**Лабораторна робота №1**

**Типи даних, умови, цикли та масиви**

**Мета:** Використовуючи теоретичне підґрунтя про типи даних, що використовуються у мові Java, правила використання умовних переходів, написання циклів та використання масивів виконати дії що будуть вказано в завданні до лабораторної роботи.

**Хід роботи**

**Посилання на GitHub:** [OOP-KB-222-Ann-Dryzhak/lab\_01 at main · aanutadrzhk/OOP-KB-222-Ann-Dryzhak (github.com)](https://github.com/aanutadrzhk/OOP-KB-222-Ann-Dryzhak/tree/main/lab_01)

**Завдання 1.** Напишіть програму, яка прочитає рядок із System.in і виведе повідомлення: "Hello, прочитаний рядок". Якщо у вхідному рядку кілька слів, потрібно надрукувати їх усі.

package lab\_01;

import java.util.Scanner;

public class task\_01 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.println("Введіть рядок:");

String input = console.nextLine();

System.***out***.println("Hello, " + input);

console.close();

}

}

Пояснення:

* Імпортуємо клас Scanner з пакету java.util, щоб отримати можливість зчитувати введення користувача
* Оголошуємо клас task\_01
* Оголошуємо метод main, який є точкою входу в програму
* Створюємо об'єкт класу Scanner, який приймає System.in як вхідний потік даних
* Виводимо текст "Введіть рядок:"
* Читаємо рядок введення користувача і зберігаємо його в змінній input
* Виводимо "Hello, " разом зі значенням змінної input
* Закриття об'єкту console для вивільнення ресурсів

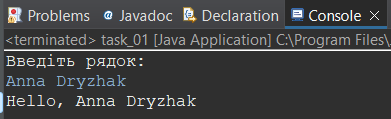


Рис. 1 - результат виконання завдання 1

**Завдання 2.** Напишіть програму, яка отримує на вхід ціле число. Це число є поточний добовий час, який обчислюється як кількість секунд, що пройшли з півночі. Потім програма повинна надрукувати значення екрану електронного годинника для цього часу.

package lab\_01;

import java.util.Scanner;

public class task\_02 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.println("Введіть кількість секунд з півночі:");

int totalSeconds = console.nextInt();

console.close();

int hours = totalSeconds / 3600;

int remainingSecondsAfterHours = totalSeconds % 3600;

int minutes = remainingSecondsAfterHours / 60;

int seconds = remainingSecondsAfterHours % 60;

System.***out***.printf("%d:%02d:%02d\n", hours, minutes, seconds);

}

}

Пояснення:

* Створюємо об'єкту console для зчитування введення користувача з консолі (System.in)
* Виводимо тексту "Введіть кількість секунд з півночі:" на консоль
* Далі йде зчитування цілого числа з консолі і зберігання його у змінній totalSeconds
* Закриття об'єкту console для вивільнення ресурсів
* int hours = totalSeconds / 3600 - розраховує кількість годин, які пройшли, розділивши загальну кількість секунд totalSeconds на 3600
* int remainingSecondsAfterHours = totalSeconds % 3600 - після розрахунку кількості годин, залишок секунд, які залишилися після врахування годин, обчислюється за допомогою оператора залишку від ділення %. Тобто, цей рядок дозволяє знайти кількість секунд, які не ввійшли в повні години
* int minutes = remainingSecondsAfterHours / 60 - обчислює кількість хвилин, розділивши remainingSecondsAfterHours на 60
* int seconds = remainingSecondsAfterHours % 60; - залишок секунд після розрахунку хвилин обчислюється за допомогою оператора %
* Виводимо результат у форматі "h:mm:ss", де %d - це цілі числа, а %02d - цілі числа, які заповнюються нулями зліва до двох символів

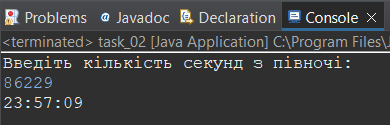


Рис. 2 - результат виконання завдання 2

**Завдання 3.** Напишіть програму, яка запитує на вхід число-пароль. Якщо воно дорівнює секретній кількості, програма виводить: "Hello, Agent", інакше друкує: "Access denied". Секретний пароль зберігається в final static int password.

package lab\_01;

import java.util.Scanner;

public class Task\_03 {

private static final int ***password*** = 123456789;

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.println("Введіть пароль:");

int inputPassword = console.nextInt();

console.close();

if (inputPassword == ***password***) {

System.***out***.println("Hello, Agent");

} else {

System.***out***.println("Access denied");

}

}

}

Пояснення:

* private static final int password = 123456 - оголошуємо константу password типу int зі значенням 123456. Ключове слово private вказує на те, що ця константа доступна тільки в межах класу Main. Static вказує на те, що ця константа є статичною і належить класу, а не конкретному об'єкту. Final вказує на те, що значення цієї змінної не може бути змінено після ініціалізації.
* if (enteredPassword == password) { - перевіряємо, чи введений пароль enteredPassword співпадає з секретним паролем.
* System.out.println("Hello, Agent"); - виводимо "Hello, Agent" на консоль, якщо введений пароль співпадає зі секретним паролем.
* System.out.println("Access denied"); - виводимо "Access denied" на консоль, якщо введений пароль не співпадає зі секретним паролем.

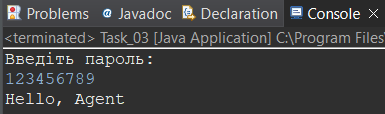


Рис. 3 - результат виконання завдання 3

**Завдання 4.** Напишіть просту програму, яка запитує кількість незнайомців, з якими потрібно зустрітися, рядково читає імена незнайомців, по рядках виводить: "Hello, (ім'я незнайомця)" для кожного незнайомця. Доповнити програму перевірками на кількість, що є від’ємною, та рівна 0.

package lab\_01;

import java.util.Scanner;

public class Task\_04 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

int numberOfStrangers;

do {

System.***out***.println("Введіть кількість незнайомців (позитивне ціле число):");

numberOfStrangers = scanner.nextInt();

} while (numberOfStrangers <= 0);

scanner.nextLine();

for (int i = 1; i <= numberOfStrangers; i++) {

System.***out***.println("Введіть ім'я незнайомця №" + (i) + ":");

String name = scanner.nextLine();

System.***out***.println("Hello, " + name);

}

scanner.close();

}

}

Пояснення:

* int numberOfStrangers; - оголошуємо змінну numberOfStrangers, яка буде використовуватися для збереження кількості незнайомців.
* do ... while (numberOfStrangers <= 0); - цикл do-while, який виконується, поки numberOfStrangers менше або дорівнює 0. Використовуючи цей цикл, програма запитує користувача ввести кількість незнайомців, і вона продовжує запитувати, поки введене число не стане додатнім.
* scanner.nextLine() - читаємо рядок, який введено після кількості незнайомців, але не використовує його. Це для того, щоб сканер перейшов до наступного рядка вводу.
* for (int i = 1; i <= numberOfStrangers; i++) ... - цикл for, який повторюється стільки разів, скільки незнайомців ввів користувач. Кожний раз, коли цикл повторюється, змінна i збільшується на 1.
* System.out.println("Hello, " + name) - виводимо "Hello, ..." разом з ім'ям незнайомця, яке ввів користувач.

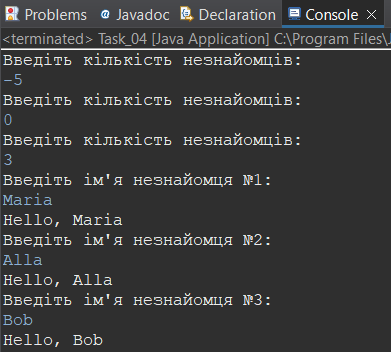


Рис. 4 - результат виконання завдання 4

**Завдання 5.** Наведено таку умову: равлик піднімається по дереву на a футів на день. Потім щоночі равлик сповзає вниз на b футів. Висота дерева — h футів. Напишіть програму з підрахунком кількості днів, які знадобляться равлику, щоб дістатися до вершини дерева. Програма читає a, b, h рядково. Вхідні значення гарантовано є додатніми цілими числами. Якщо равлик не може дістатися до вершини дерева, має виводитися повідомлення: Impossible.

package lab\_01;

import java.util.Scanner;

public class Task\_05 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.println("Введіть значення а:");

int a = console.nextInt();

System.***out***.println("Введіть значення b:");

int b = console.nextInt();

System.***out***.println("Введіть значення h:");

int h = console.nextInt();

int days = 0;

if (a >= h) {

days = 1;

} else if (a <= b) {

days = -1;

} else {

days = (int) Math.*ceil*((double)(h - a) / (a - b)) + 1;

}

if (days == -1) {

System.***out***.println("Impossible");

} else {

System.***out***.println("Результат: " + days);

}

console.close();

}

}

Пояснення:

* Зчитуємо значення a, b та h.
* Оголошує змінну days і ініціалізує її значенням 0.
* if (a >= h) ... - перевіряємо, чи равлик може дістатися до вершини дерева за один день . Якщо так, встановлює days рівним 1.
* else if (a <= b) ... - перевіряємо, чи равлик не може дістатися до вершини дерева. Якщо так, встановлює days рівним -1.
* else ... - ця частина виконується, якщо обидва попередні умови не виконуються.
* (double)(h - a) / (a - b) - перетворюємо результат виразу (h - a) / (a - b) у дробове число (конвертуємо вираз у тип double). Math.ceil((double)(h - a) / (a - b)) - викликаємо метод Math.ceil, який вищеокруглює дробове число. Він округлює результат до найближчого цілого числа в більшу сторону. (int) Math.ceil((double)(h - a) / (a - b)) - перетворюємо результат вищеокругленого дробового числа у тип int. Це робиться за допомогою (int), який конвертує вираз у тип int.
* if (days == -1) ... - перевіряємо, чи days рівний -1. Якщо так, програма виводить "Impossible".
* else ... - ця частина виконується, якщо days не рівний -1. В цьому випадку програма виводить значення days.

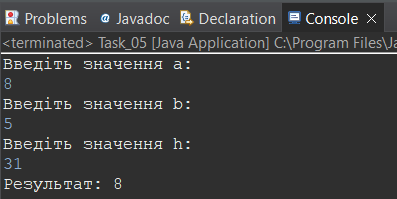


Рис. 5 - результат виконання завдання 5

**Завдання 6.** Дано така умова: компанія друзів відвідує ресторан. Вони вирішили розділити рахунок порівну та додати 10 відсотків від загальної суми рахунку як чайові. Далі друзі покривають рівними частинами загальну суму платежу. Напишіть програму, яка зчитує загальну суму рахунку та кількість друзів, а потім виводить розмір частини оплати. Доповнити програму перевірками на від’ємну суму рахунку та нульову кількість друзів.

package lab\_01;

import java.util.Scanner;

public class Task\_06 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.println("Введіть загальний рахунок: ");

int totalBill = console.nextInt();

System.***out***.println("Введіть кількість друзів: ");

int numberOfFriends = console.nextInt();

if (totalBill <= 0 || numberOfFriends <= 0) {

System.***out***.println("Invalid input");

return;

}

double eachPayment = (totalBill + 0.1 \* totalBill)/numberOfFriends;

System.***out***.printf("Оплата рахунку на 1 людину: %.2f", eachPayment);

}

}

Пояснення:

* Зчитуємо цілі числа суми рахунку та кількості друзів з клавіатури і зберігаємо в відповідних змінних.
* if (totalBill <= 0 || numberOfFriends <= 0) - перевіряємо, чи введені дані відповідають задачі, що обидва числа додатні. Якщо хоча б одне з чисел менше або дорівнює нулю, програма виводить "Invalid input" та завершується.
* double eachPayment = (totalBill + 0.1 \* totalBill) / numberOfFriends; - обчислюємо розмір кожної частини оплати. Спочатку до загальної суми рахунку додаємо 10% від цієї суми, щоб врахувати чайові. Потім результат ділимо на кількість друзів.
* System.out.printf("%.2f", eachPayment); - виводимо розмір кожної частини оплати з двома знаками після коми. %.2f означає, що вивід буде з двома знаками після коми (копійки).

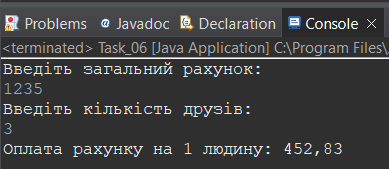


Рис. 6 - результат виконання завдання 6

**Завдання 7.** Напишіть програму, яка зчитує послідовність цілих значень зі стандартного введення і знаходить максимальне з них. Ви повинні зчитувати значення послідовності, доки наступне не стане 0. Нульове значення означає кінець вхідної послідовності та не є її елементом. Послідовність гарантовано містить хоча б одне значення.

package lab\_01;

import java.util.Scanner;

public class Task\_07 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

int maxValue = -1;

int input;

System.***out***.println("Введіть цілі числа (останнє 0): ");

while ((input = console.nextInt()) != 0) {

if (input > maxValue) {

maxValue = input;

}

}

System.***out***.println("Найбільше число: " +maxValue);

}

}

Пояснення:

* Оголошуємо змінну maxValue, яка буде використовуватися для збереження максимального значення та становлюємо початкове значення -1.
* Оголошуємо змінну input, яка буде зберігати введене число.
* while ((input = scanner.nextInt()) != 0) - починаємо цикл while. Умова циклу перевіряє, чи наступне зчитане число, що зберігається у змінній input, не дорівнює 0. Якщо воно дорівнює 0, цикл завершується.
* В середині циклу перевіряємо, чи поточне зчитане число (input) більше за поточне максимальне значення (maxValue). Якщо так, то змінюємо значення maxValue на input.
* Після завершення циклу виводимо максимальне значення, яке було знайдено.

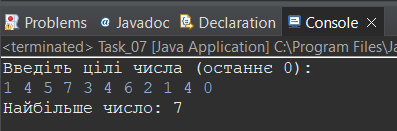


Рис. 7 - результат виконання завдання 7

**Завдання 8.** Напишіть програму, яка зчитує послідовність цілих значень зі стандартного введення та знаходить середнє значення. Ви повинні зчитувати значення послідовності, доки наступне не стане 0. Нульове значення означає кінець вхідної послідовності та не є її частиною. Послідовність гарантовано містить хоча б одне значення.

package lab\_01;

import java.util.Scanner;

public class Task\_08 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

int sum = 0;

int count = 0;

int input;

System.***out***.println("Введіть цілі числа (останнє 0): ");

while ((input = console.nextInt()) != 0) {

sum += input;

count++;

}

double average = (double) sum / count;

System.***out***.println("Середнє значення: " +average);

}

}

Пояснення:

* Оголошуємо змінну sum, яка буде використовуватися для збереження суми всіх чисел у послідовності. Встановлюємо початкове значення 0.
* Оголошуємо змінну count, яка буде використовуватися для збереження кількості чисел у послідовності. Встановлюємо початкове значення 0.
* Оголошуємо змінну input, яка буде використовуватися для збереження введеного числа.
* while ((input = scanner.nextInt()) != 0) - починаємо цикл while. Умова циклу перевіряє, чи наступне зчитане число не дорівнює 0. Якщо воно дорівнює 0, цикл завершується.
* В середині циклу додаємо зчитане число до змінної sum, щоб підрахувати суму всіх чисел. Та збільшуємо змінну count на 1, щоб підрахувати кількість чисел.
* double average = (double) sum / count; - обчислюємо середнє значення, ділячи суму всіх чисел на кількість чисел у послідовності. Один із операндів робимо у double, щоб виконати ділення як дійсне число.

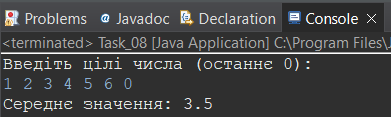


Рис. 8 - результат виконання завдання 8

**Завдання 9.** Напишіть програму для вирішення квадратних рівнянь. При виконанні врахуйте залежність значення дискримінанту до кількості результатів.

package lab\_01;

import java.util.Scanner;

public class Task\_09 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.println("Введіть коефіцієнти a, b, c для квадратного р і вняння: ");

double a = console.nextDouble();

double b = console.nextDouble();

double c = console.nextDouble();

double D = b \* b - 4 \* a \* c;

if (D > 0) {

double x1 = (-b + Math.*sqrt*(D)) / (2 \* a);

double x2 = (-b - Math.*sqrt*(D)) / (2 \* a);

System.***out***.println("Результат: " + x1 + " " + x2);

} else if (D == 0) {

double x = -b / (2 \* a);

System.***out***.println("Результат: " + x);

} else {

System.***out***.println("Результат: no roots");

}

}

}

Пояснення:

* Зчитуємо значення коефіцієнтів з консолі.
* Обчислюємо дискримінант за формулою D = b^2 - 4ac.
* Починаємо перевірку умови, чи дискримінант більше нуля. Якщо так, то рівняння має два різних розв'язки. Та обчислюємо розв’язки за відповідними формулами й виводим результат.
* Якщо дискримінант дорівнює нулю, то рівняння має один розв'язок. Обчислюємо розв'язок за формулою x = -b / (2a). Та виводимо результат.
* Якщо ж дискримінант менше нуля, то рівняння не має розв'язків. Відповідно виводимо повідомлення про відсутність розв'язків.

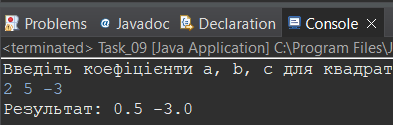


Рис. 9 - результат виконання завдання 9

**Завдання 10.** Напишіть програму, що містить метод max(…). Правильна реалізація має отримати масив значень типу int та повернути його максимальне значення. Вхідний масив гарантовано не буде порожнім або null. Метод max не повинен змінювати масив.

package lab\_01;

import java.util.Scanner;

public class Task\_10 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.print("Введіть кількість елементів масиву: ");

int size = console.nextInt();

while (size <= 0) {

System.***out***.println("Кількість елементів масиву повинна бути більше нуля.");

System.***out***.print("Введіть кількість елементів масиву: ");

size = console.nextInt();

}

int[] array = new int[size];

System.***out***.println("Введіть елементи масиву:");

for (int i = 0; i < size; i++) {

array[i] = console.nextInt();

}

int maxValue = *max*(array);

System.***out***.println("Максимальне значення: " + maxValue);

}

public static int max(int[] array) {

int max = array[0];

for (int i = 1; i < array.length; i++) {

if (array[i] > max) {

max = array[i];

}

}

return max;

}

}

Пояснення:

* Зчитуємо ціле число кількості елементів масиву з клавіатури і зберігаємо в змінній size.
* Далі розпочинаємо цикл while, який виконується, поки значення size менше або рівне 0.
* int[] array = new int[size]; - створюємо новий масив цілих чисел з розміром, що вказав користувач.
* for (int i = 0; i < size; i++) - починаємо цикл for, який виконується size разів. Кожен раз зчитуємо ціле число з клавіатури і зберігаємо його в відповідному (i) елементі масиву.
* Далі викликаємо метод max (пояснення нижче) з аргументом array і зберігаємо результат в змінній maxValue. Та виводимо текст максимальне значення на консоль.
* Оголошуємо метод max, який приймає масив цілих чисел як аргумент.
* int max = array[0]; - встановлюємо змінну max рівною першому елементу масиву.
* for (int i = 1; i < array.length; i++) - починаємо цикл for, який виконується для кожного елемента масиву, починаючи з другого (array.length разів).
* if (array[i] > max) - перевіряємо, чи поточний елемент масиву більшим за змінну max.
* max = array[i]; - значення max оновлюється , якщо поточний елемент масиву більший за поточне значення max.

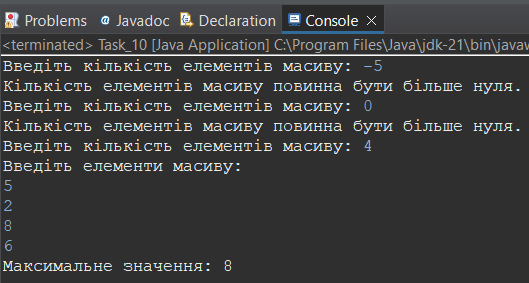


Рис. 10 - результат виконання завдання 10

**Завдання 11.** Напишіть програму, що містить метод sum. Правильна реалізація має отримати масив значень int та повернути суму парних чисел. Якщо цей масив дорівнює нулю або порожній, метод повертає 0. Метод sum не повинен змінювати масив.

package lab\_01;

import java.util.Scanner;

public class Task\_11 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.print("Введіть кількість елементів масиву: ");

int size = console.nextInt();

while (size <= 0) {

System.***out***.println("Кількість елементів масиву повинна бути більше нуля.");

System.***out***.print("Введіть кількість елементів масиву: ");

size = console.nextInt();

}

int[] array = new int[size];

System.***out***.println("Введіть елементи масиву:");

for (int i = 0; i < size; i++) {

array[i] = console.nextInt();

}

int result = *sum*(array);

System.***out***.println("Максимальне значення: " + result);

}

public static int sum(int[] array) {

int sum = 0;

for (int i = 0; i < array.length; i++) {

if (array[i] % 2 == 0) {

sum += array[i];

}

}

return sum;

}

}

Пояснення:

* Головний метод main реалізован так само як в завданні 10.
* Оголошуємо метод sum, який приймає масив array та повертає суму парних чисел.
* Початкове значення суми встановлюємо 0.
* for (int i = 0; i < array.length; i++) - розпочинаємо цикл, який повторюється array.length разів.
* if (array[i] % 2 == 0) - перевіряємо, чи поточний елемент масиву парний (якщо остача від ділення дорівнює 0, то елемент парний).
* sum += array[i]; - додаємо поточний елемент до суми, якщо цей елемент пройшов умову парності.

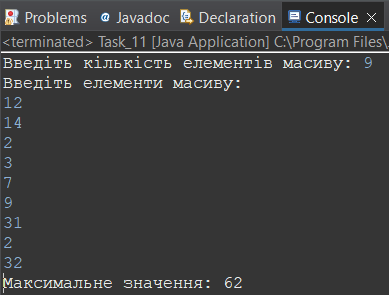


Рис. 11 - результат виконання завдання 11

**Завдання 12.** Напишіть програму, що містить метод getSumCheckArray. Правильна реалізація повинна отримати масив значень int і повернути масив логічних значень, де кожен елемент є результатом перевірки: чи є вихідний елемент сумою двох попередніх елементів у даному масиві. Гарантується, що довжина масиву становить 2 або більше. Цей масив гарантовано не дорівнює null. Метод повертає масив логічних значень, де кожен елемент є результатом для відповідного елемента в даному масиві. Перші два елементи логічного масиву завжди false.

package lab\_01;

import java.util.Scanner;

public class Task\_12 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.print("Введіть кількість елементів масиву: ");

int size = console.nextInt();

while (size <= 1) {

System.***out***.println("Кількість елементів масиву повинна бути більше нуля.");

System.***out***.print("Введіть кількість елементів масиву: ");

size = console.nextInt();

}

int[] array = new int[size];

System.***out***.println("Введіть елементи масиву:");

for (int i = 0; i < size; i++) {

array[i] = console.nextInt();

}

boolean[] result = *getSumCheckArray*(array);

System.***out***.print("Результат: [");

for (int i = 0; i < result.length; i++) {

System.***out***.print(result[i]);

if (i < result.length - 1) {

System.***out***.print(", ");

}

}

System.***out***.println("]");

}

public static boolean[] getSumCheckArray(int[] array) {

boolean[] result = new boolean[array.length];

if (array.length >= 3) {

for (int i = 2; i < array.length; i++) {

result[i] = array[i] == array[i - 1] + array[i - 2];

}

}

return result;

}

}

Пояснення:

* Головний метод main реалізован так само як в завданні 10 та 11. Але додано частину, що виводить результат, а саме:
  + Виводимо текст "Результат: [" без переведення рядка.
  + for (int i = 0; i < result.length; i++) - запускаємо цикл, який перебирає всі елементи масиву result.
  + System.out.print(result[i]); - Виводимо значення result[i] без переведення рядка.
  + if (i < result.length - 1) - перевіряємо, чи ітерація циклу не остання. Якщо не остання, то виводимо текст ", " без переведення рядка. Та в кінці виводимо символ "]".
* Оголошуємо метод getSumCheckArray, який приймає масив цілих чисел array і повертає масив булевих значень.
* boolean[] result = new boolean[array.length]; - створюємо масив result такої ж довжини як і array, для зберігання результатів перевірки.
* if (array.length >= 3) - перевіряємо, чи масив array має довжину не менше трьох елементів.
* for (int i = 2; i < array.length; i++) - запускаємо цикл, який перебирає всі елементи масиву array починаючи з третього.
* result[i] = array[i] == array[i - 1] + array[i - 2]; - перевіряємо, чи сума попередніх двох елементів дорівнює поточному елементу. Результат записується у відповідний елемент масиву result.

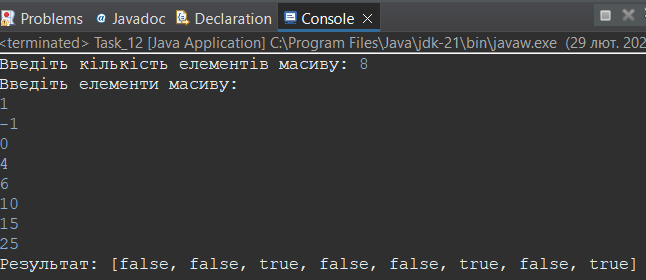


Рис. 12 - результат виконання завдання 12

**Завдання 13.** Напишіть програму, що містить метод removeLocalMaxima(…). Правильна реалізація має отримати масив значень типу int і повернути копію даного масиву з усіма видаленими локальними максимумами. Вихідний масив не можна змінювати. Розмір цього масиву гарантовано більше 1. Якщо масив не має локальних максимумів, ви повинні повернути його копію без змін. Ви можете використовувати методи java.util.Arrays.

package lab\_01;

import java.util.Scanner;

import java.util.Arrays;

public class Task\_13 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.print("Введіть кількість елементів масиву: ");

int size = console.nextInt();

while (size <= 0) {

System.***out***.println("Кількість елементів масиву повинна бути більше нуля.");

System.***out***.print("Введіть кількість елементів масиву: ");

size = console.nextInt();

}

int[] Array = new int[size];

System.***out***.println("Введіть елементи масиву:");

for (int i = 0; i < size; i++) {

Array[i] = console.nextInt();

}

int[] result = *removeLocalMaxima*(Array);

System.***out***.println("Результат: " + Arrays.*toString*(result));

}

public static int[] removeLocalMaxima(int[] Array) {

int[] resultArray = Arrays.*copyOf*(Array, Array.length);

for (int i = 1; i < Array.length - 1; i++) {

if (Array[i] > Array[i - 1] && Array[i] > Array[i + 1]) {

resultArray = *removeElement*(resultArray, i);

}

}

if (Array.length > 1 && Array[0] > Array[1]) {

resultArray = *removeElement*(resultArray, 0);

}

if (tArray.length > 1 && Array[Array.length - 1] > Array[Array.length - 2]) {

resultArray = *removeElement*(resultArray, resultArray.length - 1);

}

return resultArray;

}

public static int[] removeElement(int[] Array, int index) {

int[] newArray = new int[Array.length - 1];

System.*arraycopy*(Array, 0, newArray, 0, index);

System.*arraycopy*(Array, index + 1, newArray, index, Array.length - i ndex - 1);

return newArray;

}

}

Пояснення:

* Головний метод main реалізован так само як в завданні 10 та 11.
* int[] resultArray = Arrays.copyOf(Array, Array.length); - створюємо копію вхідного масиву Array за допомогою методу Arrays.copyOf.
* for (int i = 1; i < inputArray.length - 1; i++) - перебираємо всі елементи масиву, починаючи з другого і закінчуючи передостаннім.
* if (inputArray[i] > inputArray[i - 1] && inputArray[i] > inputArray[i + 1]) - перевіряємо, чи є поточний елемент більшим за його попередній і наступний елементи. Якщо ця умова виконується, то елемент є локальним максимумом і його потрібно видалити.
* resultArray = removeElement(resultArray, i); - видаляємо локальні максимуми з масиву resultArray за допомогою методу removeElement.
* Перевіряємо перший та останній елементи масиву, чи вони є локальними максимумами. Якщо так, то вони теж видаляються з масиву.
* Повертаємо копію вихідного масиву resultArray.
* Створюємо метод removeElement, який отримує два параметри, а саме: масив array, з якого потрібно видалити елемент, та індекс, який вказує на елемент, що має бути видалений.
* int[] newArray = new int[array.length - 1]; -створюємо новий масив newArray з довжиною, яка на одиницю менше за довжину вихідного масиву, оскільки ми видаляємо один елемент з масиву.
* System.arraycopy(array, 0, newArray, 0, index); - копіюємо всі елементи з початку вихідного масиву array до індексу index (не включаючи його) в новий масив newArray. Тобто всі елементи до елемента, який ми хочемо видалити, переносимо з змін.
* System.arraycopy(array, index + 1, newArray, index, array.length - index - 1); - копіюємо всі елементи з вихідного масиву array, починаючи з наступного після елемента, який ми хочемо видалити (з індексу index + 1), до кінця масиву в новий масив newArray. Копіюються елементи після елемента, який ми видаляємо, і до кінця масиву.
* Після видалення елемента повертаємо новий масив newArray, який містить всі елементи вихідного масиву, крім видаленого елемента.

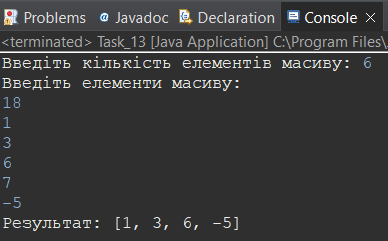


Рис. 13 - результат виконання завдання 13

**Завдання 14.** Напишіть програму що містить два методи, що реалізують наступний функціонал:

void cycleSwap(int[] array) - Зсуває всі елементи в даному масиві вправо на 1 позицію. В цьому випадку останній елемент масиву стає першим. Наприклад, 1 3 2 7 4 стає 4 1 3 2 7.

void cycleSwap(int[] array, int shift) - Зсуває всі елементи в заданому масиві вправо на shift позицій. Гарантується, що значення зсуву невід'ємне і не більше за довжину масиву. Наприклад, 1 3 2 7 4 зі зсувом 3 стає 2 7 4 1 3.

package lab\_01;

import java.util.Scanner;

public class Task\_14 {

public static void main(String[] args) {

Scanner console = new Scanner(System.***in***);

System.***out***.print("Введіть розмір масиву: ");

int size = console.nextInt();

while (size <= 0) {

System.***out***.println("Кількість елементів масиву повинна бути більше нуля.");

System.***out***.print("Введіть кількість елементів масиву: ");

size = console.nextInt();

}

int[] array = new int[size];

System.***out***.println("Введіть елементи масиву:");

for (int i = 0; i < size; i++) {

array[i] = console.nextInt();

}

System.***out***.print("Введіть зсув: ");

int shift = console.nextInt();

while (shift <= 0 || shift > array.length) {

System.***out***.println("Невірне значення зсуву! Введіть ще раз: ");

shift = console.nextInt();

}

*cycleSwap*(array);

System.***out***.println("Результат зсуву на 1 елемент:");

for (int i = 0; i < array.length; i++) {

System.***out***.print(array[i] + " ");

}

*cycleSwap*(array, shift);

System.***out***.println("\nРезультат зсуву на " + shift + " елементи:");

for (int i = 0; i < array.length; i++) {

System.***out***.print(array[i] + " ");

}

}

public static void cycleSwap(int[] array) {

int firstElement = array[0];

for (int i = array.length - 1; i > 0; i--) {

array[i] = array[i - 1];

}

array[array.length - 1] = firstElement;

}

public static void cycleSwap(int[] array, int shift) {

for (int j = 0; j < shift-1; j++) {

*cycleSwap*(array);

}

}

}

Пояснення:

* Зчитуємо введений користувачем розмір масиву та перевіряємо чи не менший він 1.
* int[] array = new int[size]; - створюємо новий масив цілих чисел із заданим розміром.
* Виводимо текст у консоль, щоб попросити користувача ввести елементи масиву.
* for (int i = 0; i < size; i++) { array[i] = console.nextInt(); } - зчитуємо введені користувачем числа та записуємо їх у масив.
* Виводимо та зчитуємо число зсуву елементів масиву (змінна shift). Та робимо перевірку чи це значення не менше 1 та не більше велечини масиву.
* cycleSwap(array, shift); - викликаємо метод cycleSwap, який зсуває елементи масиву на shift позицій.
* cycleSwap(array); - викликаємо метод cycleSwap, який зсуває елементи масиву на одну позицію.
* Виводить текст у консоль, щоб показати результат першого зсуву. for (int i = 0; i < array.length; i++) { System.out.print(array[i] + " "); } - цикл, який виводить елементи масиву у консоль через пробіл.
* cycleSwap(array, shift); - викликаємо метод cycleSwap, який зсуває елементи масиву на shift позицій.
* System.out.println("\nРезультат зсуву на " + shift + " елементи:"); - виводимо результат другого зсуву у консоль (\n - перехід на новий рядок).
* for (int i = 0; i < array.length; i++) { System.out.print(array[i] + " "); } - цикл, який виводить елементи масиву у консоль через пробіл.
* public static void cycleSwap(int[] array) - починаємо метод cycleSwap, який зсуває всі елементи масиву на одну позицію.
* int firstElement = array[0]; - зберігаємо перший елемент масиву в змінній firstElement.
* for (int i = array.length - 1; i > 0; i--) { array[i] = array[i - 1]; } - цикл, який зсуває всі елементи масиву на одну позицію. А саме йде заміна значення елемента array[i] на значення попереднього елемента array[i - 1].
* array[array.length - 1] = firstElement; - встановлюємо останній елемент масиву рівним firstElement.
* public static void cycleSwap(int[] array, int shift) - починаємо метод cycleSwap, який зсуває всі елементи масиву на shift позицій.
* for (int j = 0; j < shift - 1; j++) { cycleSwap(array); } - цикл, який викликає метод cycleSwap (shift – 1, -1 оскільки на 1 позицію елементи вже здвинуті) разів, зсуваючи елементи масиву на одну позицію вліво кожного разу.

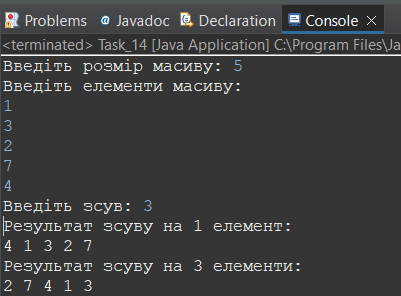


Рис. 14 - результат виконання завдання 14

**Висновок.** Під час виконання даної лабораторної роботи я використовуючи теоретичне підґрунтя про типи даних, що використовуються у мові Java, правила використання умовних переходів, написання циклів та використання масивів виконала дії що було вказано в завданні до лабораторної роботи.