МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

ІНТЕРНЕТ-ТЕХНОЛОГІЇ ТА МОВА ПРОГРАМУВАННЯ JAVA

Методичні вказівки до виконання лабораторних робіт 1-6
для підготовки здобувачів освітньо-кваліфікаційного рівня «бакалавр»
спеціальностей 122 «Комп’ютерні науки» та 126 «Інформаційні системи та технології»

Київ 2024

УДК 004.43

І

Укладач: О.А. ПОПЛАВСЬКИЙ канд. техн. наук, доцент кафедри інформаційних технологій

Рецензент О.О. Терентьєв, д-р техн. наук, професор, Київський національний університет будівництва і архітектури

Відповідальна за випуск Т.А. Гончаренко, канд. техн. наук, доцент.

Затверджено на засіданні кафедри інформаційних технологій, протокол № 9 від 27 березня 2024 року.

В авторській редакції.

І **Інтернет-технології та мова програмування Java**: методичні вказівки до виконання лабораторних/ уклад.: Поплавський О.А. – Київ: КНУБА, 2024. – 64 с.

Містять зміст, порядок оформлення і вказівки до виконання лабораторних робіт 1-6.

Призначено для здобувачів спеціальностей 122 «Комп’ютерні науки» та 126 «Інформаційні системи та технології»

Зміст

Загальні положення

Методичні вказівки до виконання лабораторних робіт з дисципліни "Інтернет-технології та мова програмування Java" призначені для надання студентам необхідних теоретичних знань та практичних навичок у сфері розробки веб-додатків та програмного забезпечення на мові програмування Java. Основна увага в методичних вказівках приділяється оволодінню основами мови Java, а також сучасним інтернет-технологіям, які широко використовуються в професійній діяльності розробників програмного забезпечення.

Перед початком виконання лабораторних робіт кожен студент повинен вивчити теоретичний матеріал, відповідний конкретній темі, та ознайомитися з базовими принципами роботи в обраному середовищі розробки. Важливим аспектом є засвоєння методів проектування та розробки веб-додатків, а також знання фреймворків та бібліотек, що використовуються разом з мовою Java для створення ефективних і функціональних рішень.

Методичні вказівки містять інструкції до виконання ряду лабораторних робіт, кожна з яких зосереджена на певних аспектах використання мови Java та інтернет-технологій. Студенти мають навчитися не тільки основам програмування, але й комплексному використанню мови для розв'язання специфічних завдань, пов'язаних з розробкою веб-додатків.

Кожен розділ методичних вказівок включає мету лабораторної роботи, вимоги до теоретичної та практичної підготовки студентів, а також детальні рекомендації щодо підготовки та виконання роботи. Серед важливих елементів кожної лабораторної роботи — опис теоретичних відомостей, необхідних для її виконання, індивідуальні завдання, контрольні запитання та вимоги до оформлення звіту про виконану роботу.

Для виконання лабораторних робіт рекомендується використовувати середовище розробки, що підтримує мову програмування Java, наприклад, IntelliJ IDEA або Eclipse. Студенти повинні оволодіти навичками роботи з інтегрованим середовищем розробки (IDE), зокрема, створенням нових проектів, редагуванням коду, використанням системи управління версіями та іншими інструментами, що сприяють ефективній розробці програмних продуктів.

У процесі роботи над лабораторними завданнями студенти мають продемонструвати здатність аналізувати задачі, вибирати оптимальні шляхи їх рішення, розробляти алгоритми та імплементувати їх за допомогою мови

програмування Java. Важливим є також розвиток навичок самостійної роботи з документацією, пошуком і вибором необхідних інструментів та бібліотек.

Звіт з лабораторної роботи повинен містити повний опис виконаних завдань, включаючи умову задачі, архітектуру програми, початковий код з коментарями, приклади виконання програми та аналіз отриманих результатів. Оформлення звіту має відповідати встановленим вимогам академічного письма.

Методичні вказівки забезпечують комплексний підхід до вивчення мови програмування Java та сучасних інтернет-технологій, сприяючи формуванню у студентів професійних компетенцій, необхідних для успішної кар'єри в галузі IT.

Лабораторна робота №1

Тема: Робота з *JDK* і *IDE IntelliJ IDEA*.

Мета: ознайомитися з основами технології *Java*. Навчитися користуватися засобами *JDK* та *IDE* для створення *java*-програм.

Теоретичні питання

1. Призначення і особливості мови *Java*.
2. Технологія *Java* і її складові. Робота з *JDK* без *IDE*.
3. Призначення і особливості *IDE IntelliJ IDEA*.

Контрольні питання

1. Опишіть призначення складників технології *Java*: *JVM, JRE, JDK, Java IDE*.
2. Які переваги дає виконання *Java*-програм на *JVM* і чому?
3. Чи можна запустити java-програму, не маючи *IDE* і *JDK*? Як це зробити?
4. Чи можна компілювати java-програму, не маючи *IDE*? Як це зробити?
5. Опишіть основні особливості *IDE IntelliJ IDEA* та її переваги порівняно з іншими *IDE*.

Індивідуальні завдання

Загальне завдання

1. Встановіть *JDK* (версію не нижче 1.8). Задайте шлях до компілятора в *PATH*. У звіті опишіть порядок своїх дій і надайте скріншоти для кожного етапу.
2. Скомпілюйте і виконайте просту програму з командного рядка, користуючись тільки засобами *JDK*. У звіті опишіть порядок своїх дій і наведіть скріншоти для кожного етапу.

```
import java.util.Scanner;                    // Підключаємо клас з бібліотеки
public class Joke {                          // Основний клас програми
    public static void main(String [] args) {  // Головний метод
        Scanner scanner = new Scanner(System.in); // Створюємо об'єкт для читання
        System.out.println("Enter your first name");// Виводимо текст
        String name = scanner.next();            // Вводимо ім'я
        String reverse = "";                     // Змінна для збереження результату
        for (int i = name.length() - 2; i> = 0; i--) {   // Міняємо порядок букв
        reverse + = name.substring(i, i + 1);
        }
```

```
        // Робимо першу букву заголовною
        reverse = name.substring(name.length() - 1, name.length())
                .toUpperCase() + reverse.toLowerCase();
        System.out.println ("Then my name is" + reverse); // Виводимо результат
        }
}
```

3. Скомпілюйте і виконайте ту ж програму в *IDE*. У звіті опишіть порядок своїх дій зі створення та налаштування проекту і надайте скріншоти для кожного етапу.
4. Протестуйте на прикладі тієї ж програми роботу засобів доповнення та автогенерації коду, підказок, а також засобів налагодження (*debugging*) *IDE*. У звіті перерахуйте можливості, якими ви навчилися користуватися, і надайте їх скріншоти.

Основні гарячі клавіші IDE

IntelliJ IDEA має великий набір гарячих клавіш, які дозволяють виконувати велику частину роботи, не відриваючи рук від клавіатури, і в такий спосіб економити час розробника. Ті ж поєднання клавіш працюють і в інших продуктах *JetBrains*, як-от *PyCharm* і *CLion*. Нижче наведено перелік найбільш уживаних гарячих клавіш.

Редагування

- *Shift + Del(Ctr + Y)* Видалити рядок (відмінність у тому, де потім залишиться курсор);
- *Ctr + Del* Видалити від поточної позиції до кінця слова;
- *Ctr + Backspace* Видалити від поточної позиції до початку слова;
- *Ctr + D* Дублювати поточний рядок;
- *Tab / Shift + Tab* Збільшити / зменшити поточний відступ;
- *Ctr + Alt + I* Вирівнювання відступів у коді;
- *Ctr + Alt + L* Приведення коду у відповідність із code style;
- *Ctr + /* Закоментувати / розкоментувати поточний рядок;
- *Ctr + Shift + /* Закоментувати / розкоментувати виділений код;
- *Ctr + - / +* Фолдінг: згорнути / розгорнути фрагмент коду.

Вікна, вкладки

- *Alt + ←/ →* Перейти на іншу вкладку редактора
- *Ctr + F4* Закрити вкладку;
- *Alt + цифра* Відкриття / закриття вікон Project, Structure,

Changes і т. д.;

- *Ctr + Tab* Перехід між вкладками та вікнами;
- *Shift + Esc* Закрити активне вікно;
- *Ctr + коліщатко* Зміна розміру шрифту (налаштовується в меню File | Settings).

Підказки та документація

- *Ctr + Q* Документація до елементу, на якому знаходиться курсор/виділення;
- *Ctr + Shift + I* Показати реалізацію методу або класу;
- *Alt + Q* Відобразити ім’я класу або методу, у якому знаходиться курсор;
- *Ctr + P* Підказка про очікувані аргументи методу;
- *Ctr + F1* Показати опис помилки або попередження під курсором;
- *Alt + Enter* Показати пропоновані варіанти дій («лампочки»).

Пошук

- *Подвійний Shift* Швидкий пошук в усьому проекті (в тексті);
- *Ctr + Shift + A* Швидкий пошук у налаштуваннях, діях і т. д.;
- *Alt + ↓ / ↑* Перейти до наступного / попереднього методу;
- *Ctr + [та Ctr +]* Перехід на початок і кінець поточної області коду;
- *Ctr + F* Пошук у поточному файлі;
- *Ctr + Shift + F* Пошук в усіх файлах проекту (перехід – F4);
- *F3 / Shift + F3* Перейти до наступного / попереднього результату пошуку;
- *Ctr + F12* Список методів із переходом до їх оголошення;
- *Ctr + H* Ієрархія наслідування поточного класу і перехід за нею;
- *Ctr + N* Пошук класу за ім’ям і перехід до нього;
- *Ctr + Shift + N* Пошук файла за ім’ям і перехід до нього;
- *Ctr + B* Перейти до оголошення змінної, класу або методу;
- *Ctr + Alt + B* Перейти до реалізації класу або методу;
- *Ctr + Shift + B* Визначити клас об’єкта і перейти до його реалізації;
- *Shift + Alt + 7* Знайти всі місця, де використовується метод або змінна;
- *Ctr + Alt + 7* Аналогічно до попереднього пункту, але результат виводиться у спливаючому вікні.

Генерація коду і рефакторінг

- *Ctr + Space* Автодоповнення коду;
- *Ctr + Shift + Space* Розумне автодоповнення;
- *Ctr + O* Перевизначити метод батьківського класу;
- *Alt + Insert* Автогенерація коду (відобразити варіанти);
- *Shift + F6* Перейменування об'єкта під курсором (змінної, класу, методу);
- *Alt + Delete* Безпечне видалення класу, методу або атрибута.

Інше

- *Ctr + ~* Швидке перемикання колірної схеми, *code style* і т.д.;
- *Alt + F12* Відкрити / закрити термінал;
- *F10* Запустити останню запущену програму або тест;
- *Shift + F10* Запустити відкриту в редакторі програму або тест;
- *Shift + Alt + F10* Список програм / тестів, які нещодавно запускалися;
- *Ctr + F2* Зупинити запущену програму.

Лабораторна робота №2

Тема: Базові конструкції і типи Java. Використання масивів у *Java*.

Мета: ознайомитися з основами синтаксису *Java*. Навчитися використовувати базові типи і конструкції Java при написанні прикладних програм.

Теоретичні питання

1. Оголошення та керуючі конструкції.
2. Стандартні потоки введення і виведення. Введення даних за допомогою класу *java.util.Scanner*. Форматований вивід за допомогою об'єкта *System.out*.
3. Регулярні вирази.
4. Примітивні типи даних і правила перетворення типів.
5. Змінні. Їх види. Область видимості і час життя змінної.
6. Робота з масивами.
7. Передача параметрів до методів.

Контрольні питання

1. Перерахуйте найважливіші особливості синтаксису *Java*.
2. У чому відмінність керуючих операторів (розгалуження і циклів) у

Java від керуючих операторів *C++*? Наведіть приклади використання обох форм циклу *for*.

3. Опишіть основні можливості класу *java.util.Scanner*.
4. Якими способами можна прочитати дійсне число з рядка (тобто змінної типу *String*)?
5. Складіть регулярні вирази:
    a. для розпізнавання дати, яка може бути записана в будь-якому з форматів *dd-mm-yy, dd\mm\yy, dd-mm-yyyy,* або *dd\mm\yyyy* (один вираз має передбачати всі варіанти);
    b. для розпізнавання адрес електронної пошти.
6. Які примітивні типи даних є в *Java*? Опишіть правила перетворення типів.
7. Яких спеціальних значень можуть набувати змінні числових типів? Як можна отримати мінімальне позитивне дійсне число типу *float* і максимальне ціле число типу *int*?
8. У чому відмінність між примітивними типами і типами посилань? Наведіть приклади ситуацій, у яких ці відмінності проявляються.
9. Як передаються параметри примітивних типів і типів посилань при виклику методів?
10. Перерахуйте способи ініціалізації масиву в *Java. 11.* Які операції можна виконувати з масивами?

Індивідуальні завдання

Для виконання кожного індивідуального завдання напишіть консольний додаток. Реалізуйте введення вихідних даних із клавіатури і контроль правильності введення. Результат виведіть у консоль у відформатованому і зручному для читання вигляді.

Загальне завдання

Реалізуйте простий калькулятор, який обробляє рядкові команди виду «*a operation b*», де *a, b* – дійсні числа, *oper* – один зі знаків операцій +, –, *, /, ^. Якщо перший операнд не вказано, то замість нього потрібно використовувати результат попередньої дії. Наприклад:

2 ^ 4

16

- 2.5

13.5

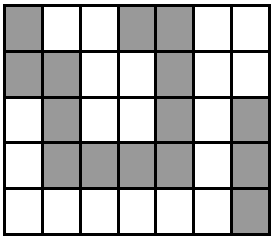
Для розбору вхідних виразів і перевірки правильності введення використовуйте регулярні вирази.

Варіант 1

1. Для довільного цілого числа знайдіть довжину найдовшої послідовності цифр, що повторюються у десятковому записі цього числа. Наприклад, для 14454446 результат дорівнює 3 (цифра 4 повторюється 3 рази поспіль), а для 3434719 – 1 (немає повторення цифр).

   Вказівка. Для поділу числа на цифри використовуйте цілочисельне ділення (/) та ділення за модулем (%) на 10.

2. Чорно-біле зображення задано двовимірним масивом (n×m) пікселів, де 0 відповідає чорному, а 1 – білому. Зображення містить чорну лінію товщиною 1 піксель на білому тлі, що починається з пікселя (1,1), наприклад:



   Виведіть координати чорних пікселів, у порядку, в якому вони розташовані на лінії.

   Вказівка. Вважати, що кожен чорний піксель на лінії, яка відрізняється від початкового і кінцевого, має рівно два сусідніх пікселів чорного кольору.

Варіант 2

1. Реалізуйте генератор псевдовипадкових чисел, що використовує алгоритм Ньюмана:
   1.1. Вибрати довільне початкове 4-значне число (від 0000 до 9999).
   1.2. Помножити число саме на себе – вийде 8-значне число (числа меншої довжини доповнюються зліва нулями).
   1.3. Відкинути перші дві і останні дві цифри – вийде знову 4-значне число.
   1.4. Повертатися до кроку 2, поки не буде знову отримано вихідне число або будь-яке інше число з тих, що раніше зустрічалися у згенерованій послідовності (*).

   Приклад послідовності:

*3456 – 3456 * 3456 – 11943936 – 9439 – 9439 * 9439 – 89094721 – 947 – ...*

(*) Довжина отриманої послідовності значно залежить від вибору початкового числа. Експериментально визначте, яке з можливих початкових

чисел забезпечує найдовшу послідовність.

2. Два полінома $P_1 = a_n x^n + a_{n-1}x^{n-1} + ... + a_1x + a_0$ і $P_2 = b_m x^m + b_{m-1}x^{m-1} + ... + b_1x + b_0$ задані наборами своїх коефіцієнтів (a_n , a_{n-1}, ... , a_1, a_0) і (b_n , b_{n-1}, ... ,b_1, b_0). Знайдіть добуток цих поліномів за правилом множення «у стовпчик». Наприклад:

$P_1 =$ $2x^3 + 0x^2 - 3x - 1$

$P_2 =$ $2x^2 + x - 5$

$-5P_1 :$ $-10x^3 - 0x^2 + 15x + 5$

$+ xP_1:$ $2x^4 + 0x^3 - 3x^2 - x$

$+2x^2P_1 :$ $4x^5 + 0x^4 - 6x^3 - 2x^2$

$P_1 \times P_2 =$ $4x^5 + 2x^4 - 16x^3 - 5x^2 + 14x + 5$

Варіант 3

1. Опуклий багатокутник на площині заданий координатами своїх вершин. Знайдіть його площу.

   Вказівка. Розбийте багатокутник діагоналями на кілька трикутників і знайдіть площу кожного трикутника окремо.

2. Заданий двомірний масив (не обов'язково квадратний) поверніть на 90º за годинниковою стрілкою і виведіть отриманий таким чином новий масив.

Варіант 4

1. 6-значний номер трамвайного квитка є «щасливим», якщо збігаються суми перших трьох і останніх трьох цифр номера. Визначте, скільки всього існує «щасливих» номерів.

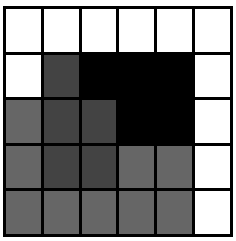
   Вказівка. Для цього досить перебрати всі можливі 6-значні номери.

2. У заданому двовимірному масиві розташуйте рядки у порядку зростання суми їх елементів.

Варіант 5

1. Опуклий багатокутник на площині заданий координатами своїх вершин. Перевірте, чи є він правильним. Для цього перевірте рівність усіх його сторін, а також кутів між сусідніми сторонами.

2. Зображення задано двовимірним масивом (n×m) пікселів, де 0 відповідає чорному, а 10 – білому, а всі інші значення – різним градаціям сірого. Складіть гістограму зображення – тобто масив із 10 елементів, де в кожній *i*-й комірці зберігається кількість пікселів на зображенні, які мають колір *i*.

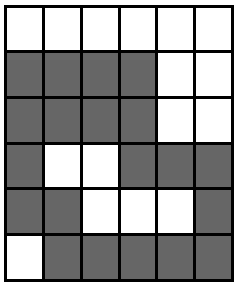


Варіант 6

1. Дано дійсні числа a, b. Знайдіть рішення рівняння $cos(x) = x^2$ на проміжку *[a; b]* методом ділення навпіл з точністю до 0,001.
2. Нехай масив розміру 3×3 у довільний спосіб заповнений числами 0 і 1. Шляхом перебору знайдіть і виведіть усі такі масиви, в яких суми елементів у рядках, стовпцях і діагоналях збігаються.

Варіант 7

1. Дано натуральні числа *a* і *b*. Знайдіть їх найбільший спільний дільник. *Вказівка.* Скористайтеся тим фактом, що якщо *a>b>0*, то НСД(*a,b*) = НСД(*a mod b, b*).
2. Чорно-біле зображення задано двовимірним масивом (n×m) пікселів, де 0 відповідає чорному, а 1 – білому. Зображення являє собою довільну чорну лінію товщиною 1 піксель на білому тлі, наприклад:

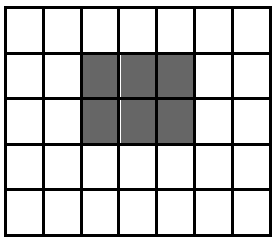


Визначте, чи є ця лінія замкнутою.

Вказівка. Вважати, що кожен чорний піксель на лінії, що відрізняється від початкового і кінцевого, має рівно два сусідні пікселі чорного кольору.

Варіант 8

1. Виведіть усі прості числа, які не перевищують заданого числа a.
   Вказівка. Для того, щоб визначити, чи є число n простим, досить перевірити залишки від ділення цього числа на всі цілі числа від 1 до n.
2. Чорно-біле зображення задано двовимірним масивом (n×m) пікселів, де 0 відповідає чорному, а 1 – білому. Зображення являє собою чорний прямокутник на білому тлі, наприклад:



Знайдіть координати центру прямокутника.

Варіант 9

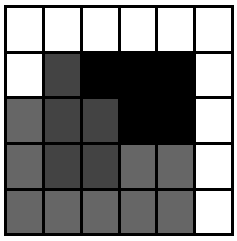
1. Відомо, що якщо від десяткового числа відняти суму його цифр, то залишок ділиться без остачі на 9. Перевірте це твердження для 1000 різних цілих чисел. Числа генеруйте у випадковий спосіб.

   Вказівка. Для поділу числа на цифри використовуйте цілочисельне ділення (/) та ділення за модулем (%) на 10.
2. Заданий двомірний масив (не обов'язково квадратний) поверніть на 90º проти годинникової стрілки і виведіть отриманий у такий спосіб новий масив.

Варіант 10

1. Дано ціле десяткове число *a*. Знайдіть число, отримане перестановкою цифр числа *a* у зворотному порядку. Мають підтримуватись числа, що містять до 9 знаків.

   Вказівка. У цьому завданні заборонено користуватися методами класу *String*. Для поділу числа на цифри використовуйте цілочисельне ділення (/) та ділення за модулем (%) на 10.
2. Зображення задано двовимірним масивом ($n \times m$) пікселів, де 0 відповідає чорному, а 255 – білому, а всі проміжні значення – різним градаціям сірого, наприклад:



Зменшіть зображення удвічі, тобто отримайте новий масив ($n/2 \times m/2$) пікселів, де кожен піксель визначається як середнє між чотирма пікселями вихідного масиву.

Варіант 11

1. Для довільного цілого числа знайдіть цифру в десятковому записі цього числа, яка повторюється найбільшу кількість разів поспіль.

Наприклад, для числа 14454446 результат дорівнює 4 (цифра 4 повторюється 3 рази поспіль), а для 3434719 – 3 (всі цифри повторюються тільки один раз, тому відповіддю є перша з них).

Вказівка. Для поділу числа на цифри використовуйте цілочисельне ділення (/) та ділення за модулем (%) на 10.

2. Дано дві прямокутні (не обов'язково квадратні) матриці A і B. Виконайте їх матричне множення.

   Вказівка. Якщо A має розмір ($n \times m$) і B має розмір ($m \times p$), то їх твором є матриця C розміру ($n \times p$), в якій:

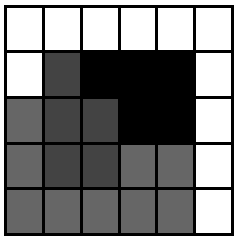
$$c_{ij} = \sum_{k=1}^{m} (a_{ik} \cdot b_{kj})$$

Варіант 12

1. Для заданого цілого числа *n* виведіть, скільки існує простих чисел, що не перевищують *n*.

   Вказівка. Для того, щоб визначити, чи є число x простим, досить перевірити залишки від ділення цього числа на всі числа від 1 до $\sqrt{x}$.

2. Зображення задано двовимірним масивом ($n \times m$) пікселів, де 0 відповідає чорному, а 255 – білому, а всі проміжні значення – різним градаціям сірого, наприклад:



Збільшіть зображення удвічі, тобто отримайте новий масив ($2n \times 2m$) пікселів, у якому кожному пікселю вихідного масиву відповідає чотири однакові сусідні пікселі.

Варіант 13

1. Дано ціле число. Знайдіть кількість значущих нулів у десятковому записі цього числа.

   Вказівка. Для поділу числа на цифри використовуйте цілочисельне ділення (/) та ділення за модулем (%) на 10.

2. Відомо, що середнє геометричне набору позитивних чисел не перевищує їх середнього арифметичного: $((a_1 + a_2 + \ldots + a_n)/n \geq$

$(a_1 \cdot a_2 \cdot \ldots \cdot a_n)^{1/n}$). Для введеного користувачем значення n перевірте це твердження для 1000 різних наборів. Числа a_i генеруйте у випадковий спосіб у діапазоні від 0 до 100, *кількість чисел у наборах також має були випадковою* – від 5 до 20.

Виведіть всі набори, в яких різниця лівої і правої частин нерівності склала менше 5 %.

Варіант 14

1. Користувач вводить в один рядок довільну кількість чисел, розділених комами і пробілами. Знайдіть серед цих чисел довжину найдовшої зростаючої підпослідовності. Наприклад, для послідовності «1, 2, 3, 1, 2, 5, 10, 2» правильний результат – 4.
2. Нехай масив розміру 3×3 у довільний спосіб заповнений числами 1, 2, 3 (можливі повторення). Шляхом перебору знайдіть і виведіть усі такі масиви, в яких суми елементів у всіх рядках і стовпцях збігаються.

Варіант 15

1. Дано 9-значне ціле число. Визначте, чи є в десятковому записі цього числа цифри, що повторюються поспіль. Якщо є, то виведіть (*по одному разу*) всі цифри, які повторюються. Наприклад, для числа 445663331 5 результат має бути «4, 6, 3».

   Вказівка. У цьому завданні заборонено користуватися методами класу *String*. Для поділу числа на цифри використовуйте цілочисельне ділення (/) та ділення за модулем (%) на 10.
2. Відомо, що для будь-яких позитивних $a_1, a_2, \ldots, a_n$ виконується нерівність $(\sqrt{a_1^2 + a_2^2 + \ldots + a_n^2/n} \geq (a_1 + a_2 + \ldots + a_n)/n)$. Для заданого n перевірте це твердження для 10 000 різних наборів. Числа генеруйте у випадковий спосіб у діапазоні від 1 до 20, *кількість чисел у наборах також має бути випадковою* – від 3 до 10. Виведіть той набір, для якого різниця лівої і правої частин нерівності виявилася найбільшою.

Додаткові завдання

3. Модифікуйте програму-калькулятор, щоб вона могла працювати в будь- якій системі числення. Для перемикання систем числення передбачте окрему команду.
4. Група з n дітей грає в квача. Щоб визначити, хто квач, вони використовують звичайну «лічилку» з m слів. Діти стають у коло і починають «лічилку». Той, на кого випадає останнє слово «лічилки»

– виходить з кола, і лічилка починається заново з наступного за вибулим гравця. Напишіть програму, яка дасть можливість визначити номер гравця у *початковому* колі, який залишиться останнім у кінці лічилки.

5. Петро запропонував Ані зіграти в азартну гру: Петро кілька разів поспіль кидає монетку, і якщо Аня вгадає, скільки разів поспіль випаде «орел», то Петро заплатить їй $5; якщо ж вона не вгадає, то заплатить Петрові всього $1. Аня знає, що монетка Петра не проста, і що вона випадає «орлом» у 86 % випадків.

   Напишіть програму, яка змоделює процес кидання такої монетки і дасть можливість Ані приблизно визначити:

   a. яка кількість «орлів», що випадають поспіль, є найбільш ймовірною?
   b. чи вигідно взагалі грати в таку гру? Тобто який середній виграш (або програш) вона отримає за одну гру, якщо буде постійно згадувати оптимальну кількість «орлів».

   Вказівка. Використовуйте генератор випадкових чисел, а також масиви для зберігання проміжних даних.

Додаткові відомості.

Типи даних

Таблиця 1

Особливості роботи з примітивними та об’єктними типами

	Примітиви	Об’єкти
Зберігає	Значення	Посилання
Може бути null	Ні	Так
Ініціалізація	Без **new :** int a=1	Type obj = **new** Type (param)
Час життя	До кінця блоку	Поки існують посилання
Оператор =	Копіює значення	Копіює *посилання*, але не об’єкт
Порівняння	a==b	a.equals(b) (*крім масивів*)

Таблиця 2

Порівняння різних типів змінних Java

Тип змінної	**Можливі значення**	**Значення за замовчуванням**
Примітивний	Визначаються границями типу	false, ‘\u0000’, 0
Клас	null (нульове посилання), або посилання на об’єкт цього класу чи його спадкоємця	null
Інтерфейс	null, або на об’єкт будь-якого класу, що реалізує цей інтерфейс	null
Масив	null, або посилання на масив об’єктів відповідного типу	null
Object	null, або посилання на *будь-який* об’єкт	null

Додаткові відомості.

Регулярні вирази

Java підтримує всі можливості ***регулярних виразів***, зокрема:

1. Можна задавати класи символів, перерахувавши всі відповідні символи у квадратних дужках:
   [az] один із двох символів: 'a', 'z';
   [a-z] будь-яка з 26 малих латинських букв від 'a' до 'z';
   [^a-z] будь-який символ, крім 26 малих латинських букв від 'a' до 'z';
2. Деякі символьні класи можна замінити спеціальними *метасимволами* (табл. 3):

Примітка. У Java у строкових константах знак \ необхідно екранувати: \\.

3. Для *групування* декількох послідовних символів використовуються дужки (). Далі у виразі можна звернутися до групи за номером за допомогою спеціальних символів \1, \2, ..., наприклад:
   (\\w\\d)=\\1 відповідає рядкам "a1=a1", "b2=b2" і т. д.
   Об’єкт *Scanner* після виклику *findInLine()* дає змогу отримати підрядки, що відповідають кожній групі, за допомогою виклику методу *match().group(n)*.

Таблиця 3

Найбільш вживані спеціальні символи регулярних виразів

Символ	Еквівалент	Відповідність
.		Будь-який символ
\d	[0-9]	Цифровий символ
\D	[^0-9]	Нецифровий символ
\s	[\f\n\r\t\v]	Пробільний символ (пробіл, табуляція або переведення рядка)
\S	[^ \f\n\r\t\v]	Непробільний символ
\w	[[:word:]]	Літера, цифровий символ або знак підкреслення
\W	[^[:word:]]	Будь-який символ, крім літери, цифри та знака підкреслення

4. Для варіювання частин виразу використовується символ |, наприклад:
   (one|two|three)=[123] відповідає рядкам "*one=1*", "*two=3*", "*two=1*" і т. д.;
   one|two|three=[13] відповідає рядкам "*one*", "*two*", "*three=1*", "*three=3*".
5. Кількість повторень символу чи групи задається за допомогою *квантифікаторів* (табл. 4):

Таблиця 4

Квантифікатори регулярних виразів

Позначення	Кількість повторень	Еквівалент	Приклад	Відповідність
{n,m}	Від n до m		colou{1,2}r	colour, colouur, але не color
?	Нуль чи одне	{0,1}	colou?r	color, colour
*	Нуль чи більше	{0,}	colou*r	color, colour, colouuur і т.д.
+	Одне чи більше	{1,}	colou+r	colour, colouur і т.д. (але не color)

Наприклад, вираз "-?\\d+\\.?\\d*" розпізнає дійсне число: знак «–» може з’являтися 0 або 1 раз, далі 1 або більше цифр (\\d+), потім символ «.» – 0 або 1 раз, і нарешті, дробова частина – будь-яка кількість цифр (\\d*). Приклади розпізнаваних чисел: 001.33, –34. , 12. Зауважимо, що в такому разі нам потрібно, щоб символ «.» позначав крапку, а не довільний символ, тому ми його екранували слешами.

Лабораторна робота №3

Тема: Класи в *Java*. ООП та колекції.

Мета: ознайомитися з основами можливостями об’єктно-орієнтованого програмування у *Java*. Навчитися використовувати колекції та виключення при написанні прикладних програм.

Теоретичні питання

1. Визначення класу в *Java*. Модифікатори способу доступу.
2. Змінні класу і змінні реалізації. Використання модифікаторів *static* і *final* для змінних у класі.
3. Методи класу. Статичні і фінальні методи. Конструктори. Метод *finalize()*.
4. Абстрактні методи і класи. Інтерфейси.
5. Клас *Object* і його методи.
6. Рефлексія. Клас *Class* і оператор *instanceof*.
7. Ієрархія класів стандартної бібліотеки *Java*. Класи *System, Math, String, java.util.Arrays*.
8. Класи-колекції в *Java*. Інтерфейси *List, Set* і *Map*. Клас *java.util.Collections*.
9. Виключення. Класи виключень. Генерація та обробка виключень.

Контрольні питання

1. Нехай оголошено рядок *String s = new String("aaa");*. Який результат дадуть вираження *(S=="aaa"), (S.equals("aaa"))* і чому?
2. Нехай у класі програми оголошені методи:

```
static void change (String s) { s = "bbb";}
static void change (int a []) {
    if (a! = null) for (int i = 0; i <a.length; i ++) a [i] = 0;
}
    static void change (int a
        [] []) { if (a! =
```

```
            null)
            for (int i = 0; i <a.length; i ++)
                  for (int j = 0; j <a [i] .length; j ++) a [i] [j] = 0;
      }
```

а також локальні змінні:

```
String s = "aaa";
int x [] = {1,2,3}, y [] [] = {{1,2,3}, {4,5,6}, {7,8,9}};
```

Чи зміняться значення цих змінних після кожного з таких викликів:

a. change(s.clone()); change(x.clone()); change(y.clone());
b. change(s); change(x); change(y);

і чому?

3. Чим відрізняється дія модифікаторів способу доступу *public, private, protected* у Java від іншої знайомої вам мови (C++, C# або ін.)?
4. Опишіть дію модифікаторів *static* і *final* при оголошенні змінних у класі. Наведіть приклади використання таких змінних.
5. Опишіть дію модифікаторів *static* і *final* при оголошенні методів класу. Наведіть приклади використання таких методів.
6. Чим конструктор відрізняється від інших методів класу?
7. Для чого використовуються ключові слова *this* і *super*? Наведіть приклади їх використання.
8. Як оголошуються і для чого використовуються абстрактні методи і класи, а також інтерфейси?
9. Опишіть призначення методів класу *Object*. У яких випадках необхідно перевизначати кожен з цих методів?
10. Які можливості надає інтерфейс *Comparable*? Як він реалізується?
11. Коротко опишіть призначення та основні можливості класів *System, Math, String, java.util.Arrays*.
12. Які реалізації мають інтерфейси *List, Set* і *Map*, і в чому відмінність цих реалізацій?
13. Які можливості надає клас *java.util.Collections*?
14. Як генеруються і обробляються виключення в *Java*?

Індивідуальні завдання

Для виконання кожного індивідуального завдання напишіть консольний додаток. Реалізуйте введення вихідних даних із клавіатури і контроль правильності введення. Результат виведіть у консоль у

відформатованому і зручному для читання вигляді. Реалізуйте *обробку можливих виключень*.

Загальне завдання

1. Напишіть програму, яка демонструє різні способи копіювання і порівняння масивів. Для цього створіть п’ять масивів рядків:

   String a [] = { "1", "2", "3"}, b1 [] = a;
   String b2 [] = a.clone(),
   b3 [] = Arrays.copyOf (a,
   a.length), b4 [] = { "1", "2", "3"};

   Попарно порівняйте ці масиви за допомогою операції порівняння (==), методу *Object.equals()*, методу *Arrays.equals()* і методу *Arrays.deepEquals()*.

   Результати порівняння виведіть у вигляді (табл. 5):

   Таблиця 5

Результат порівняння

b	*b==a*	*b.equals(a)*	*Arrays.equals(a,b)*	*Arrays.deepEquals(a,b)*	*b[0]==a[0]*
b=a	*true*	*true*	*?*	*?*	*?*
b=a.clone()	*?*	*?*	*?*	*?*	*?*
b=Arrays.copyOf(a)	*?*	*?*	*?*	*?*	*?*
b4={"1","2","3"}	*?*	*?*	*?*	*?*	*?*

   Виведення одного рядка таблиці реалізуйте у вигляді окремого методу. Поясніть отримані результати.
2. Виконайте завдання 1 для *двовимірних* масивів рядків.
   Поясніть результати.
3. Виконайте завдання 3 *за варіантом*, при цьому для всіх варіантів необхідно:

- у базовому класі визначити конструктор за замовчуванням, конструктор з параметрами, а також конструктор копіювання;
- у базовому класі перевизначити всі нефінальні методи класу *Object*;
- кілька об’єктів обох класів зберегти в один масив;
- пройти за масивом і для кожного об’єкта вивести:
  - ім’я класу цього об’єкта;
  - результати роботи методів, визначених у базовому класі;
  - результати роботи методів, визначених тільки в похідному класі (якщо це можливо для цього об’єкта).

4. Об’єднайте кілька створених об’єктів свого класу (*з завдання 3 за*

варіантом) у список *List*. Реалізуйте додавання, видалення елементів зі списку, а також виведення списку на екран. Для отриманого списку виконайте *завдання 4 за варіантом* двома способами:

a. за допомогою методів класу Collections;
b. без використання методів класу Collections (тобто реалізувавши алгоритм самостійно).

5. *(*Для програмістів*) Збережіть ті ж об'єкти в колекцію HashMap, де ключами будуть строкові імена об'єктів. Напишіть методи для обробки такої колекції:
   - метод для пошуку значення за ключем;
   - метод для пошуку ключів за значенням;
   - метод, що виконує операцію з *завдання 5 за варіантом*.

Варіант 1

3. Створіть клас «Автобусний маршрут». Внутрішнє представлення маршруту – список (*List*) або масив назв зупинок (пунктів, якими проходить маршрут), а також масив інтервалів за часом між сусідніми зупинками. Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):
   - для додавання нових зупинок;
   - для виведення маршруту в рядок у вигляді «Початковий_пункт – пункт1 – Пункт2 ... Кінцевий_пункт».

   Створіть похідний клас, у якому:
   - додайте метод для знаходження часу проїзду між будь-якими двома заданими зупинками;
   - перевизначте метод toString() для виведення маршруту у вигляді «Маршрут від Початковий_пункт до Кінцевий_пункт займає X хвилин».
4. Напишіть метод для сортування списку маршрутів за назвою початкового пункту, а потім за назвою кінцевого пункту.
5. Напишіть метод, що приймає як параметри дві колекції Map і видаляє з першої колекції всі значення (*не ключі*), які зустрічаються у другій колекції.

Варіант 2

3. Створіть клас «Ламана лінія». Внутрішнє представлення лінії – список (*List*) або масив цілочисельних координат її вузлових точок. Метод *toString()* повинен виводити лінію у вигляді «лінія з точки (x_1, y_1) до точки (x_n, y_n)». Реалізуйте методи класу (статичні) і методи реалізації

(нестатичні):

- для додавання і видалення вузлів;
- для знаходження загальної довжини лінії;
- для виведення в рядок послідовності вузлових точок лінії.

Створіть похідний клас, у якому:

- перевизначте метод *toString()* для виведення лінії у вигляді: «лінія довжини x з точки (x_1, y_1) до точки (x_n, y_n)»;
- додайте метод перевірки, чи має лінія повторювані вузли.

4. Напишіть метод для сортування списку ліній за їх довжиною. Порядок сортування – від довгих ліній до коротких.
5. Напишіть метод, що приймає як параметри дві колекції *Map* і створює нову колекцію *Map*, у яку потрапляють тільки ті ключі, які присутні в першій колекції і відсутні у другій.

Варіант 3

3. Створіть клас «Колір». Внутрішнє представлення кольору – три цілі числа в діапазоні [0; 100], що задають відсоток яскравості червоної, зеленої і синьої складових. Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):
   - для створення кольору за заданими трьома цілими числами в діапазоні [0; 255];
   - для створення кольору за трьома дійсними числами в діапазоні [0; 100] або [0; 1];
   - метод, що за заданими двома кольорами a і b створює градієнт між ними, тобто новий масив об’єктів-кольорів, у якому перший колір дорівнює a, останній дорівнює b, а всі інші є проміжними кольорами.
   - метод визначення яскравості кольору, а також метод порівняння кольорів на «більше» – «менше» за значенням яскравості кольору.

   Створіть похідний клас, у якому:

   - додайте метод для перетворення кольору в модель *HSV*;
   - додайте метод, який у масиві об’єктів-кольорів знаходить той колір, що є найближчим до заданого (за значенням параметра *H*, «*hue*»);
   - перевизначте метод *toString()* для виведення кольору у форматі *HSV*.
4. Напишіть метод для сортування списку кольорів спочатку за

величиною червоної компоненти, потім за величиною синьої компоненти. Порядок сортування – за зростанням.

5. Напишіть метод, який у колекції *Map* видаляє всі повторювані значення (*не ключі*).

Варіант 4

3. Створіть клас «Користувач соціальної мережі». Внутрішнє представлення користувача: ім'я, логін, пароль, дата народження (*java.util.Date*), масив постів користувача (де кожен пост – це об'єкт із двома властивостями: дата публікації та текст). Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):
   - для визначення віку користувача (цілого числа років) на поточну дату;
   - для визначення *рейтингу* користувача як кількості його постів, зроблених за останній місяць;
   - метод *toString()*, який видає інформацію користувача у форматі «ім'я – логін».

   Створіть похідний клас, у якому:
   - додайте статичний метод визначення правильності логіна і пароля (логін має складатися з букв і цифр і починатися з великої літери; пароль має бути не коротшим за 8 символів, обов'язково включати цифри і великі та маленькі літери);
   - перевизначте метод *toString()* для виведення інформації про користувача у форматі «логін – вік – текст останнього за часом посту».
4. Напишіть метод для сортування списку користувачів спочатку за рейтингом, потім за віком (від старших до молодших).
5. Напишіть метод, який виводить вміст колекції *Map* (тобто пари ключ– значення) в алфавітному порядку значень, тобто сортує записи за значенням (*не за ключем*).

Варіант 5

3. Створіть клас «Матриця». Внутрішнє представлення – двовимірний масив чисел. Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):
   - для створення матриці заданого розміру та її заповнення числами (з перевіркою на неприпустимі значення);
   - метод знаходження поелементного множення двох матриць;
   - метод *toString()*, який видає інформацію про матрицю у форматі

«Матриця *n***m*».

Створіть похідний клас, у якому:

- перевизначте метод «множення» для знаходження матричного множення замість звичайного;
- додайте метод знаходження зворотної матриці;
- перевизначте метод *toString()* для виведення елементів матриці за рядками (рядки розділяйте символом *'\ n'*).

4. Напишіть метод для сортування списку матриць за величиною максимального елемента.
5. Напишіть метод, який у колекції *Map* видаляє всі повторювані значення (*не ключі*).

Варіант 6

3. Створіть клас «Адреса». Внутрішнє представлення – індекс (5-значне ціле число), місто, вулиця, номер будинку, номер квартири. Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):
   - метод створення об’єкта-адреси на основі рядка виду «індекс, вулиця, будинок, квартира місто», де індекс і квартира можуть бути не вказані, наприклад:«34668 Вінниця, вул. Інженерна, 25», «Вінниця, вул. Інженерна, 25, кв. 32»;
   - метод, що порівнює дві адреси та визначає, яка частина адреси збігається (повертає спільну частину адреси у вигляді рядка);
   - метод перетворення адреси в рядок виду «04123 просп. Перемоги, 35, м. Київ».

   Створіть похідний клас, у якому:
   - додайте метод, що має параметром масив адрес, у якому адреси можуть повторюватись, і повертає масив адрес без повторень;
   - перевизначте метод *toString()* для виведення адреси англійською мовою. При перекладі на англійську мову російські букви мають транслітеруватись, наприклад, «Chernigiv» замість «Чернігів».
4. Напишіть метод для сортування списку адрес за алфавітом – спочатку за назвою міста, потім за вулицею.
5. Напишіть метод, який у колекції *Map* підраховує кількість значень (*не ключів*) без урахування повторень.

Варіант 7

3. Створіть клас «Дата». Внутрішнє представлення дати – три цілі

числа, що задають день, місяць і рік. Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):

- для створення дати за заданими трьома цілими числами (з перевіркою на неприпустимі значення);
- метод для знаходження дати, наступної за датою (з урахуванням кількості днів у місяці: наприклад, наступною датою за 30 числом місяця може бути 31 або 1 число);
- метод знаходження кількості днів, які розділяють дві задані дати.

Створіть похідний клас, у якому:

- додайте метод визначення дня тижня;
- перевизначте метод *toString()* для виведення дати у форматі «дд.мм.рррр – день_тижня».

4. Напишіть метод для сортування списку дат спочатку за роком, потім за місяцем.
5. Напишіть метод, який у колекції *Map* підраховує кількість значень (*не ключів*) без урахування повторень.

Варіант 8

3. Створіть клас «Еліпс». Внутрішнє представлення еліпса – дійсні координати вершин описаного навколо нього прямокутника. Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):
   - для знаходження координат центру еліпса, площі еліпса;
   - для знаходження відстані між центрами двох еліпсів.

   Створіть похідний клас «окружність», у якому:
   - перевизначте метод *toString()* для виведення окружності у вигляді «Окружність із центром (*x, y*) та радіусом *r*»;
   - додайте метод, що генерує масив точок, що лежать на окружності (кількість точок передається як параметр);
   - додайте метод перевірки, чи лежить задана точка всередині окружності.
4. Напишіть метод для сортування списку еліпсів за величиною їх площі.
5. Напишіть метод, що приймає як параметри дві колекції *Map* і видаляє з першої колекції все ключі, які зустрічаються у другій колекції.

Варіант 9

3. Створіть клас «Час» для зберігання значення часу доби. Внутрішнє представлення часу – три цілих числа, які визначають години, хвилини і секунди. Реалізуйте методи класу (статичні) і методи

реалізації (нестатичні):

- для створення об’єкта за заданими трьома цілими числами (з перевіркою на неприпустимі значення);
- метод порівняння об’єктів на рівність, метод визначення кількості секунд, що пройшли з початку доби;
- метод знаходження кількості секунд, які розділяють два задані об’єкти часу.

Створіть похідний клас, у якому:

- додайте метод для визначення відповідного часу в іншому часовому поясі (номер поясу передається як параметр);
- перевизначте метод *toString()* для виведення часу в текстовому вигляді, наприклад, «десять годин двадцять п’ять хвилин» замість «10:25:00».

4. Напишіть метод для сортування списку значень часу спочатку за кількістю годин, потім за кількістю хвилин.
5. Напишіть метод, який виводить вміст колекції *Map* (тобто пари ключ– значення) в алфавітному порядку значень, тобто сортує записи за значенням (*не за ключем*).

Варіант 10

3. Створіть клас «Трикутник». Внутрішнє представлення трикутника – цілочисельні координати його вершин. Реалізуйте методи класу (статичні) і методи реалізації (нестатичні) для:
   - знаходження площі трикутника, знаходження довжини його сторін;
   - перевірки, чи лежить задана точка всередині трикутника.

   Створіть похідний клас, у якому:
   - перевизначте метод *toString()* для виведення трикутника в вигляді трьох чисел – довжин його сторін;
   - додайте метод перевірки, чи перетинається трикутник із іншим заданим трикутником.
4. Напишіть метод для сортування списку трикутників за величиною найдовшої сторони.
5. Напишіть метод, що приймає як параметр колекцію *Map* і будує нову колекцію *Map*, у якій ключі і значення міняються місцями. Всі повторювані значення з вихідної колекції (які не вдалося зберегти в новій колекції) потрібно вивести окремо.

Варіант 11

3. Створіть клас «Ламана лінія». Внутрішнє представлення лінії – список (*ArrayList*) цілочисельних координат її вузлових точок. Метод *toString()* повинен виводити лінію у вигляді «лінія з точки (x_1, y_1) до точки (x_n, y_n)». Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):
    - для додавання і видалення вузлів (у довільному місці лінії);
    - для знаходження загальної довжини лінії;
    - для виведення в рядок послідовності вузлових точок лінії.

   Створіть похідний клас, у якому:
    - перевизначте метод *toString()* для виведення лінії у вигляді: «лінія довжини *x* з точки (x_1, y_1) до точки (x_n, y_n)»;
    - додайте метод перевірки, чи не виходить лінія за межі заданого прямокутника.
4. Напишіть метод для сортування списку ліній за кількістю вузлів у ній. Порядок сортування – від коротких ліній до довгих.
5. Напишіть метод, що приймає як параметри дві колекції *Map* і видаляє з першої колекції все ключі, які зустрічаються у другій колекції.

Варіант 12

3. Створіть клас «Автобусний маршрут». Внутрішнє представлення маршруту – список (*List*) або масив назв зупинок (пунктів, якими проходить маршрут), а також масив інтервалів за часом між сусідніми зупинками. Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):
    - для додавання нових зупинок;
    - для виведення маршруту в рядок у вигляді «Початковий_пункт – пункт1 – Пункт2 ... Кінцевий_пункт».

   Створіть похідний клас, у якому:
    - додайте метод для знаходження часу проїзду між будь-якими двома заданими зупинками;
    - перевизначте метод *toString()* для виведення маршруту у вигляді «Маршрут від Початковий_пункт до Кінцевий_пункт займає X хвилин».
4. Напишіть метод для сортування списку маршрутів за часом проїзду маршрутом від початку до кінця. Порядок сортування – від коротких маршрутів до довгих.
5. Напишіть метод, що приймає як параметри дві колекції *Map* і створює нову колекцію *Map*, до якої потрапляють тільки ті ключі, які

присутні в першій колекції і відсутні у другій.

Варіант 13

3. Створіть клас «Користувач соціальної мережі». Внутрішнє представлення користувача: ім’я, логін, пароль, дата народження (*java.util.Date*), масив постів користувача (де кожен пост – це об’єкт із двома властивостями: дата публікації та текст). Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):
    - для визначення віку користувача (цілого числа років) на поточну дату;
    - для визначення *рейтингу* користувача як кількості його постів, зроблених за останній місяць;
    - метод *toString()*, який видає інформацію користувача у форматі «ім’я – логін».

    Створіть похідний клас, у якому:
    - додайте статичний метод визначення правильності логіна і пароля (логін має складатися з букв і цифр і починатися з великої літери; пароль має бути не коротшим за 8 символів, обов’язково включати цифри і великі та маленькі літери);
    - перевизначте метод *toString()* для виведення інформації про користувача у форматі «логін – вік – текст останнього за часом посту».
4. Напишіть метод для сортування списку користувачів спочатку за датою останнього посту, потім за ім’ям (за алфавітом).
5. Напишіть метод, який виводить вміст колекції *Map* (тобто пари ключ– значення) в алфавітному порядку значень, тобто сортує записи за значенням (*не за ключем*).

Варіант 14

3. Створіть клас «Колір». Внутрішнє представлення кольору – три цілі числа в діапазоні [0; 100], що задають відсоток яскравості червоної, зеленої і синьої складових. Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):
    - для створення кольору за заданими трьома цілими числами в діапазоні [0; 255];
    - для створення кольору за трьома дійсними числами в діапазоні [0; 100] або [0; 1];
    - метод, що за заданими двома кольорами *a* і *b* створює градієнт між ними, тобто новий масив об’єктів-кольорів, у якому

перший колір дорівнює *a*, останній дорівнює *b*, а всі інші є проміжними кольорами;

- метод визначення яскравості кольору, а також метод порівняння кольорів на «більше» – «менше» за значенням яскравості кольору.

Створіть похідний клас, у якому:

- додайте метод для перетворення кольору в модель *HSV*;
- додайте метод, який у масиві об'єктів-кольорів знаходить той колір, що є найближчим до заданого (за значенням параметра *H*, «*hue*»);
- перевизначте метод *toString()* для виведення кольору у форматі *HSV*.

4. Напишіть метод для сортування списку кольорів за величиною яскравості – від темних до світлих.
5. Напишіть метод, що приймає як параметри дві колекції *Map* і створює нову колекцію *Map*, до якої потрапляють тільки ті ключі, які присутні в обох вхідних колекціях.

Варіант 15

3. Створіть клас «Адреса». Внутрішнє представлення – індекс (5-значне ціле число), місто, вулиця, номер будинку, номер квартири. Реалізуйте методи класу (статичні) і методи реалізації (нестатичні):
   - метод створення об'єкта-адреси на основі рядка виду «індекс, вулиця, будинок, квартира місто», де індекс і квартира можуть бути не вказані, наприклад: «34668 Вінниця, вул. Інженерна, 25», «Вінниця, вул. Інженерна, 25, кв. 32»;
   - метод, що порівнює дві адреси та визначає, яка частина адреси збігається (повертає спільну частину адреси у вигляді рядка);
   - метод перетворення адреси в рядок виду «04123 просп. Перемоги, 35, м. Київ».

   Створіть похідний клас, у якому:
   - додайте метод, що приймає як параметри масив адрес, у якому адреси можуть повторюватись, і повертає масив адрес без повторень;
   - перевизначте метод *toString()* для виведення адреси на англійській мові. При перекладі на англійську мову російські букви мають транслітеруватись, наприклад, «Chernigiv» замість «Чернігів».

4. Напишіть метод для сортування списку адрес за довжиною текстового представлення адреси *(toString())*. Порядок сортування – в напрямку зменшення довжини рядка.
5. Напишіть метод, що приймає як параметр колекцію *Map* і будує нову колекцію *Map*, в якій ключі і значення міняються місцями. Всі повторювані значення з вихідної колекції (які не вдалося зберегти в новій колекції) потрібно вивести окремо.

Додаткові завдання

6. Для свого варіанту індивідуального завдання опишіть приклади ситуацій, коли некоректна реалізація методів *Object* для вашого класу може призводити до безповоротної втрати об'єктів цього класу при спробі зберегти їх у колекціях.
7. Використовуючи існуючий клас *HashMap*, створіть власну реалізацію інтерфейсу *Map*, яка за будь-якого способу проходу нею буде гарантовано видавати ключі в тому ж порядку, у якому вони додавалися в колекцію.

Додаткові відомості.
Колекції JCF

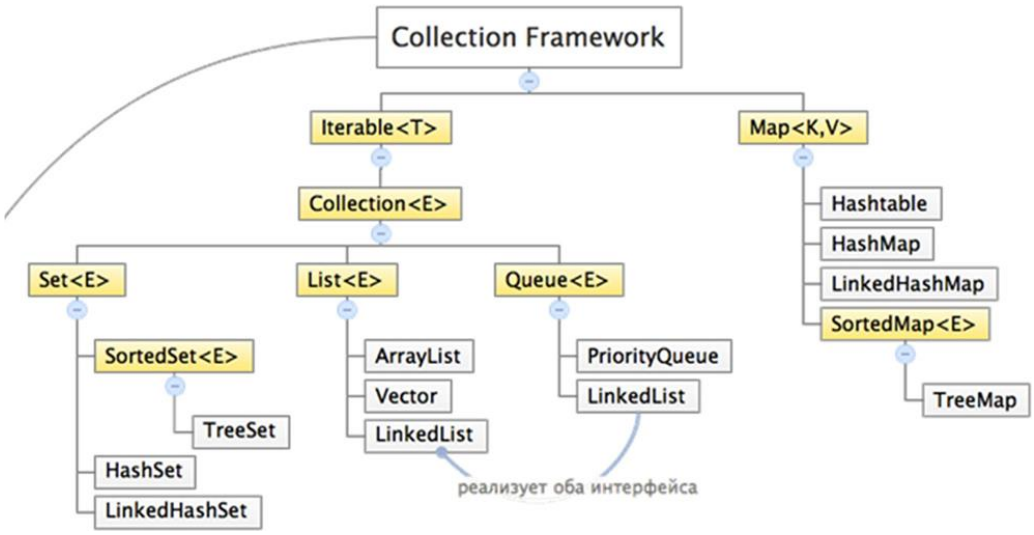


Рис. 1. Ієрархія інтерфейсів JCF і їх реалізацій

Виключення

Загальні класи виключень (java.util)

Таблиця 6

Виключення, що перевіряються (Exception)

ClassNotFoundException	Клас не знайдено (наприклад, бібліотека, в якій він реалізований, недоступна під час виконання програми)
CloneNotSupportedException	Спроба клонувати об’єкт, який не реалізує інтерфейс Cloneable
IllegalAccessException	Заборонено доступ до класу
InstantiationException	Спроба створити об’єкт абстрактного класу або інтерфейсу
InterruptedException	Потік виконання програми перерваний іншим потоком
NoSuchFieldException	Запитуване поле не існує
NoSuchMethodException	Запитуваний метод не існує
ReflectiveOperationException	Виключення, пов’язане з рефлексією

Таблиця 7

Виключення, що не перевіряються (RuntimeException)

ArithmeticException	Арифметична помилка, наприклад, ділення на нуль
ArrayIndexOutOfBoundsException	Вихід індексу за межі масиву (неприпустимий номер комірки)
ArrayStoreException	Присвоєння елементу масиву об’єкта несумісного типу
ClassCastException	Неправильне приведення типу. Клас об’єкта не відповідає типу, до якого він приводиться
EnumConstantNotPresentException	Спроба використання невизначеного значення перерахування (enum)
IllegalArgumentException	Використання неправильного значення для аргументу при виклику методу
IllegalStateException	Некоректний стан об’єкта, за якого виклик методу неможливий
IllegalThreadStateException	Запитувана операція несумісна з поточним станом потоку

InterruptedException	Сталося переривання потоку виконання в багатопотоковому додатку. Переривання може вивести потік зі стану очікування (wait) або сну (sleep)
IndexOutofBoundsException	Значення індексу вийшло за допустимі межі. Має два дочірні класи: ArrayIndexOutOfBoundException і StringIndexOutOfBoundException
NegativeArraySizeException	Спроба створити масив від'ємного розміру
NullPointerException	Спроба звернутися до об'єкта за посиланням null
NumberFormatException	Помилка при перетворенні рядка в числовий формат
SecurityException	Порушення безпеки
StringIndexOutOfBounds	Вихід індексу за межі рядка (неприпустимий номер символу)
TypeNotPresentException	Клас, запитаний за ім'ям (у вигляді рядка), не знайдено. На відміну від ClassNotFoundException, це виключення не перевіряється
UnsupportedOperationException	Виклик непідтримуваної операції, наприклад, якщо об'єкт підтримує (тобто реалізує) не всі методи загального інтерфейсу

Правила коректної обробки виключень

1. Обробляйте виключення *тільки* в межах відповідальності поточного класу і поточного методу одним із таких способів:
   1.1. Обробити на цьому рівні абстракції і заново викинути виключення того ж класу: *catch (Type e) {... throw e; };*
   1.2. Обробити на цьому рівні абстракції і викинути виключення іншого класу: *throw new SomeException("message", e);* – тобто виключення-обгортку вищого рівня;
   1.3. Не викидати нових виключень – *тільки* якщо на поточному

рівні проблема вирішується повністю.

2. Обробляйте кожен клас виключень окремо.
3. На початку мають знаходитися обробники для більш специфічних класів виключень, потім – для більш загальних.
4. Останній обробник – *catch (Exception e)* – може обробляти *всі* виключення, які не були оброблені раніше. Використовується нечасто, оскільки суперечить правилу 2.
5. Уникайте порожніх обробників *(catch (Type e) {})*. Кожне виключення вимагає явних дій.
6. Якщо виключення ігнорується *(catch (Type e) {})*, то задокументуйте, чому.
7. Використовуйте блок *finally* тільки для звільнення ресурсів (*close(), release()* і т. д.).
8. Обробляйте виключення, що виникають у блоках *catch()* і *finally*.

Лабораторна робота №4

Тема: Робота з файловою системою.

Мета: навчитися використовувати можливості *Java* по роботі з файловою системою для зберігання даних програми на диску, а також для маніпулювання файлами і директоріями. Ознайомитися з технологією серіалізації об’єктів.

Теоретичні питання

1. Потоки введення / виведення. Класи потоків *InputStream, OutputStream, Reader* і *Writer*.
2. Буферизація введення / виведення.
3. Робота з бінарними файлами через *DataInputStream* і *DataOutputStream*.
4. Робота з текстовими файлами за допомогою *PrintStream* і *Scanner*.
5. Серіалізація об’єктів. Інтерфейс *Serializable*. Потоки об’єктів.
6. Робота з файлами і директоріями. Класи *File* і *Files*.

Контрольні питання

1. Коротко опишіть призначення класів-нащадків *InputStream* і *OutputStream*.
2. У чому відмінність між класами *InputStream*, *OutputStream* і класами *Reader*, *Writer*?
3. Як відкрити текстовий файл для додавання записів у кінець файла?

4. Навіщо застосовується буферизація введення / виведення? Які класи її забезпечують? Наведіть приклад.
5. Навіщо застосовується метод *flush()*?
6. Для чого застосовується серіалізація? Які методи включає інтерфейс *Serializable* і як зробити клас здатним до серіалізації?
7. У чому відмінність класів *DataInputStream* і *DataOutputStream* від класів *ObjectInputStream* і *ObjectOutputStream*?
8. Яку інформацію про фото можна отримати з об'єкта *File*?
9. Які операції над файлами і директоріями реалізує клас *File*?
10. Якими способами можна скопіювати файл у *Java*?

Індивідуальні завдання

Загальне завдання

1. Для класів із завдання 3 лабораторної роботи 3 (для базового і для похідного) реалізуйте механізм серіалізації.
2. Напишіть програму, яка:
    - читає набір об'єктів (базового і похідного класу) з CSV файла і поміщає їх до масиву або списку;
    - серіалізує цей масив (або список) у бінарний файл;
    - десеріалізує масив (або список) із бінарного файла.

   Імена файлів задаються у вигляді параметрів при запуску програми.
   Програма має коректно обробляти помилкові дані у вхідному файлі!

Варіант 1

3. Дано файл, що містить список директорій (кожен рядок файла містить повний шлях до певної директорії). Напишіть програму, яка знаходить у цьому списку найбільшу за об'ємом директорію та виводить її вміст на екран.
4. **(Для програмістів)* Напишіть програму, яка вводить назву директорії і маску імені файла (наприклад, масці *«a.?xt»* відповідають імена файлів *a.txt* і *bba.xxt*), а потім перейменовує всі файли директорії, імена яких відповідають масці. При перейменуванні розширення файла зберігається, а ім'я змінюється на номер файла в алфавітному порядку.

Варіант 2

3. Напишіть програму, яка вводить ім'я директорії та підраховує в цій директорії кількість файлів кожного типу. Тип файла визначається за його розширенням (наприклад, .java, .class, .txt і т. д.).

4. *(*Для програмістів*) Напишіть програму, яка вводить імена двох директорій і регулярний вираз, а потім копіює з першої директорії до другої всі файли, імена яких відповідають регулярному виразу. Програма повинна обробляти всі вкладені директорії. При копіюванні структура вкладених директорій має зберігатися.

Варіант 3

3. Напишіть програму, яка перейменовує всі файли з розширенням .txt із заданої директорії, даючи їм імена 1.txt, 2.txt, 3.txt і так далі по порядку. Виведіть старі імена файлів на екран і в текстовий файл.
4. *(*Для програмістів*) Напишіть програму, яка порівнює вміст двох заданих директорій і виводить список файлів, які відрізняються. Для порівняння файлів використовуйте ім’я, розмір і дату останньої зміни файла. Програма має обробляти всі вкладені директорії.

Варіант 4

3. Напишіть програму, яка копіює з першої заданої директорії до другої всі файли, змінені за останні 3 дні. Виведіть імена файлів, що копіюються, на екран і в текстовий файл.
4. *(*Для програмістів*) Напишіть програму, яка шукає задану фразу в усіх файлах директорії. Для кожної знайденої фрази виведіть назву файла, номер рядка та позицію в рядку. Програма має обробляти всі вкладені директорії. Виведені дані мають бути відсортовані за повним ім’ям файла.

Варіант 5

3. Напишіть програму, яка для заданої директорії знаходить максимальну глибину вкладеності піддиректорій. Виведіть імена всіх піддиректорій на екран і в текстовий файл.
4. *(*Для програмістів*) Напишіть програму, яка для заданої директорії визначає, яку частину загального обсягу (у відсотках) займають файли кожного типу. Тип файла визначається його розширенням. Для зберігання даних використовуйте колекцію HashMap.

Варіант 6

3. Напишіть програму, яка для заданої директорії визначає кількість файлів з урахуванням вкладених директорій. Виведіть імена всіх цих файлів на екран і в текстовий файл.
4. *(*Для програмістів*) Напишіть програму, яка шукає адреси електронної пошти в усіх текстових файлах заданої директорії і виводить усі знайдені адреси на екран. При виведенні адреси

відсортуйте за ім’ям домену.

Вказівка. Визначити, чи є файл текстовим, можна за його розширенням; список допустимих розширень текстових файлів (наприклад, txt, htm, html, csv, java, ...) програма повинна читати з окремого файла.

Варіант 7

3. Напишіть програму, яка знаходить кількість рядків у кожному .txt-файлі заданої директорії. Ім’я директорії вводити з клавіатури. Результати виведіть на екран і в текстовий файл.
4. *(*Для програмістів*) Напишіть програму, яка для заданої директорії видаляє директорії найнижчого рівня вкладеності, а файли з них переміщує на один рівень вище.

Варіант 8

3. Задано ім’я файла name і ціле число N. Напишіть програму, яка розбиває заданий файл на фрагменти довжиною не більше N байт. Кожен фрагмент потрібно зберегти в окремому файлі з ім’ям *name-xx*, де xx – порядковий номер фрагмента. Виведіть імена створюваних файлів та їх розміри на екран, а також у текстовий файл.
4. *(*Для програмістів*) Напишіть програму, яка за заданим ім’ям файла-фрагмента name-xx шукає інші фрагменти і об’єднує їх в один файл з ім’ям *name*. Якщо якийсь із фрагментів відсутній, видайте повідомлення про помилку.

Варіант 9

3. Дано файл, що містить список директорій (кожен рядок файла містить повний шлях до певної директорії). Вивести назви всіх файлів із цих директорій одним списком в алфавітному порядку.
4. *(*Для програмістів*) Напишіть програму, яка зменшує розмір папки проекту *Visual C++* (або *Visual C#*), видаляючи з неї всі файли, без яких проект може бути успішно відкритий і заново компільований. Програма повинна також зберігати всі файли готових програм із підпапок *Release*.

Варіант 10

3. Напишіть програму, яка вводить три імені директорій (A, B і C) та об’єднує дві директорії A і B в одну директорію C. Якщо директорія C вже існує, то виведіть попередження. Якщо директорії A і B містять файли з однаковими іменами, то в C потрібно помістити тільки одну, найновішу версію цього файла.

4. *(*Для програмістів*) Напишіть програму, яка сканує всі файли вихідних кодів *Java* в заданій директорії і визначає, скільки відсотків коду складають коментарі. Необхідно враховувати і однорядкові, і багаторядкові коментарі. Програма повинна обробляти всі вкладені директорії.

Варіант 11

3. Напишіть програму, яка копіює з першої заданої директорії до другої всі файли, змінені за останні 3 дні. Виведіть імена файлів, що копіюються, на екран і в текстовий файл.
4. *(*Для програмістів*) Напишіть програму, яка порівнює вміст двох заданих директорій і виводить список файлів, які розрізняються. Для порівняння файлів використовуйте ім'я, розмір і дату останньої зміни файла. Програма повинна обробляти всі вкладені директорії.

Варіант 12

3. Задані ім'я файла name і ціле число N. Напишіть програму, яка розбиває заданий файл на фрагменти довжиною не більше N байт. Кожен фрагмент потрібно зберегти в окремому файлі з ім'ям name-xx, де xx – порядковий номер фрагмента. Виведіть імена створюваних файлів і їх розміри на екран, а також у текстовий файл.
4. *(*Для програмістів*) Напишіть програму, яка за заданим ім'ям файла-фрагмента name-xx шукає інші фрагменти та об'єднує їх в один файл з ім'ям *name*. Якщо якийсь із фрагментів відсутній, видайте повідомлення про помилку.

Варіант 13

3. Напишіть програму, яка знаходить кількість рядків у кожному .txt-файлі заданої директорії. Ім'я директорії потрібно вводити з клавіатури. Результати виведіть на екран і в текстовий файл.
4. *(*Для програмістів*) Напишіть програму, яка вводить імена двох директорій і регулярний вираз, а потім копіює з першої директорії до другої всі файли, імена яких відповідають регулярному виразу. Програма повинна обробляти всі вкладені директорії. При копіюванні структура вкладених директорій має зберігатися.

Варіант 14

3. Напишіть програму, яка вводить три імені директорій (A, B і C) і об'єднує дві директорії A і B в одну директорію C. Якщо директорія C вже існує, то виведіть попередження. Якщо директорії A і B містять файли з однаковими іменами, то в C потрібно помістити тільки одну,

найновішу версію цього файла.

4. *(*Для програмістів*) Напишіть програму, яка для заданої директорії визначає, яку частину загального обсягу (у відсотках) займають файли кожного типу. Тип файла визначається його розширенням. Для зберігання даних використовуйте колекцію *HashMap*.

Варіант 15

3. Напишіть програму, яка для заданої директорії визначає кількість файлів з урахуванням вкладених директорій. Виведіть імена всіх цих файлів на екран, а також у текстовий файл.
4. *(*Для програмістів*) Напишіть програму, яка шукає адреси електронної пошти в усіх текстових файлах заданої директорії і виводить усі знайдені адреси на екран. При виведенні адреси відсортуйте за ім'ям домену.

   Вказівка. Визначити, чи є файл текстовим, можна за його розширенням; список допустимих розширень текстових файлів (наприклад, txt, htm, html, csv, java, ...) програма повинна читати з окремого файла.

Додаткові завдання

У *Java* для читання і запису zip-архівів застосовуються класи *ZipInputStream* і *ZipOutputStream*. Використовуючи ці класи, напишіть програму, яка може архівувати і розпаковувати набір файлів або директорій. Вхідні дані (тобто набір імен файлів або директорій) програма повинна отримувати у вигляді параметрів командного рядка.

Налаштуйте конфігурацію запуску проекту так, щоб отримати файл *.jar*, який можна запустити в *ОС Windows* подвійним клацанням по ньому. Протестуйте його роботу, викликаючи його з командного рядка і передаючи в командному рядку імена файлів (директорій).

Додаткові відомості

Архітектура *Java IO*

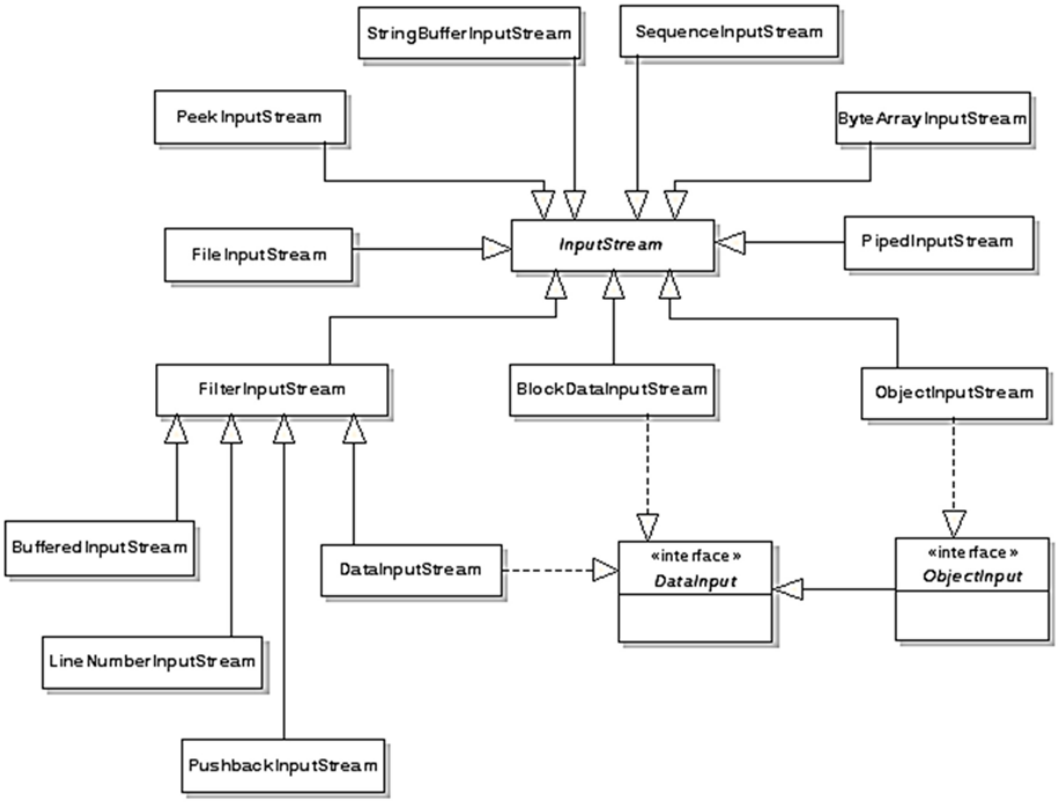


Рис. 2. Ієрархія байтових потоків введення *InputStream*

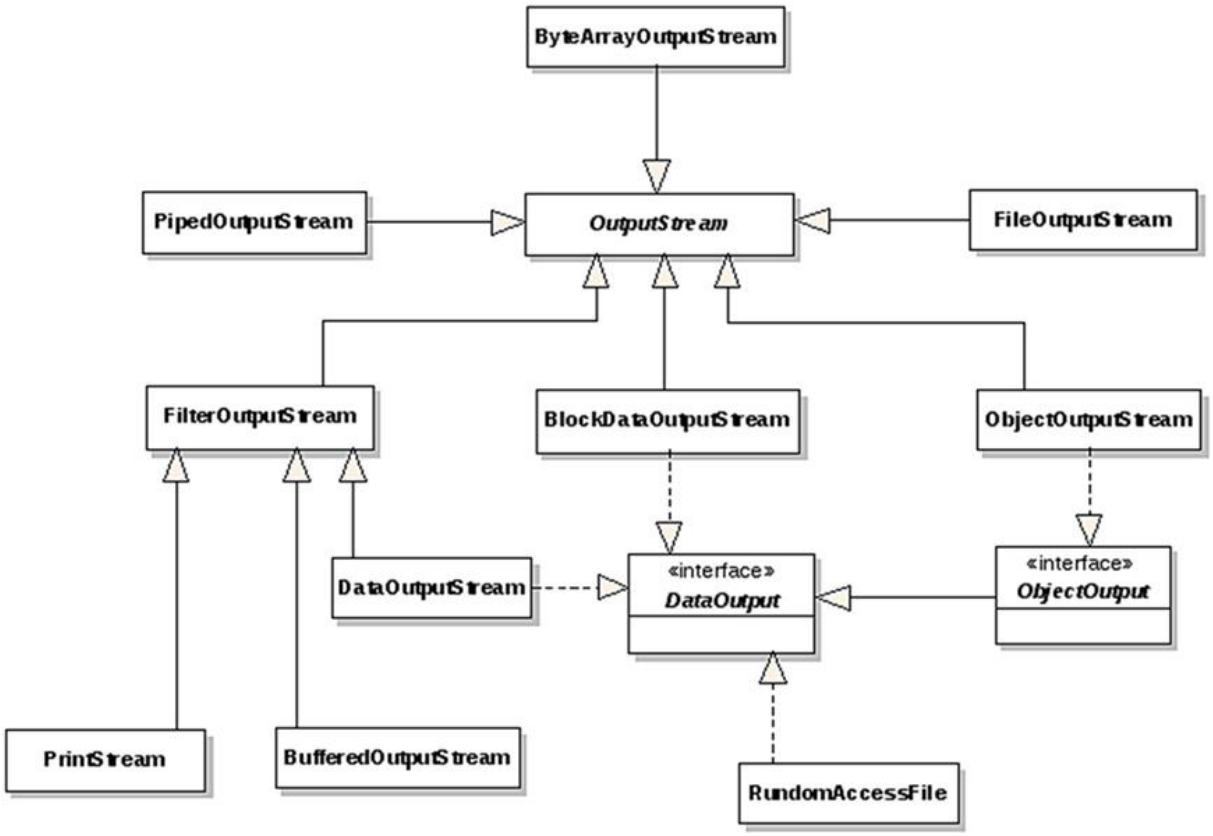


Рис. 3. Ієрархія байтових потоків виведення *OutputStream*

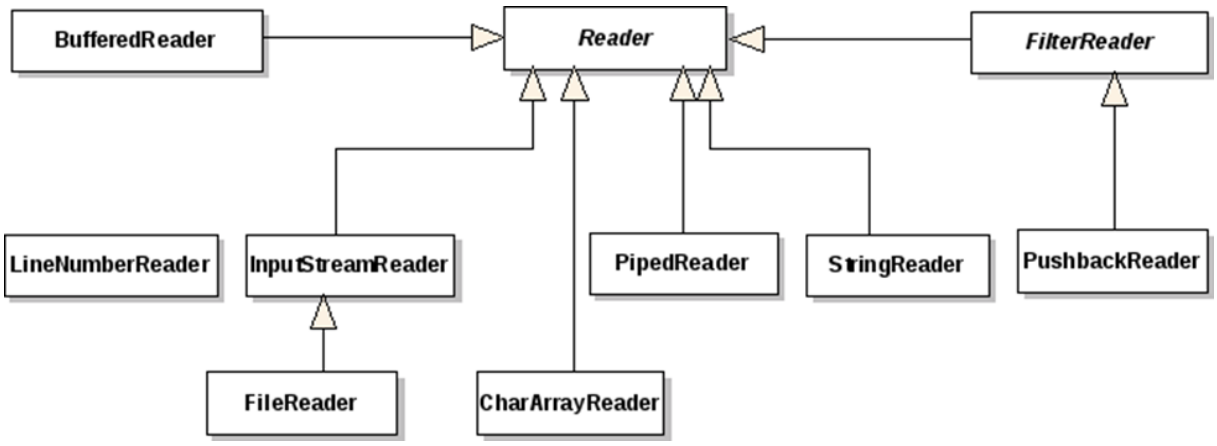


Рис. 4. Ієрархія текстових потоків введення *Reader*

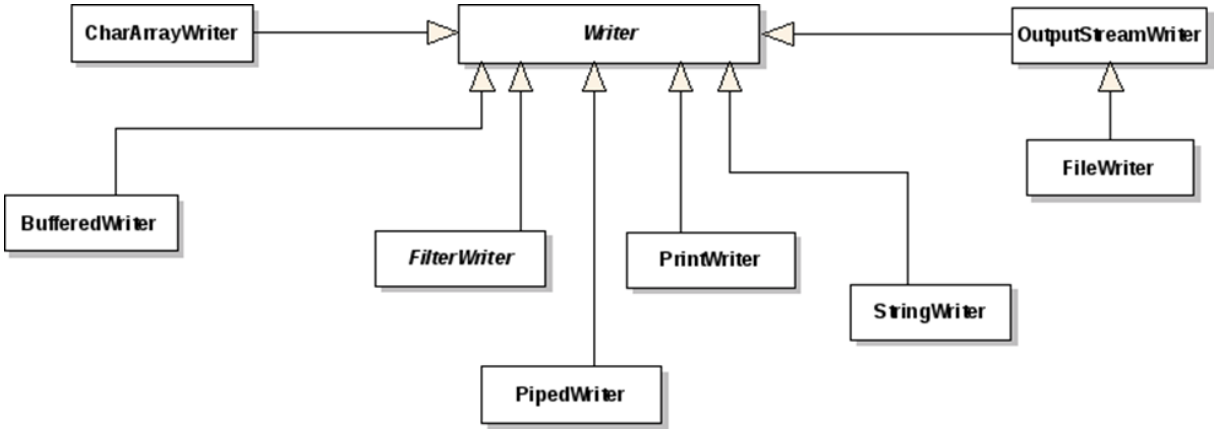


Рис. 5. Ієрархія текстових потоків виведення *Writer*

Лабораторна робота №5

Тема: Розробка графічного віконного інтерфейсу.

Мета: отримати навички створення додатків із графічним інтерфейсом користувача. Вивчити можливості пакета Swing.

Теоретичні питання

1. Можливості та особливості бібліотек *Swing* і *AWT*.
2. Поняття і принципи *Usability*.
3. *GUI*-редактор *IDEA* і його можливості.
4. Клас *JFrame*.
5. Класи графічних компонент *JPanel, JLabel, JButton* (і його нащадки), *JTextBox, JTextArea, JComboBox, JScrollPane, JSlider, JTabbedPane*.
6. Допоміжні класи: *Color, Dimension, Font*.
7. Класи розміщень (*layouts*).
8. Вирівнювання елементів. Ручне і автоматичне задання розмірів

елементів.

9. Події. Механізм обробки подій *Swing*.
10. Меню і панелі інструментів. Обробка їх подій.

Контрольні питання

1. Перерахуйте основні відмінності бібліотек *Swing* і *AWT*. Яку з них краще використовувати і чому?
2. Перерахуйте основні стандарти, що регулюють розташування елементів екранної форми.
3. Наведіть приклад коду, що створює вікно і кнопку на ньому.
4. Як за допомогою методів класу JFrame можна динамічно змінювати зовнішній вигляд вікна?
5. У якому вигляді зберігається графічний дизайн форми, створений *GUI*-редактором?
6. Коротко опишіть призначення та основні методи класів графічних компонентів: *JPanel, JLabel, JButton, JTextBox, JTextArea, JComboBox*.
7. Які класи розміщень (layouts) реалізовані в *Swing* і чим вони відрізняються?
8. Опишіть призначення класів *Color, Dimension, Font* і наведіть приклади їх використання.
9. Як можна задати вирівнювання компонента, його мінімальний і максимальний розміри?
10. Як у *Swing* обробляються події *GUI*?
11. Що таке слухач (*listener*) і які стандартні класи слухачів є у *Swing*?
12. Чим відрізняється *Adapter* від *Listener*?
13. Як використовувати абстрактні класи *MouseAdapter* і *ActionListener* для обробки натискання кнопки (*JButton*) мишкою? Наведіть приклади.
14. Як можна обробити поєднання клавіш *Ctrl-Alt-F5* за допомогою *KeyAdapter*?
15. Опишіть основні методи класів *JMenu* і *JToolbar*. Як найлегше додати однакові обробники і для пункту меню, і для відповідної кнопки панелі інструментів?

Індивідуальні завдання

Загальне завдання

1. Створіть вікно входу до системи. Вікно повинне містити:
    - текстові поля для введення логіна і пароля з підписами (введений пароль не має бути видно);

- кнопки «Увійти», «Реєстрація» і «Скасувати».

Вікно має відповідати стандартам розташування елементів управління.

При виборі «Скасувати» робота програми завершується.

При виборі «Увійти» виконується перевірка правильності логіна і пароля і видається повідомлення про результат перевірки.

При виборі «Реєстрація» вікно переходить у режим реєстрації нового користувача (змінюється заголовок вікна, з'являється нове поле для підтвердження пароля, ховається кнопка «Увійти»).

Для зберігання логінів і паролів у пам'яті використовуйте колекцію *Map*.

Реалізуйте серіалізацію цих даних до файла.

(**Для програмістів*) Паролі в пам'яті і на диску зберігайте тільки в зашифрованому вигляді. Для шифрування використовуйте клас *MessageDigest* і алгоритм *SHA-512*.

2. Створіть додаток (за варіантом) з віконним інтерфейсом. Вікно додатка має містити меню і панель інструментів. Усі дії мають виконуватися кількома способами: через меню, кнопки панелі інструментів, гарячі клавіші.

   Реалізуйте збереження даних програми у файл і завантаження з файла.

   Перелік обов'язкових пунктів меню:

   - *File: New, Open, Save, Save as, Exit;*
   - *Help: About.*

   Варіант 1

3. Гра «Пазл». На початку гри на формі у довільний спосіб розміщуються прямокутні фрагменти зображення-пазла. Далі користувач може виділяти і переміщати фрагменти пазла за допомогою мишки, а також за допомогою клавіш-стрілок. Гра завершується, коли всі фрагменти пазла розміщені у правильному порядку.

   Для відображення кожного фрагмента використовуйте *JLabel* або *JButton* із зображенням. При виділенні у фрагмента має з'являтися червона рамка. Після кожного ходу програма повинна автоматично визначати, чи завершена гра.

   (*Для не програмістів*) Фрагменти картинки можна підготувати заздалегідь у вигляді окремих файлів, тобто набір фрагментів не змінюється, змінюється тільки їх

розташування у вікні програми.

(**Для програмістів*) Реалізуйте можливість вибору вихідної картинки, кількості фрагментів пазла, а також автоматичне розділення картинки на фрагменти.

Варіант 2

3. Гра «Реверс». Ігрове поле 66 складається з клітинок, кожна з яких забарвлена в один із двох кольорів – білий або синій. На початку гри клітинки фарбуються у випадковий спосіб. У процесі гри гравець може змінювати колір клітинок, натискаючи на них. При натисканні на клітинку одночасно змінюється колір усіх клітинок у її рядку та в її стовпці. Мета гри – пофарбувати всі клітинки в один колір.

   Вікно повинне мати текстовий напис, що повідомляє, скільки клітинок кожного кольору наразі є на полі. Після кожного ходу програма повинна автоматично визначати, чи завершена гра.

   Для клітинок поля використовуйте кнопки з картинками. У файл зберігайте поточний стан гри.

   (**Для програмістів*) Реалізуйте можливість вибору розміру ігрового поля, реалізуйте режим ручного завдання кольорів клітинок на початку гри.

Варіант 3

3. Гра «Сапер». Правила гри аналогічні до стандартної гри Windows. Використовуйте ігрове поле 1010. Вікно повинне мати текстовий напис, що повідомляє, скільки «мін» залишилося не знайденими. Після кожного ходу програма повинна автоматично визначати, чи завершена гра і чи виграв гравець.

   Для комірок використовуйте кнопки з картинками. У файл зберігайте поточний стан гри.

   (** Для програмістів*) Реалізуйте можливість вибору розміру ігрового поля і кількості «мін». Реалізуйте автоматичне каскадне розкриття сусідніх комірок при розкритті комірки з нульовим значенням.

Варіант 4

3. Програма «Щоденник». У кожен момент часу вікно додатка має відображати один запис щоденника, який включає:
   - назву події (текстове поле);
   - докладний опис (багаторядкове текстове поле);
   - дату, час (текстові поля);
   - прапорець «важлива подія».

У вікні також мають бути кнопки «Наступна», «Попередня» для переходу між подіями.

Має бути можливість редагування всіх атрибутів події. За кнопкою «Застосувати» зміни повинні зберігатися в пам’яті. У файл зберігайте всі записи щоденника.

(**Для програмістів*) Реалізуйте перевірку правильності введення дати і часу. Реалізуйте фільтр записів за датою (для пошуку задається діапазон дат), а також за довільною фразою в тексті опису події.

Варіант 5

3. Програма «Адресна книга». У кожен момент часу вікно додатка має відображати один запис адресної книги, який включає:
    - ім’я людини (текстове поле);
    - її телефон та e-mail (текстові поля);
    - групу (список, що випадає), наприклад, «Друзі», «Знайомі»;
    - прапорець «доданий до обраних контактів»;
    - день народження (текстове поле).

   У вікні мають бути кнопки «Наступний», «Попередній» для переходу між записами.

   Має бути можливість редагування всіх полів даних. За кнопкою «Застосувати» ці зміни мають зберігатися в пам’яті.

   У файл зберігайте всі записи адресної книги.

   (**Для програмістів*) Реалізуйте перевірку правильності введення телефону та e-mail. Реалізуйте можливість сортування записів за різними атрибутами.

Варіант 6

3. Програма «Замовлення». У кожен момент часу вікно додатка має відображати інформацію про одне замовлення, яка включає:
    - дату замовлення, найменування товару, ім’я замовника і його телефон (текстові поля);
    - категорію товару (список, що випадає), наприклад, «Ноутбук», «Нетбук», «Планшет»;
    - кількість (текстове поле, допустимо тільки ціле число);
    - адресу замовника (багаторядкове текстове поле);
    - прапорці «Терміновий» і «Оплата готівкою».

   У вікні мають бути кнопки «Наступний», «Попередній» для переходу між записами.

   Має бути можливість редагування всіх полів даних. За кнопкою

«Застосувати» ці зміни мають зберігатися в пам’яті. У файл зберігайте всі замовлення.

(**Для програмістів*) Реалізуйте вибір найменування товару зі списку, який автоматично формується залежно від обраної категорії товару. Реалізуйте фільтр записів за категорією товару, а також за ознаками «Терміновий» і «Оплата готівкою».

Варіант 7

3. Гра «Морський бій». Користувач грає проти комп’ютера. Комп’ютер розміщує «кораблі» на полі у випадковий спосіб, а гравець повинен їх знайти. Використовуйте ігрове поле 8*8. Гравець виграє, якщо він знаходить усі «кораблі» противника, і програє, якщо не знаходить їх за 25 ходів. У вікні мають бути текстові написи, скільки «кораблів» наразі «вбито», «поранено» і скільки ще не знайдено. Після кожного ходу програма має автоматично визначати, чи завершена гра.

   Для комірок використовуйте кнопки з картинками. У файл зберігайте поточний стан гри.

   (**Для програмістів*) Реалізуйте можливість гри двох гравців, зокрема й режим початкової розстановки «кораблів» гравцями вручну.

Варіант 8

3. Гра «Хрестики-нулики». Ігрове поле 5*5 складається з клітинок, у кожній із яких може стояти хрестик, нулик або порожньо. Гравці по черзі ставлять хрестики і нулики в порожні клітинки поля. Гра завершується, коли один з гравців має три хрестика або нулика в ряд. Вікно повинне мати текстовий напис, що повідомляє, який гравець повинен ходити. Після кожного ходу програма повинна автоматично визначати, чи завершена гра, і хто виграв. Для клітинок поля використовуйте кнопки з картинками. У файл зберігайте поточний стан гри.

   (**Для програмістів*) Реалізуйте можливість вибору розміру ігрового поля. Реалізуйте можливість гри з комп’ютером.

Варіант 9

3. Гра «Реверс». Ігрове поле 7*7 складається з клітинок, кожна з яких забарвлена в один із двох кольорів – білий або синій. На початку гри клітинки фарбуються у випадковий спосіб. У процесі гри гравець може змінювати колір клітинок, натискаючи на них. При натисканні на клітинку одночасно змінюється колір усіх клітинок у її рядку та в її стовпці. Мета гри – пофарбувати всі клітинки в один колір.

Вікно повинне мати текстовий напис, що повідомляє, скільки клітинок кожного кольору наразі є на полі. Після кожного ходу програма повинна автоматично визначати, чи завершена гра.

Для клітинок поля використовуйте кнопки з картинками. У файл зберігайте поточний стан гри.

(**Для програмістів*) Реалізуйте можливість вибору розміру ігрового поля, реалізуйте режим ручного задання кольорів клітинок на початку гри.

Варіант 10

3. Гра «Сапер». Правила гри аналогічні до стандартної гри Windows. Використовуйте ігрове поле 15*15. Вікно повинне мати текстовий напис, що повідомляє, скільки «мін» залишилося не знайденими. Після кожного ходу програма повинна автоматично визначати, чи завершена гра і чи виграв гравець.

   Для комірок використовуйте кнопки з картинками. У файл зберігайте поточний стан гри.

   (**Для програмістів*) Реалізуйте можливість вибору розміру ігрового поля і кількості «мін». Реалізуйте автоматичне каскадне розкриття сусідніх комірок при розкритті комірки з нульовим значенням.

Варіант 11

3. Програма «Щоденник». У кожен момент часу вікно додатка має відображати один запис щоденника, який включає:
   - назву події (текстове поле);
   - докладний опис (багаторядкове текстове поле);
   - дату, час (текстові поля);
   - прапорець «важлива подія».

   У вікні також мають бути кнопки «Наступна», «Попередня» для переходу між подіями.

   Має бути можливість редагування всіх атрибутів події. За кнопкою «Застосувати» зміни повинні зберігатися в пам’яті. У файл зберігайте всі записи щоденника.

   (**Для програмістів*) Реалізуйте перевірку правильності введення дати і часу. Реалізуйте фільтр записів за датою (для пошуку задається діапазон дат), і за довільною фразою в тексті опису події.

Варіант 12

3. Програма «Адресна книга». У кожен момент часу вікно додатка має відображати один запис адресної книги, який включає:

- ім’я людини (текстове поле);
- його телефон та e-mail (текстові поля);
- групу (список, що випадає), наприклад, «Друзі», «Знайомі»;
- прапорець «доданий до обраних контактів»;
- день народження (текстове поле).

У вікні мають бути кнопки «Наступний», «Попередній» для переходу між записами.

Має бути можливість редагування всіх полів даних. За кнопкою «Застосувати» ці зміни повинні зберігатися в пам’яті. У файл зберігайте всі записи адресної книги.

(**Для програмістів*) Реалізуйте перевірку правильності введення телефону та e-mail. Реалізуйте можливість сортування записів за різними атрибутами.

Варіант 13

3. Програма «Замовлення». У кожен моментчасу вікно додатка має відображати інформацію про одне замовленні, яка включає:
   - дату замовлення, найменування товару, ім’я замовника і його телефон (текстові поля);
   - категорію товару (список, що випадає), наприклад, «Ноутбук», «Нетбук», «Планшет»;
   - кількість (текстове поле, допустиме тільки ціле число)
   - адресу замовника (багаторядкове текстове поле);
   - прапорці «Терміновий» і «Оплата готівкою».

   У вікні мають бути кнопки «Наступний», «Попередній» для переходу між записами.

   Має бути можливість редагування всіх полів даних. За кнопкою «Застосувати» ці зміни повинні зберігатися в пам’яті. У файл зберігайте всі замовлення.

   (**Для програмістів*) Реалізуйте вибір найменування товару зі списку, який автоматично формується залежно від обраної категорії товару. Реалізуйте фільтр записів за категорією товару, а також за ознаками «Терміновий» і «Оплата готівкою».

Варіант 14

3. Гра «Морський бій». Користувач грає проти комп’ютера. Комп’ютер розміщує «кораблі» на полі у випадковий спосіб, а гравець повинен їх знайти. Використовуйте ігрове поле 10*10. Гравець виграє, якщо він знаходить всі «кораблі» противника, і програє, якщо не знаходить їх

за 25 ходів. У вікні мають бути текстові написи, скільки «кораблів» наразі «вбито», «поранено» і скільки ще не знайдено. Після кожного ходу програма повинна автоматично визначати, чи завершена гра.

Для комірок використовуйте кнопки з картинками. У файл зберігайте поточний стан гри.

(**Для програмістів*) Реалізуйте можливість гри двох гравців, зокрема й режим початкової розстановки «кораблів» гравцями вручну.

Варіант 15

3. Гра «Хрестики-нулики». Ігрове поле 66 складається з клітинок, у кожній з яких може стояти хрестик, нулик або порожньо. Гравці по черзі ставлять хрестики і нулики в порожні клітинки поля. Гра завершується, коли один із гравців має три хрестики або нулики в ряд. Вікно повинне мати текстовий напис, що повідомляє, який гравець повинен ходити. Після кожного ходу програма повинна автоматично визначати, чи завершена гра, і хто виграв. Для клітинок поля використовуйте кнопки з картинками. У файл зберігайте поточний стан гри.

   (**Для програмістів*) Реалізуйте можливість вибору розміру ігрового поля. Реалізуйте можливість гри з комп'ютером.

Додаткові відомості

Каркас мінімального GUI-додатка

```
public class MyWnd {     // Клас додатка

    private JFrame frame;     // Головне (і єдине) вікно

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {  //Для створення вікна
            public void run() { // зазвичай запускається
                try {   // окремий потік.
                    MyWnd window = new MyWnd();
                    // Створюємо вікно
                    window.frame.setVisible(true);
                    // Відображаємо його
                } catch (Exception e)
                    { e.printStackTrace();
                }
            }
```

```
            });
        }
public MyWnd() {  initialize(); }          // Конструктор класу додатка

private void initialize() {  // Створення вікна і його компонентів
        frame = new JFrame();   // Створюємо саме вікно
        frame.setBounds(100, 100, 450, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Тут код створення компонентів і додавання для них обробників
        }
}
```

Приклад. Простий текстовий редактор

Редактор дає змогу:

- вводити ім'я файла безпосередньо до текстового поля (JTextField);
- вибирати файл за допомогою стандартного діалогу відкриття файла (JFileChooser) за кнопкою «Open»;
- редагувати вміст файла у багаторядковому текстовому полі (JTextArea);
- зберігати зміни у файлі за кнопкою «ОК»;
- вийти з програми за кнопкою «Cancel», відображаючи попередження перед виходом.

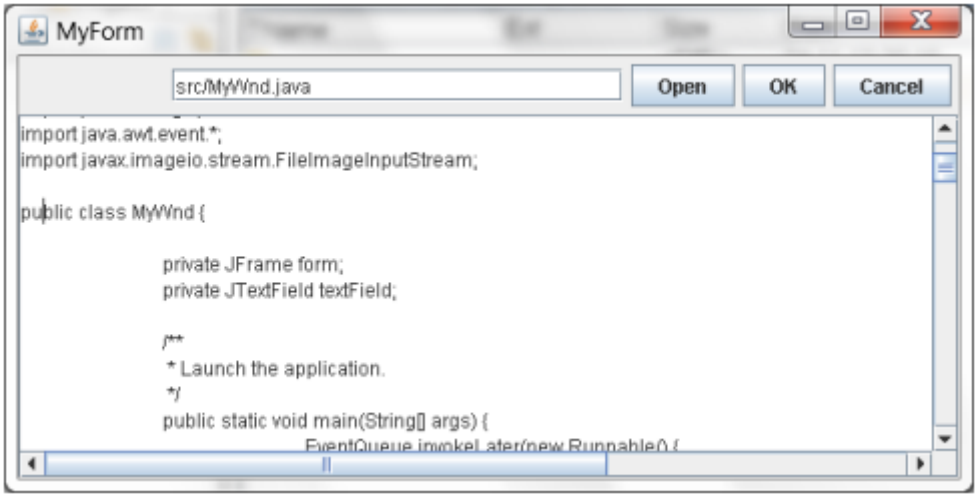


Рис. 6. Вікно додатка редактора

Код програми:

```
import java.io.*;
import java.util.Scanner;
import java.awt.*;
```

```
import javax.swing.*;
import java.awt.event.*;

public class MyApp {      // Клас додатка

        private JFrame form;              // Головне вікно
        private JTextField txtName;       // Поле для імені файла
        private JTextArea txtFile;        // Вміст файла

        public static void main(String[] args) {
                MyApp app = new MyApp();     // Створити об'єкт - додаток
                app.form.setVisible(true);        // Показати його вікно
        }

        public MyApp() {
        // Створюємо вікно
        form = new JFrame(); form.setTitle("MyForm");
        form.setBounds(100, 100, 600, 300);
        form.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Розміщення для елементів головної панелі
        Container pane = form.getContentPane();
        pane.setLayout(new BoxLayout(pane, BoxLayout.Y_AXIS));

        // Створюємо верхню панель
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout(FlowLayout.RIGHT));
        panel.setMaximumSize(new Dimension(600,30));
        pane.add(panel);

        // Додаємо елементи верхньої панелі
        txtName = new JTextField();
        txtName.setColumns(25);
        panel.add(txtName);
        // Перевірка правильності введення
        txtName.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent arg0) {
```

```
            if (!txtName.getText().Matches(".+\\.html?"))
            txtName.setForeground(new Color(255, 100, 100));
            else txtName.setForeground(Color.BLACK);
        }
});
        JButton btnOpen = new JButton("Open");
        panel.add(btnOpen);
        JButton btnOK = new JButton("OK");
        panel.add(btnOK);
        JButton btnCancel = new JButton("Cancel");
        panel.add(btnCancel);

        // Створюємо поле з прокруткою для виведення текста
        txtFile = new JTextArea("No file selected");
        JScrollPane scroll = new JScrollPane(txtFile);
        form.getContentPane().add(scroll);

        // Додаємо обробники:
        // для кнопки Open
        btnOpen.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                // Створюємо діалог вибору файла
                JFileChooser fc = new JFileChooser();
            // Якщо файл обрано
            if (JFileChooser.APPROVE_OPTION ==
                fc.showOpenDialog(form))
            // то його ім'я записуємо в txtName
            txtName.setText( fc.getSelectedFile()
                .getAbsolutePath());
            }
        });
        // для кнопки OK
        btnOK.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                try {
                // Створюємо об'єкт File
                File f = new File(txtName.getText());
```

```
                // Перевіряємо, що файл існує
                if (!f.exists()) txtFile.setText("No such file");
                else {
                        // Відкриваємо файл для читання
                        Scanner sc = new Scanner(f); String s = "";
                        // і записуємо весь його вміст у поле txtFile
                        while (sc.hasNextLine())
                                s = s + sc.nextLine() + "\n"; txtFile.setText(s);
                }
        } catch (Exception e) {
                // Виводимо повідомлення про помилку
                JOptionPane.showMessageDialog(null, e.getMessage());
        }
}
});
// для кнопки Cancel
btnCancel.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
        // Створюємо діалог підтвердження з кнопками OK і CANCEL
        if (JOptionPane.OK_OPTION==
        JOptionPane.showConfirmDialog(form,
                "Exit without saving?", "Exit",
                JOptionPane.OK_CANCEL_OPTION))
        // Якщо натиснуто OK, то завершуємо програму
        System.exit(0);
        }
});
}
}
```

Лабораторна робота №6

Тема: Багатопотокові програми. Малювання графічних примітивів.

Мета: навчитися створювати багатопотокові програми, що використовують засоби синхронізації.

Теоретичні питання

1. Створення потоків за допомогою класу *Thread* і інтерфейсу *Runnable*.
2. Засоби синхронізації потоків у *Java*.
3. Засоби малювання *Swing*, класи *java.awt.Graphics* і *java.awt.Graphics2D*.
4. Подвійна буферизація.

Контрольні питання

1. Чим відрізняються два способи створення потоку: успадкування класу *Thread* і реалізація інтерфейсу *Runnable*?
2. Опишіть основні методи класу *Thread*. Чим відрізняються методи *run*() і *start*()? У чому полягає різниця між методами *wait*(), *sleep*() і *yield*()?
3. Опишіть дію ключового слова *synchronized*.
4. Що таке синхронізовані класи? Чи необхідно синхронізувати код, який використовує тільки синхронізовані класи і чому?
5. Наведіть приклад взаємного блокування потоків.
6. У чому різниця між методами *paint*() і *repaint*() для компонентів *Swing*? Яку роль кожен із них відіграє у малюванні вмісту компонента?
7. Як за допомогою класу *Graphics2D* можна намалювати основні графічні примітиви: лінії, еліпси, прямокутники, багатокутники?
8. Як вивести текстовий напис і як можна задати шрифт цього напису?
9. Як задати колір, товщину ліній і спосіб заливки графічних примітивів?
10. Що таке області відсікання і як вони задаються?
11. Як реалізувати подвійну буферизацію, використовуючи клас *Graphics2D*?

Індивідуальні завдання

Загальне завдання

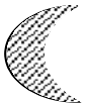
1. Дано набір .csv файлів, кожен із яких містить опис декількох об’єктів (за варіантом). Напишіть програму, яка зчитує вміст цих файлів, створює відповідні об’єкти і зберігає їх усі в один список або масив. Одночасно зі створенням об’єктів програма повинна малювати їх у своєму вікні. Зчитування *кожного* вхідного файла (і створення описаних у ньому об’єктів) виконуйте в окремому потоці. Після зчитування кожного об’єкта виконуйте затримку на 0,5 секунди. Після завершення всіх потоків, які читають файли, виведіть у вікно напис «Читання завершено».

2. (**Для програмістів*) Напишіть програму, яка у своєму вікні відображає кілька рухомих об'єктів (тип об'єктів і закон їх руху – *за варіантом*), а також один об'єкт, керований користувачем за допомогою клавіатури та миші (*реалізуйте обидва варіанти керування*).

   Кожен об'єкт має керуватися окремим потоком. При зіткненні об'єктів між собою або з межами вікна вони повинні міняти напрям руху. Можна вважати, що об'єкти стикаються, коли стикаються прямокутники, що їх обмежують.

   (**Додатково*) Створюйте об'єкти різних (випадкових) розмірів і реалізуйте закон збереження імпульсу при зіткненнях.

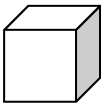
Варіант 1

1. Ламана лінія. Задаються колір, товщина лінії і координати опорних точок.
2. *Вид об'єкта:  Закон руху: синусоїдальна траєкторія з постійною швидкістю.

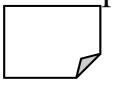
Варіант 2

1. Еліпс. Задаються колір заливки, колір обмежувальної лінії, координати еліпса.
2. *Вид об'єкта:  Закон руху: по прямій зі змінною швидкістю.

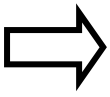
Варіант 3

1. Стрілка. Задаються колір, довжина і ширина стрілки, координати її початку.
2. *Вид об'єкта:  Закон руху: по колу з центрому заданій точці з постійною швидкістю.

Варіант 4

1. Трикутник. Задаються колір заливки, колір обмежувальної лінії і координати вершин.
2. *Вид об'єкта:  Закон руху: по спіралі з постійною швидкістю.

Варіант 5

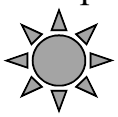
1. Правильний зафарбований шестикутник. Задаються колір і товщина обмежувальної лінії, координати центру і розмір.
2. *Вид об'єкта:  Закон руху: коливальний рух

навколо заданої точки.

Варіант 6

1. П’ятикутна зірка. Задаються колір заливки, колір обмежувальної лінії, координати центру і розмір (радіус).
2. *Вид об’єкта:  Закон руху: рух у полі сили тяжіння(по параболі).

Варіант 7

1. Ромб. Задаються колір і товщина обмежувальної лінії, координати центру і два розміри – ширина і висота.
2. *Вид об’єкта:  Закон руху: «блукаючий» рух (на кожному кроці вектор швидкості змінюється на невелику випадкову величину).

Варіант 8

1. Правильний зафарбований трикутник. Задаються колір і товщина обмежувальної лінії, координати центру і розмір.
2. *Вид об’єкта:  Закон руху: коливальний рух навколо заданої точки.

Варіант 9

1. Квадрат, повернений на довільний кут. Задаються колір заливки, колір обмежувальної лінії, координати центру, розмір і кут повороту.
2. *Вид об’єкта:  Закон руху: рух у полі сили тяжіння (по параболі).

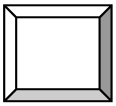
Варіант 10

1. Правильний п’ятикутник. Задаються колір заливки, колір обмежувальної лінії, координати центру і розмір.
2. *Вид об’єкта:  Закон руху: «блукаючий» рух (на кожному кроці вектор швидкості змінюється на невелику випадкову величину).

Варіант 11

1. Еліпс. Задаються колір заливки, колір обмежувальної лінії, координати прямокутника.
2. *Вид об’єкта:  Закон руху: по колу з центром у заданій точці зі змінною швидкістю.

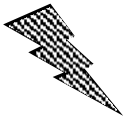
Варіант 12

1. Стрілка. Задаються колір, довжина і ширина стрілки, координати її початку.
2. *Вид об’єкта:  Закон руху: по прямій зі змінною швидкістю.

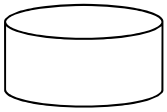
Варіант 13

1. Трикутник.Задаються колір заливки, колір обмежувальної лінії і координати вершин.
2. *Вид об’єкта:  Закон руху: по синусоїдальній траєкторії з постійною швидкістю.

Варіант 14

1. Ламана лінія. Задаються колір, товщина лінії і координати опорних точок.
2. *Вид об’єкта:  Закон руху: по спіралі з постійною швидкістю.

Варіант 15

1. Зафарбований прямокутник. Задаються координати кутів прямокутника, товщина обмежувальної лінії, колір лінії, колір заливки.
2. *Вид об’єкта:  Закон руху: «блукаючий» рух (на кожному кроці вектор швидкості змінюється на невелику випадкову величину).

Додаткові відомості

Приклад малювання заповнених фігур засобами *Java2D*

```
Graphics2D g = (Graphics2D)
getContentPane().GetGraphics(); g.translate(20,20);
// Створюємо область у вигляді трикутника
Area a = new Area(new Polygon(
                    new int[] {0,0,100},
                    new int[] {0,100,50}, 3));
// Додаємо до області ще один трикутник
a.add(new Area(new Polygon(
              new int[] {50,50,150}, new int[] {0,100,50}, 3)));
// Встановлюємо градієнтну заливку
g.setPaint(new GradientPaint(0,50, Color.YELLOW, 150,50, Color.RED));
// Заповнюємо область
```

```
g.fill(a);
// Створюємо еліпс
g.translate(200, 0);
Shape e = new Ellipse2D.Double(0,0,150,100);
// Завантажуємо з файла текстуру для заповнення
try {
        BufferedImage img = ImageIO.read(new
        File("1.png")); g.setPaint(new TexturePaint(img,
        new
Rectangle2D.Double(0,0,16,16)));
} catch (Exception ex) {}
// Заповнюємо еліпс цією текстурою
g.fill(e);
// Встановлюємо параметри обмежувальної лінії – колір і товщину
g.setColor(Color.BLUE);
g.setStroke(new BasicStroke(3));
// Малюємо обмежувальну лінію
g.draw(e);
```

Результат роботи програми:

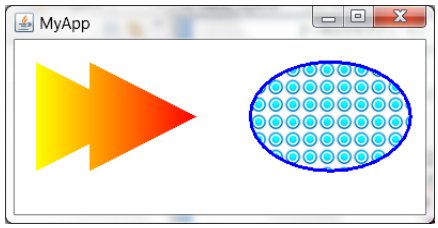


Рис. 7. Результат роботи програми

Приклад роботи з потоками

Створимо додаток із трьома потоками:

- головний потік (у ньому працює весь стандартний інтерфейс користувача);
- потік класу Repainter, що читає рядки з файла з інтервалом в 1 секунду;
- потік класу Reader, що відображає прочитані рядки у вікні з частотою 20 разів за секунду.

```
import javax.swing.*;
```

```
import java.awt.*;
import java.io.File;
import java.util.*;
import java.util.List;

 public class MyApp extends JFrame {                    // Клас додатка і вікна

        // Список відображуваних рядків
        List<String> lst = new ArrayList<String>(100);

        // Потік перемальовування вікна
        static class Repainter extends Thread {
            private JFrame wnd;                         // Вікно, яке
            перемальовувати
            public Repainter(JFrame _wnd) {wnd =
            _wnd;} @Override
            public void run() {                         // Функція потоку
                while (true) {                                  // Цикл
                перемальовування
                    try { sleep(50); } catch (InterruptedException e) {}
                    if (wnd!=null) { wnd.repaint(); }                   //
                    Перемалювати
                }
            }
        }
        // Потік читання з файла
        static class Reader extends Thread {
            private File f = null;                              // Файл
            private List<String> lst;                           // Куди читати
            рядки private boolean bStop = false;                // Прапор
            закінчення public Reader(List<String> l, File _f) {lst = l; f =
            _f; }
            public boolean isRunning() {return !bStop; }
            @Override
            public void run() {                                 // Функція потоку
                Scanner sc = null;
                try { sc = new Scanner(f); } catch (Exception e1) { }
```

```
            while (sc.hasNextLine()) {                  // читання рядка
                synchronized(lst) {
                    lst.add(sc.nextLine(
                    ));
        try { sleep(1000); } catch
(Exception e) { }
            }
            bStop = true;
            synchronized(this) {                    // Повідомити іншим
                потокам, notifyAll();               // що читання закінчено
            }
        }
}

public static void main(String[] args) {
        MyApp wnd = new MyApp();        // Створити вікно
        wnd.setVisible(true);           // і відобразити його
}

public MyApp() {
        super("MyApp");
        setBounds(100, 100, 450,
        300);
        setBackground(Color.WHITE);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Запустити потік відтворення
        Repainter rep = new
        Repainter(this); rep.start();
        // Запустити потік читання з файла
        final Reader r = new Reader(lst, new File("1.txt"));
        r.start();

        // Очікуємо завершення потоку читання
        Thread waitThread = new Thread() { // Для цього створюємо потік
            public void run() {
                while (r.isRunning()) {         // який чекає закінчення r
```

```
                    synchronized(r) {
                        try { r.wait(500); }
                        catch (InterruptedException ie) { }
                    }
                }
                synchronized(lst) {              // і потім додає рядок
                    lst.add("Читання
                      закінчено");
                        }
                        }
        };
        waitThread.start();

}
@Override
public void paint(Graphics gg) {                 // Малювання
        Container p = getContentPane();          // Клієнтська область
        вікна
        // Подвійна буферизація
        Image buf = createImage(p.getWidth(),
        p.getHeight()); Graphics g2 = buf.getGraphics();
        g2.setFont(new Font("Times new roman", Font.BOLD, 18));
        synchronized(lst) {                      // Вивести рядки
            for (int i = 0; i <lst.size(); i ++) { g2.drawString(lst.get(i), 20,
                20*(i+1));
            }
        }
        p.getGraphics().drawImage(buf, 0, 0, null);
}
```

Список літератури

1. Шілдт Г. Java. Полное руководство. 10 – издание. Том 1. / Г. Шілдт. – Діалектика.: Київ, 2020. – 730 с.
2. Шілдт Г. Java. Полное руководство. 10 – издание. Том 2. / Г. Шілдт. – Діалектика.: Київ, 2020. – 780 с.
3. Троян С.О. Програмування мовою Java: навч. посіб. / С. О. Троян. – Умань: ФОП Жовтий О. О., 2017. – 132 с.
4. Кадомський К.К., Ніколюк П.К. Java. Теорія і практика: навч. посіб./ Кадомський К.К., Ніколюк П.К. – Вінниця: Донну, 2019. – 197 с.
5. Тарнавський Ю.А., Java-програмування: комп'ютерний практикум [Електронний ресурс] : навч. посіб. / КПІ ім. Ігоря Сікорського ; уклад.: Ю. А. Тарнавський. – Київ : КПІ ім. Ігоря Сікорського, 2021. – 95 с.
6. Бібліотека КНУБА. URL: http://library.knuba.edu.ua/
7. Освітній сайт КНУБА. URL: http://org.knuba.edu.ua/

Навчально-методичне видання

ІНТЕРНЕТ-ТЕХНОЛОГІЇ ТА МОВА ПРОГРАМУВАННЯ JAVA

Методичні вказівки
до виконання лабораторних робіт 1-6
для підготовки здобувачів освітньо-кваліфікаційного рівня «бакалавр»
спеціальностей 122 «Комп'ютерні науки»

Автор: Олександр ПОПЛАВСЬКИЙ

Комп'ютерне верстання

Підписано до друку 22.02.2018 Формат 60 × 84 1/ 16
Ум. друк. арк. 1,16. Обл.-вид. арк. 1,25.
Електронний документ. Вид № 59/ІІІ-17.

Видавець і виготовлювач
Київський національний університет будівництва і архітектури

Повітрофлотський проспект, 31, Київ, Україна, 03037
Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи ДК № 808 від 13.02.2002 р.