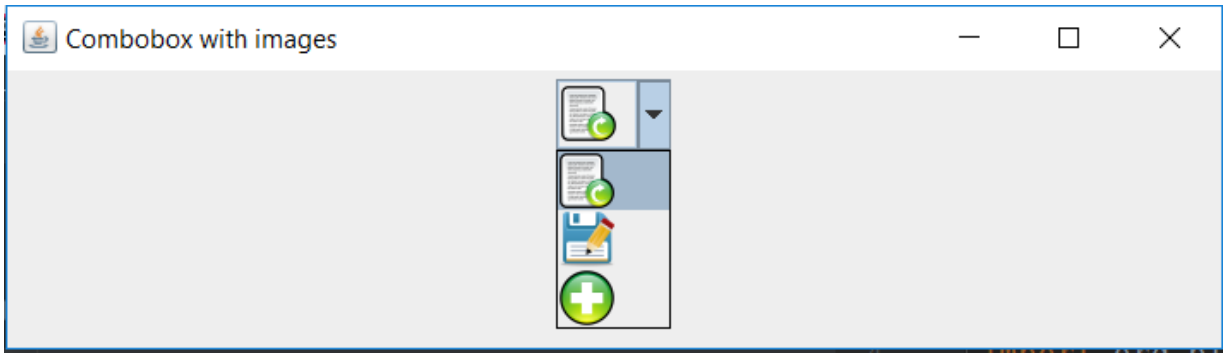


Рис. 23. Приклад випадаючого списку з елементами іконками

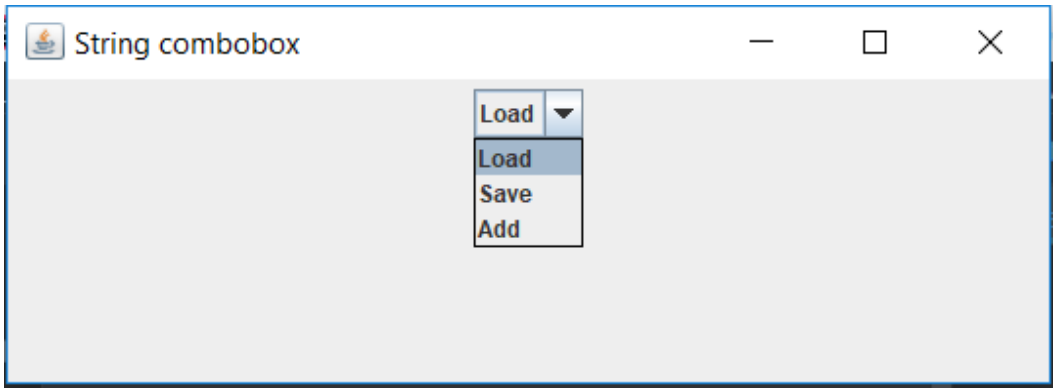


Рис. 24. Приклади розгорнутого випадаючого списку

За допомогою бібліотеки можна створити звичайний випадаючий список або випадаючий список з мовами для зміни мови інтерфейсу програми. Приклади створених випадаючих списків наведені на рисунках 21-24. Випадаючий список є прямокутною панеллю де у лівій частині знаходиться поточний обраний елемент, а у правій - стрілка «донизу».

За допомогою цієї стрілки можна розгорнути список. Змінити поточний обраний елемент можна кликнувши на будь-який інший елемент у розгорнутому списку елементів. Елементами звичайного випадаючого списку можуть виступати як рядкові змінні, так і графічні об’єкти (іконки).

Проведемо серію тестів з створення та використання радіо-кнопок:

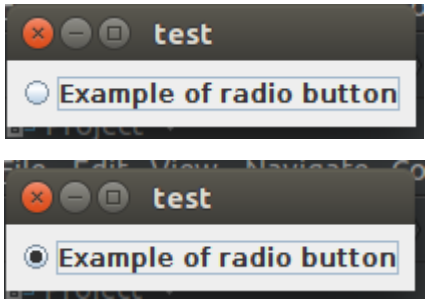


Рис. 25,26. Роботи радіо-кнопки «Example of radio button»

Рис. 27,28. Робота радіо-кнопки «Radiobutton»

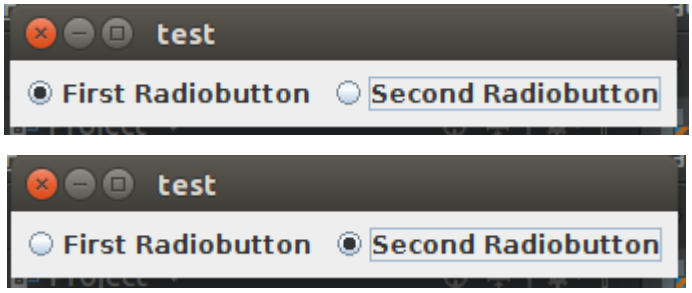
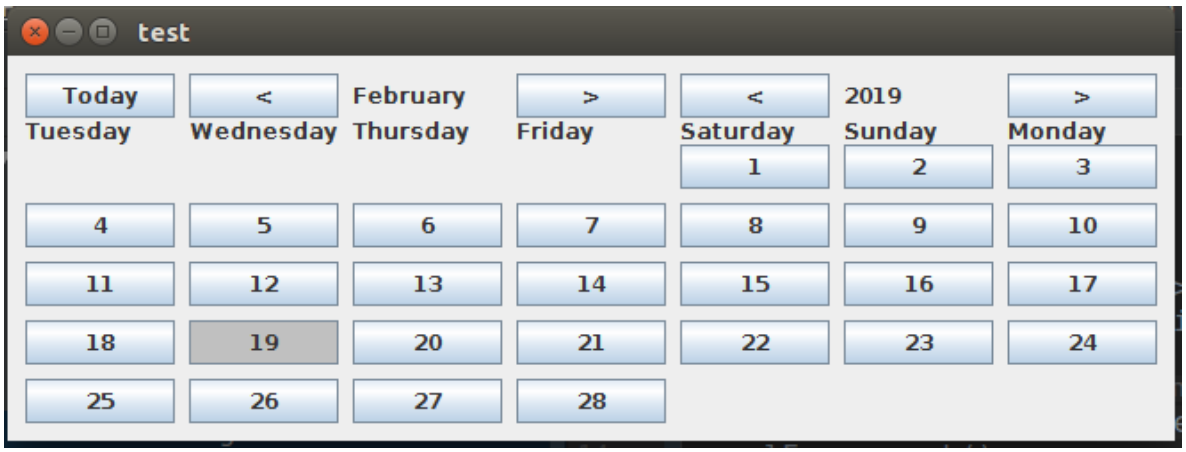


Рис. 29,30. Приклад одночасної роботи двох радіо-кнопок

За допомогою бібліотеки можна створювати радіо-кнопки. Приклади створення та роботи радіо-кнопок наведені на рисунках 25-30. Радіо-кнопка є елементом, який складається з правої частини – поля яке є текстовою змінною і кружечку у лівій частині вікна. У кружечку встановлюються стан, у якому знаходиться кнопка, пустий кружечок відповідає стану «не вибрано», а кружечок з чорною точкою у середині – стану «вибрано». Для створення радіо-кнопки застосовується функція LradioButton. Текст значення радіо-кнопки знаходитеся на тлі програми, а кружечок для зміни стану виділений лінією.

Проведемо серію тестів з створення та використання календаря:



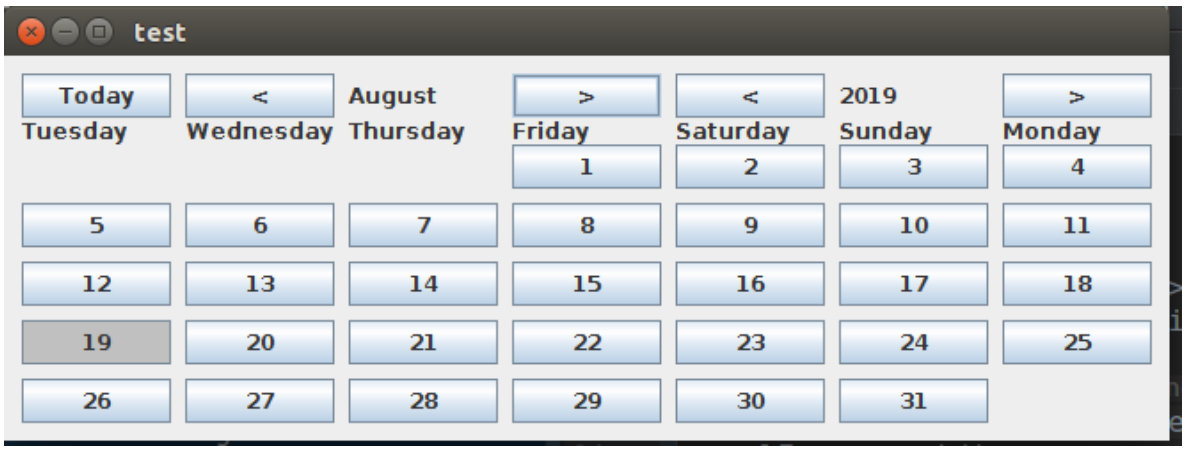


Рис. 31-33. Приклади згенерованого календарята його роботи

За допомогою бібліотеки можна створити календар. Приклади створення та використання календаря наведені на рисунках 31-33. За Календар складається з верхньої панелі курування, рядка з назвами днів тижня та поля, де відображаються дати.

Верхня панель календаря складається з наступних елементів: кнопка «Today», панелі місяця та панелі року. Кнопка «Today» переміщує каретку календаря на поточну дату з будь-якої іншої обраної дати. Панель місяця складається з двох кнопок-стрілок для зміни місяця та з назви місяця, що знаходиться між цими кнопками. Панель року складається з двох кнопок-стрілок для зміни року та з номера року, що знаходиться між цими кнопками. У полі, де відображаються дати, кнопки з датами розташовані по 7 у ряд, що відповідає дням тижня, як і підписано над кожною колонкою дат.

При зміні місяця чи року одразу генерується розкладка дат на поточний місяць. Активним залишається день с тим самим номером, що і був обраний попередньо, змінюється лише його день тижня і місяць чи рік відповідно.

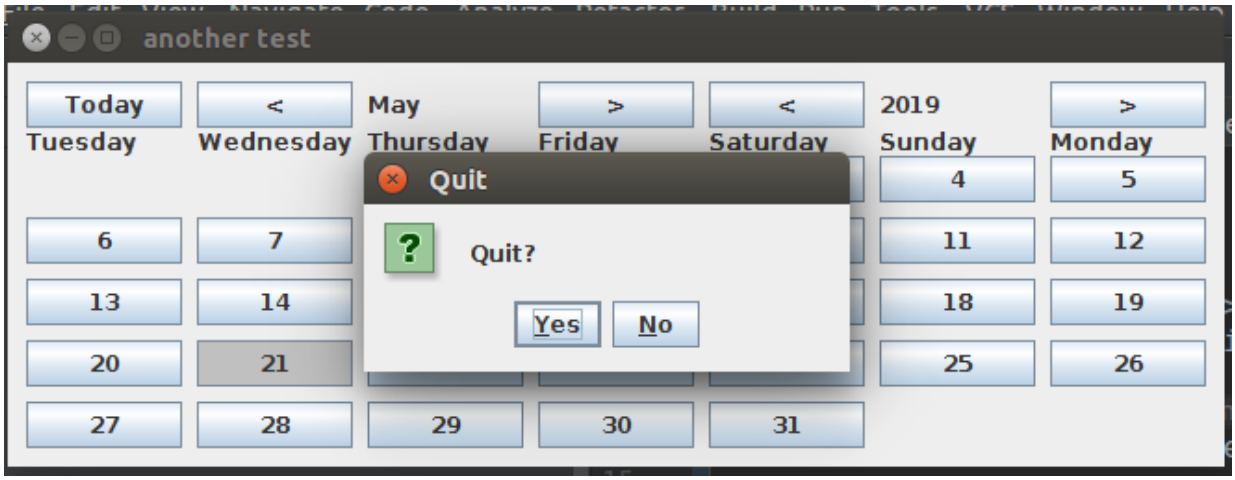


Рис. 34. Закриття вікна програми

Кожна тестова програма запускалась у вікні з рамкою. У лівому верхньому кутку рамки знаходяться кнопки «закрити», «згорнути» і «розгорнути на весь екран» або «згорнути до попереднього розміру». При закритті програми з'являється вікно з питанням «Quit?» та варіантами відповіді «Yes» та «No», яке зображено на рисунку 34. Вікно з запитанням можна також закрити за допомогою хрестика у його верхньому лівому кутку, у такому випадку сама програма не буде закритою.

Рамка для програми герується за допомогою функції LFrame.

4.1.Висновки до розділу

У бібліотеці знаходяться розроблені шаблони для створення багатьох елементів ГІК. За допомогою цих елементів можна легко створити простий та інтуїтивно зрозумілий для користувача інтерфейс. Всі розроблені шаблони створені у одному стилі, це спрощую візуальне сприйняття програми.

Шаблони розроблені для роботи з різними типами даних: строковими (використовуються у кнопках, текстових написах, випадаючих меню та в інших структурах), чисельними (використовуються у кнопки, випадаючих меню, кнопках перемикачах та в інших структурах),

зображеннями (використовуються у кнопках-іконках, кнопках-перемикачах).

Проведені серії тестів провіряють коректність створення елементів з шаблонів бібліотеки. У тестах показані створені елементи з різними параметрами та робота цих елементів, взаємодія однакових елементів один з одним.

4. ПОРІВНЯННЯ З АНАЛОГАМИ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

Розроблена бібліотека є високорівневою обгорткою над існуючими функціями у JVM та вже створеними бібліотеками. При створенні бібліотеки використані методи інших програмних продуктів для досягнення лаконічності у описанні кожного з елементів ГІК.

Щоб зрозуміти цінність розробленої бібліотеки та розглянути перспективи її розвитку необхідно проаналізувати бібліотеки які також спеціалізуються на створенні графічних інтерфейсів.

4.1 Порівняння з аналогами

Для оцінки ефективності слід порівняти створену бібліотеку для розробки ГІК з вже існуючими аналогами. Отже, розглянемо існуючі аналоги.

Потреба створювати графічний інтерфейс на мовах родини JVM у розробників програм була завжди. Не дивно, що створено багато бібліотек, які спеціалізуються саме на функціях та методах для створення компонентів ГІК.

Розглянемо та проаналізуємо переваги та недоліки декількох найпопулярніших бібліотек з шаблонами для створення ГІК.

Однією з перших спроб автоматизувати створення графічного інтерфейсу для мови JVM (зокрема Java) стала розроблена графічна бібліотека AWT [13]. Ця бібліотека використовує методи з бібліотек написаних на C, які, у свою чергу, створюють та використовують компоненти ОС.

Стратегія розробки цієї бібліотеки ГІК має перевагу у тому, що візуально така програма буде схожа на інші програми написані для даної ОС. Однак ця стратегія має і недолік - відмінності у шрифтах та інших

компонентах у різних ОС може зіпсувати зовнішній вигляд програми до невпізнанності.

До переваг цієї бібліотеки для розробки графічного інтерфейсу також можна віднести швидкість роботи програм з її використанням(вона працює дуже швидко), а також те, що вона є частиною JDK. До недоліків цієї бібліотеки належать те, що вона використовує нативні компоненти, і це накладає обмеження бо деякі з них працюють не на всіх платформах. AWT не має засобів для відображення іконок, створення таблиць, а також в AWT відсутні спливаючі підказки.

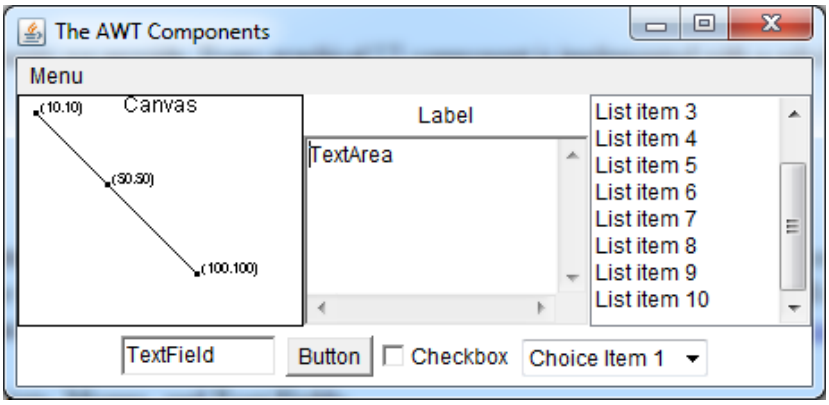


Рис. 35. Інтерфейс створений за допомогою AWT

Приклад програми з використанням бібліотеки AWT наведений на рисунку 35.

Трохи пізніше була розроблена бібліотека ГІК під назвою Swing. Елементи цієї бібліотеки вже повністю написані на мові програмування Java. Вся графіка у цій бібліотеці була виконана у 2D. Набор реалізованих компонентів був значно більшим, кращим і різноманітнішим ніж у AWT. Перелік класів наведені у додатку «Класи бібліотеки Swing» », назви класів співпадають з назвами компонентів.. Ця бібліотека отримала підтримку різних стилів і можливість створення нових компонентів х існуючих за допомогою наслідування. На основі цієї бібліотеки існує безліч багаторівневих обгорток, що демонструє її актуальність і популярність.

До переваг Swing належить те, що вона також є частиною JDK (а отже не потребує встановлення додаткових бібліотек). Існує багато книжок та мережевих ресурсів з докладними інструкціями з роботи з цією бібліотекою.

Недоліком цієї бібліотеки є те, що програми з її використанням часто тормозять та виснуть, особливо програми з використанням перших версій Swing. Для складних інтерфейсів робота з менеджером компонування є складною і не зручною, це значно ускладнює процес розробки програми. Swing є одним з найпопулярніших фреймворків серед розробників, що працюють з мовами JVM для створення ГІК.

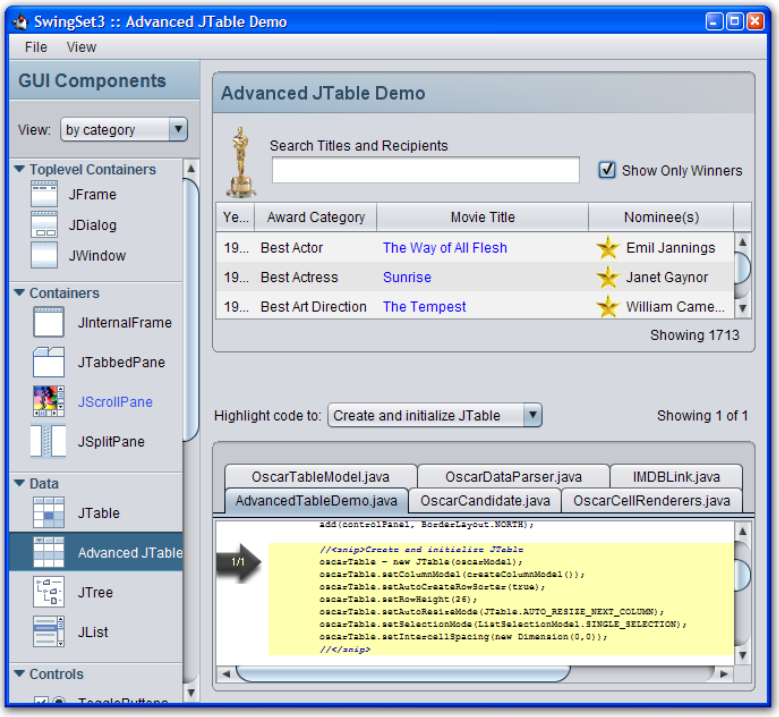


Рис. 36. Інтерфейс створений за допомогою Swing

Приклад програми з використанням бібліотеки Swing наведений на рисунку 36.

SWT – бібліотека для розробки ГІК, що була створена у часи, коли Swing була ще дуже повільною та потребувала використовувати багато ресурсів. SWT так само як і AWT використовує вже вбудовані у ОС компоненти, але ця бібліотека для кожної платформи має свої інтерфейси

взаємодії. Для кожної системи треба ставляти окрему JAR-бібліотеку з відповідною версією SWT, завдяки цьому з'явилась можливість максимально повно використовувати існуючі функції компонентів кожної ОС. Компоненти і функції, яких бракувало, були реалізовані за допомогою 2D так само як і у бібліотеці Swing.

До переваг SWT можна віднести швидкість її роботи, що пояснюється використанням компонентів системи. Візуальний редактор форм дістався SWT від Eclipse. Так само, як і у Swing, для цієї бібліотеки існує багато літератури з документацією та прикладами, що спрощую процес навчання роботі з нею. Також SWT має можливість використовувати як компоненти Swing, так і компоненти AWT [13].

Недоліком SWT є необхідність встановлення окремої бібліотеки для кожної платформи та потреба відстежувати використання ресурсів пам'яті під час їх звільнення оскільки це не відбувається автоматично. SWT має складну архітектуру що дуже ускладнює створення свого власного інтерфейсу, майже унеможливлює це.

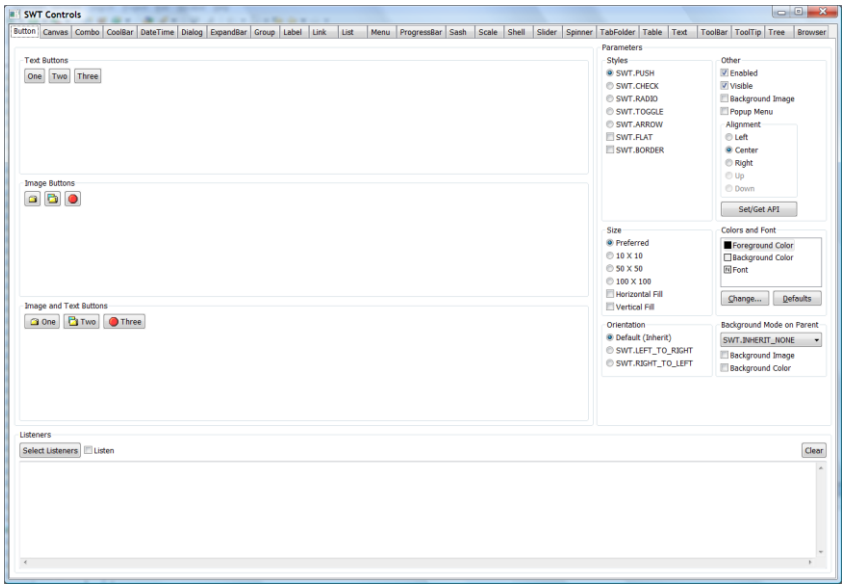


Рис. 37. Інтерфейс створений за допомогою SWT

Приклад програми з використанням бібліотеки SWT наведений на рисунку 37.

Проривом для ГІК бібліотек стала розроблена JavaFX. Значно пришвидшує роботу, програми створеної з використанням JavaFX, графічний конвеєр для промалювання компонентів інтерфейсу. Ця бібліотека має великий набір вбудованих компонентів, зокрема вона має компонент для побудови графіків. Перелік класів наведені у додатку «Класи бібліотеки JavaFX», назви класів співпадають з назвами компонентів. У цій бібліотеці реалізована мультимедійність, анімація, мультитач та безліч ефектів відображення зображень. JavaFX дозволяє реалізувати інсталятор для найпопулярніших платформ: exe або msi для Windows, deb або rpm для Linux, dmg для Mac. У Інтернеті можна знайти багато інструкцій та прикладів використання JavaFX у різноманітних проектах.

Перевагами JavaFX є швидка робота, що забезпечується за допомогою конвеєра, великий вибір різноманітних компонентів, які можна створити за допомогою цієї бібліотеки, підтримка стилів CSS. Приємним фактом про JavaFX для програміста також є наявності утіліти для створення інсталятора програми та можливості запуску через неї програми на комп’ютері.

Недоліком цієї бібліотеки є те, що при її використанні іноді «падає» вся програма або виникають глюки. Це пояснюється тим фактом, що бібліотека є ще новою і знаходиться на стадії активної розробки, а отже зміни у неї впроваджуються досить часто. Також недоліком є те, що JavaFX не стала поки популярною бібліотекою серед розробників, а отже окрім офіційних ресурсів бібліотеки існує мало інструкцій та прикладів її використання, у програмістів, що працюють з мовами JVM, мало досвіду її використання.

Рис. 38. Інтерфейс створений за допомогою JavaFX

Приклад програми з використанням бібліотеки JavaFX наведений на рисунку 38.

Розроблена бібліотека є високорівневою обгорткою над вже існуючими бібліотеками, зокрема над бібліотекою Swing. Шаблонізатор використовує Swing однієї з останніх версій. Вона зводить до мінімуму код, який потрібно написати розробнику для того, щоб створити будь-який з елементів ГІК. Також є дуже зручною для використання завдяки простоті та зрозумілості функцій, які викликають той чи інший компонент інтерфейсу. Так, як розроблена бібліотека також спирається на вбудовані функції ОС, програма ,яка була розроблена з її допомогою має дещо різний вигляд при запуску на різних системах. Побачити різницю у дизайні програм, що виконуються на різних ОС, можна розглянувши рисунки 9-12. За допомогою закладених у бібліотеку функцій розробник може легко створювати нові компоненти інтерфейсу для потреб проекту.

До переваг розробленої бібліотеки шаблонів належить простота роботи з нею. Швидкодія цієї бібліотеки також є достатньо високою, щоб програма з її використанням не висла і працювала плавно та зручно для користувача, так, як за основу взяті бібліотеки з високою швидкодією і використовуються їх прості функції, що швидко обробляються

комп’ютером. Всі компоненти, що можуть бути створені, виглядають у одному стилі, що візуально приємно для користувача.

Недоліком розробленої бібліотеки є невелика кількість реалізованих компонентів графічного інтерфейсу. Також не зручною властивістю для розробників може стати не можливість використання різних стилів, які підтримує, наприклад, JavaFX.

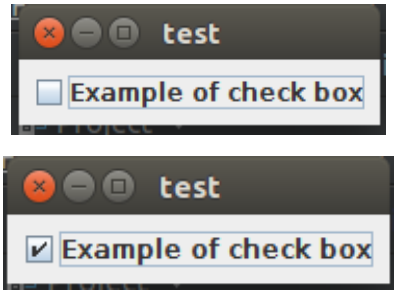


Рис. 39,40. Приклад роботи кнопки-перемикача

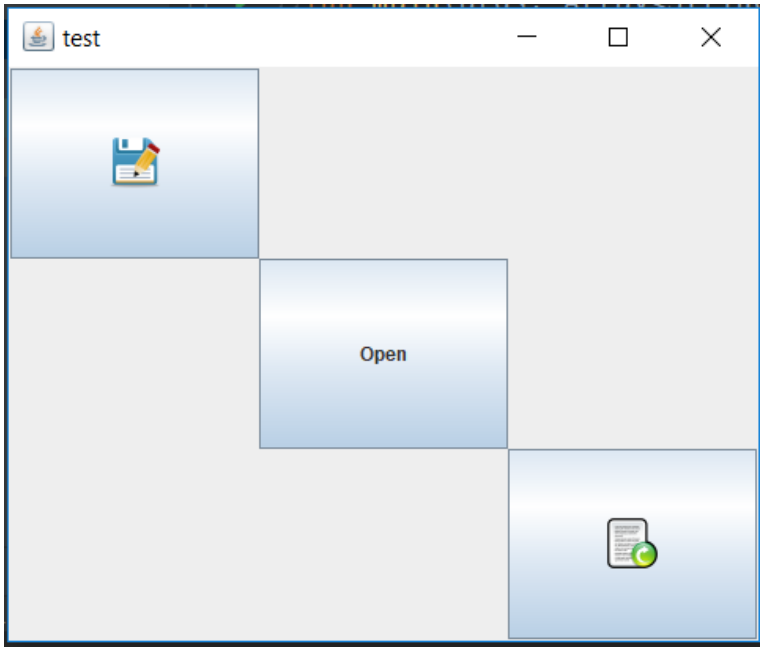


Рис. 41. Приклад створення кнопок

Рис. 42. Приклад створення текстового напису «Label»

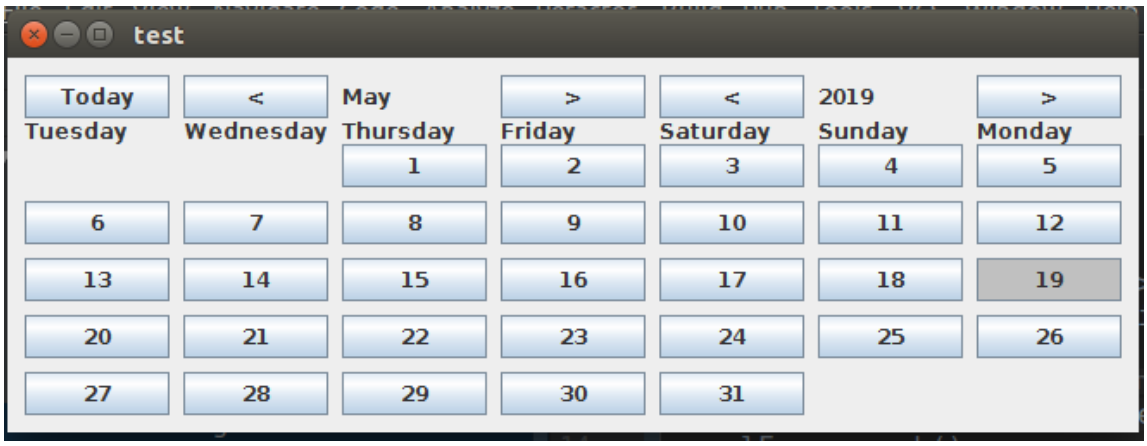


Рис. 43. Приклад створення календаря

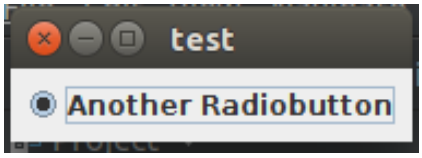


Рис. 44. Радіо-кнопка «Another Radiobutton»

Щоб порівняти графічну складову розглянемо приклади інтерфейсів, створених за допомогою розповсюджених бібліотек, та за допомогою розробленої бібліотеки які наведені на рисунках, які наведені на рисунках 35-44. Інтерфейс розробленої бібліотеки є простішим та мінімалалістичніший за JavaFX, дуже схожим на інтерфейси розроблені за допомогою Swing, AWT, SWT. Розроблена бібліотека має не великий набір компонентів, але він задовольняє потребам розробників. Інтерфейс розробленої бібліотеки використовую функції ОС і завдяки цьому гарно поєднується з іншими програмами будь-якої ОС. Розроблена бібліотека, на відміну від більшості конкурентів, переважно використовує блакитні та сині відтінки які приємні для ока людини і не ріжуть зір користувача. Компоненти розробленої бібліотеки виглядають об’ємними, чого немає у деяких конкурентів, що спрощує сприйняття користувачем цих компонентів.

4.2. Перспективи розвитку

Будь-який новостворений продукт приречений на розвиток та еволюцію. Це зумовлено не лише недосконалістю створеної програми, бібліотеки або будь-якого іншого програмного продукту, а й зміною потреб ринку, архітектури комп'ютерів, на яких буде виконуватись програма, мінливістю моди та багатьма іншими факторами.

Щоб проаналізувати найперспективніші напрямки розвитку продукту треба звернути увагу на найслабкіші сторони свого продукту, на найсильніші сторони конкурентів, передбачити яких функцій може не вистачати користувачам продукту, проаналізувати переваги та недоліки впровадження кожного з них у свій проект.

О одним з недоліків розробленої бібліотеки є невелика кількість реалізованих компонентів, а в бібліотеках-конкурентах реалізовано більше компонентів. Можна стверджувати, що розробка нових компонентів графічного інтерфейсу є основним напрямком розвитку створеного програмного продукту. Так, наприклад, у бібліотеці немає шаблону для створення поля для вводу даних користувачем, що може бути корисним для програм у яких використовуються усілякі анкети або хід програми залежить від вибору користувача. Також у бібліотеці відсутній шаблон для створення спливаючих підказок, це може бути корисно для користувачів, що мають погані навички роботи з комп'ютерними програмами, або для неуважних користувачів.

Іншим недоліком створеного програмного продукту є неможливість створення різних за стилем інтерфейсів. Цей недолік може бути подоланий доданням стилів CSS, однак це може призвести до збільшення громісткості та зниження швидкодії бібліотеки. Це може зробити бібліотеку менш придатною для використання в малих проектах або на проміжних етапах великих проектів, що може негативно позначитись на актуальності даного

продукту. Збільшення стилів, що входять до складу бібліотеки не є перспективним напрямком її розвитку.

Перевагою деяких конкурентів є більша швидкодія компонентів бібліотеки. Це зумовлено безпосереднім використанням функцій ОС. Для розробленої бібліотеки такий спосіб підвищення є неможливим оскільки вона є обгорткою над бібліотеками, що використовують функції системи. Завдяки цьому розроблена бібліотека має лаконічні функції для створення компонентів. Лаконічність функцій та простота їх застосування і є основною ідеєю розробленої бібліотеки.

Зручним для програмістів є наявність методів легко створити інсталятор для програми та можливість простого запуску її користувачем. Така функція реалізована у бібліотеці JavaFX і є можливою перспективою розвитку бібліотеки. Ця функція може бути корисною для невеликих проектів, але не є актуальною для проміжних етапів великих проектів чи програм, що створюються з навчальною метою.

4.3. Висновки до розділу

Існує багато бібліотек для створення ГІК серед мов родини JVM серед яких найпопулярнішими та найвідомішими є Swing, JavaFX, SWT та AWT. У кожної з бібліотек є свої переваги та недоліки серед яких різні набори реалізованих компонентів графічного інтерфейсу, різні швидкодії, різні набори стилів та різна зручність для використання програмістом. Для створення бібліотеки JVM використовують різні підходи серед яких використання функцій написаних на C, використання вбудованих у ОС методів та функцій та інше.

Для розробленої бібліотеки є місце серед інших схожих програмних продуктів зі схожою галуззю застосування. Не зважаючи на певні недоліки створеної бібліотеки, переваги дають можливість вдало використовувати її у невеликих проектах або у навчальних цілях. Основними перевагами

розробленого шаблонізатора є достатній набор компонентів для створення зручного ГІК, простота та інтуїтивна зрозумілість у використанні функцій, гнучкість системи для створення нових компонентів, які не були передбачені, але можуть знадобитися при створенні програми.

Основним перспективним напрям розвитку бібліотеки є збільшення шаблонів щоб задовільнити потреби, які можуть виникнути при розробці програм різної складності. Найменш перспективним напрямком розвитку бібліотеки є збільшення її швидкодії. Так сталося через те, що збільшення швидкодії може призвести до незручностей використання бібліотеки і змінити її цільовий сектор програм для використання.

ВИСНОВКИ

Розроблений бібліотека мов родини JVM створений щоб спростити розробку графічного користувацького інтерфейсу. Основною метою бібліотеки є зведення створення будь-якого компоненту інтерфейсу до одного рядка коду який складається з назви функції та параметрів, які їй передаються.

Бібліотека включає в себе шаблони для створення вікна програми, кнопки, випадаючого списку, кнопки-перемикача, радіо-кнопки, текстового напису, календаря, вкладки, іконки та інше. Бібліотека забезпечує використання багатомовності інтерфейсу.

Розроблена бібліотека створена для використання її у невеликих проектах, на проміжних етапах великих проектів або у навчальних цілях при написанні лабораторних робіт чи інших програм.

Розроблена бібліотека має переваги, які роблять її актуальною для обраної сфери застосування і дають можливість використовувати її у різних проектах. Основними перевагами створеного шаблонізатора є:

- простота у використанні;
- достатній набор компонентів для створення різноманітних інтерфейсів;
- висока швидкодія елементів інтерфейсу;
- однаковий стиль всіх компонентів;
- не вимагає задіяти багато ресурсів;
- підтримка багатомовного інтерфейсу;
- можливість створення нових компонентів на основі вже існуючих.

Бібліотека має деякі недоліки зумовлені її особливостями реалізації та обмеженнями, що накладає її цільове призначення. Недоліками розробленого шаблонізатора є:

- відсутність всіх теоретично можливих компонентів інтерфейсу;
- неможливість зміни стилю розроблених програм;
- відсутність функції створення інсталятора.

Можливі зміни для покращення роботи розробленої бібліотеки:

- збільшення шаблонів компонентів ГІК;
- забезпечення автоматичного створення інсталятора.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Wilkes M. V., Wheeler D. J., Gill S. Preparation of Programs for an Electron ic Digital Computer. — Addison-Wesley, 1951.
2. Уілкс М., Уіллер Д., Гілл С. «Своркння програм для електроних рахункових машин». Издательсо иностранной литературы, 1953
3. ISO/IEC/IEEE 24765-2010 Systems and software engineering — Vocabulary.
4. Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship, 2008.
5. Raj Lal. «User Interface evolution in last 50 years», Digital Design and Innovation Summit, San Francisco, 2013.
6. "About PARC – PARC, a Xerox company" . parc.com (Електронний ресурс).
7. Edson J. Design Like Apple: Seven Principles For Creating Insanely Great Products, Services, and Experiences, 2012.
8. Scott J. Byte Magazine Volume 08 Number 06 - 16-Bit Designs, 1983.
9. Lance E. "Editor's Notes". Compute's Gazette, 1988.
10. Петцольд Ч. Programming Windows, 1990.
11. Лонг Ф. Руководство для программиста на Java: 75 рекомендаций по написанию надежных и защищенных программ, 2014.
12. Gosling J., Joy B., Steele G., Bracha G.. The Java Language Specification, Third Edition, 2005.
13. Хорстманн С. К., Корнелл Г.. Java2. Библиотека профессионала, том 1. Основы, 2007.
14. Zukowski J. The Definitive Guide to Java Swing, Third Edition, 2005.