**Лабораторна робота №1**

**Тема:** 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:**

**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.

Спочатку я використовую конструкцію import java.util.Scanner;, щоб імпортувати клас Scanner, який дозволить мені отримувати введені данні від користувача. Потім я створюю клас Task01, де оголошую метод main. У методі main я створюю новий об'єкт Scanner з назвою InputScanner, який приймає введення з системного вводу (System.in). Потім я виводжу повідомлення "Вкажіть ваше ім'я:" за допомогою System.out.println(). Після цього я отримую введення користувача у вигляді рядка за допомогою методу nextLine() об'єкту InputScanner та зберігаю його у змінній text. Після того, як я отримав текст, я закриваю об'єкт InputScanner, використовуючи метод close(), щоб уникнути зайвого використання ресурсів і нарешті виводжу привітання, складене з рядка "Hello, " та введеного користувачем тексту із змінної text, за допомогою System.out.println().

|  |
| --- |
| import java.util.Scanner;    public class Task01 {  public static void main(String[] args) {  Scanner InputScanner = new Scanner(System.in);    System.out.println("Вкажіть ваше ім'я:");  String text = InputScanner.nextLine();    InputScanner.close();    System.out.println("Hello, " + text);  }  } |



Рисунок 1.1 - Демонстрація роботи коду

**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. Вхідне значення отримується через System.in. Вихідне значення має бути надруковане у System.out. Гарантується, що вхідне значення — невід'ємне. Формат виводу: hh:mm:ss (можливі значення: [0:00:00; 23:59:59]).

Спочатку я використовую конструкцію import java.util.Scanner;, щоб імпортувати клас Scanner, який дозволить мені отримувати данні від користувача. Потім я оголошую клас Task02 і його метод main. У методі main я створюю новий об'єкт Scanner з назвою InputScanner, що дозволяє зчитувати введення користувача з консолі. Потім я виводжу на екран повідомлення "Вкажіть кількість секунд:" за допомогою System.out.println(). Далі я зчитую введене користувачем значення в цілочисельну змінну InputValue за допомогою методу nextInt() об'єкту InputScanner. Після цього я перевіряю, чи введене число від'ємне, використовуючи цикл while. Якщо воно від'ємне, я виводжу повідомлення "Було вказано від'ємну кількість секунд, вкажіть нове значення:" і знову зчитую нове значення з консолі. Цей процес повторюється, доки користувач не введе не від'ємне число. Після закриття об'єкта InputScanner я обчислюю кількість годин, хвилин і секунд за допомогою арифметичних операцій. В кінці я вивожу ці значення у форматі години:хвилини:секунди, використовуючи метод printf() з форматуванням %d:%02d:%02d\n, де %d позначає ціле число, а %02d означає ціле число, яке відображається у двоцифровому форматі, де якщо число має одну цифру, то перед нею додається 0.

|  |
| --- |
| import java.util.Scanner;    public class Task02 {  public static void main(String[] args) {  Scanner InputScanner = new Scanner(System.in);  System.out.println("Вкажіть кількість секунд:");  int InputValue = InputScanner.nextInt();    while (InputValue < 0) {  System.out.println("Було вказано від'ємну кількість секунд, вкажіть нове значення:");  InputValue = InputScanner.nextInt();  }    InputScanner.close();  int hours = InputValue / 3600;  int minutes = (InputValue % 3600) / 60;  int seconds = InputValue % 60;  System.out.printf("%d:%02d:%02d\n", hours, minutes, seconds);  }  } |



Рисунок 1.2 - Демонстрація роботи коду

**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.

Спершу я використав конструкцію import java.util.Scanner;, щоб імпортувати клас Scanner, який дозволить мені отримувати дані від користувача. Потім я оголосив клас Task03, де також є вказав змінну pass, що містить значення пароля. В методі main, я створюю новий об'єкт Scanner з назвою InputScanner, щоб зчитувати введення користувача з консолі. Далі я виводжу на екран повідомлення "Введіть пароль:" за допомогою System.out.println(). Після цього я зчитую введене користувачем значення в цілочисельну змінну InputPass за допомогою методу nextInt() об'єкту InputScanner. Після зчитування я закриваю об'єкт InputScanner, використовуючи метод close(). Далі йде перевірка введеного користувачем пароля за допомогою оператора if. Я порівнюю введений пароль зі значенням, що зберігається у змінній pass, якщо вони співпадають, виводжу повідомлення "Hello, Agent", інакше - "Access denied".

|  |
| --- |
| import java.util.Scanner;    public class Task03 {  private static final int pass = 55555;    public static void main(String[] args) {  Scanner InputScanner = new Scanner(System.in);  System.out.println("Введіть пароль:");  int InputPass = InputScanner.nextInt();    InputScanner.close();  if (InputPass == pass) {  System.out.println("Hello, Agent");  } else {  System.out.println("Access denied");  }  }  } |

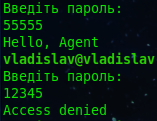


Рисунок 1.3 - Демонстрація роботи коду

**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.

Спершу я використовую конструкцію import java.util.Scanner;, щоб імпортувати клас Scanner, який дозволяє отримувати введені дані від користувача. Потім я оголошую клас Task04 і його метод main, у методі main я створюю новий об'єкт Scanner з назвою InputScanner, щоб зчитувати введення користувача з консолі. Я також оголошую змінну InputValue та ініціалізую її значенням 0, яке буде використано для зберігання кількості незнайомців, яких користувач хоче зустріти. Далі йде цикл while, який продовжується, доки введена кількість незнайомців (InputValue) менше або дорівнює 0. У циклі користувачу пропонується ввести кількість незнайомців, і я зчитую це значення за допомогою методу nextInt() об'єкту InputScanner. Якщо введене значення виявиться менше або дорівнює 0, програма виведе повідомлення про те, що кількість незнайомців має бути додатнім числом, і користувачу буде пропоновано ввести нове значення. Після цього я викликаю метод nextLine() об'єкту InputScanner, щоб очистити буфер вводу після зчитування числа. Далі програма виводить повідомлення "Введіть імена незнайомців" і після чого використовується цикл for, який перебирає кожного незнайомця, якого хоче зустріти користувач. Кожного разу виводиться повідомлення про введення імені незнайомця під певним номером, після чого за допомогою методу nextLine() об'єкту InputScanner зчитується введене ім'я незнайомця в рядок StrangerName і після чого, програма виводить привітання з кожним незнайомцем, використовуючи "Hello, " і їхнє ім'я. Після цього об'єкт InputScanner закривається за допомогою методу close().

|  |
| --- |
| import java.util.Scanner;    public class Task04 {  public static void main(String[] args) {  Scanner InputScanner = new Scanner(System.in);  int InputValue = 0;    while (InputValue <= 0) {  System.out.println("Вкажіть кількість незнайомців для зустрічі:");  InputValue = InputScanner.nextInt();    if (InputValue <= 0) {  System.out.println("Кількість незнайомців має бути додатнім значенням, введіть нове значення:");  }  }    InputScanner.nextLine();    System.out.println("Введіть імена незнайомців");    for (int xx = 1; xx <= InputValue; xx++) {  System.out.println("Ім'я незнайомця під номером " + (xx) + ":");  String StrangerName = InputScanner.nextLine();  System.out.println("Hello, " + StrangerName);  }  InputScanner.close();  }  } |

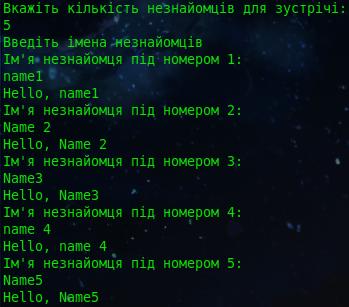


Рисунок 1.4 - Демонстрація роботи коду

**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

Спочатку я використовую конструкцію import java.util.Scanner;, щоб імпортувати клас Scanner, який дозволить мені отримувати введені дані від користувача. Потім я оголошую клас Task05 і його метод main. У методі main я створюю новий об'єкт Scanner з назвою InputScanner, щоб зчитувати введення користувача з консолі. Далі програма виводить на екран повідомлення, яке просить користувача ввести висоту дерева, швидкість руху равлика та швидкість спуску. Після цього програма зчитує введені користувачем значення в цілочисельні змінні: height, speed та MinusSpeed, за допомогою методу nextInt() об'єкту InputScanner. Після зчитування програма закриває об'єкт InputScanner, використовуючи метод close(), щоб уникнути витоку ресурсів. Далі програма викликає метод CalcDay(height, speed, MinusSpeed), який розраховує кількість днів, необхідних для підняття равлика на висоту height зі швидкістю speed та спуску зі швидкістю MinusSpeed. Цей метод використовує цикл while, щоб поступово додавати швидкість підняття та віднімати швидкість спуску, доки равлик не досягне вершини дерева. Коли равлик досягає або перевищує висоту дерева, повертається кількість днів, яка знадобилася для цього. Якщо равлик не може дістатися до вершини дерева, повертається значення -1. На основі результату, програма виводить відповідне повідомлення: якщо кількість днів дорівнює -1, виводиться "Impossible", інакше - "Кількість днів, які потрібні для підняття до вершини:" разом із розрахованою кількістю днів.

|  |
| --- |
| import java.util.Scanner;    public class Task05 {  public static void main(String[] args) {  Scanner InputScanner = new Scanner(System.in);  System.out.println("Введіть висоту дерева, швидкість равлика та швидкість спуску");  System.out.println("Висода дерева: ");  int height = InputScanner.nextInt();  System.out.println("Швидкість равлика: ");  int speed = InputScanner.nextInt();  System.out.println("Швидкість спуску равлика: ");  int MinusSpeed = InputScanner.nextInt();    InputScanner.close();  int day = CalcDay(height, speed, MinusSpeed);  if (day == -1) {  System.out.println("Impossible");  } else {  System.out.println("Кількість днів, які потрібні для підняття до вершини: " + day);  }  }  public static int CalcDay(int height, int speed, int MinusSpeed) {  int day = 0;  int HeightXX = 0;  while (HeightXX < height) {  HeightXX += speed;  day++;  if (HeightXX >= height) {  return day;  }  HeightXX -= MinusSpeed;  }  return -1;  }  } |

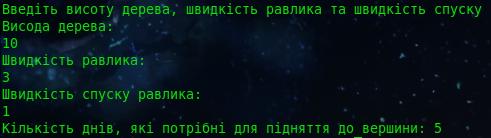


Рисунок 1.5 - Демонстрація роботи коду

**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.

Спершу я використовую конструкцію import java.util.Scanner;, щоб імпортувати клас Scanner, який дозволить отримати введені дані від користувача. Потім я оголошую клас Task06 і його метод main. У методі main я створюю новий об'єкт Scanner з назвою InputScanner, щоб зчитувати введення користувача з консолі. Далі програма виводить на екран повідомлення "Вкажіть загальну суму рахунку та кількість друзів:". Після цього програма зчитує введені користувачем значення: загальну суму рахунку в змінну bill за допомогою методу nextDouble(), та кількість друзів в змінну friends за допомогою методу nextInt() об'єкту InputScanner. Після зчитування програма закриває об'єкт InputScanner, використовуючи метод close(). Далі програма перевіряє, чи введені значення bill і friends більше за 0. Якщо bill менше або дорівнює 0, програма виводить повідомлення про те, що загальна сума рахунку має бути більше 0, і завершує виконання методу за допомогою return;. Те ж саме робиться і для перевірки кількості друзів. Якщо friends менше або дорівнює 0, програма виводить повідомлення про те, що кількість друзів має бути більше 0, і також завершує виконання методу. Якщо обидва введені значення є коректними, програма обчислює суму оплати на одного друга за допомогою формули bill \* 1.1 / friends, множення на 1.1 відбувається, оскільки друзі домовилися залишити 10% в якості чайових, після чого та зберігає результат у змінну result. В кінці програма виводить результат за допомогою методу printf(), де форматує результат до двох десяткових знаків після коми, і виводить повідомлення "Сума оплати на одного друга: " і розраховану суму.

|  |
| --- |
| import java.util.Scanner;    public class Task06 {  public static void main(String[] args) {  Scanner InputScanner = new Scanner(System.in);  System.out.println("Вкажіть загальну суму рахунку та кількість друзів:");  double bill = InputScanner.nextDouble();  int friends = InputScanner.nextInt();    InputScanner.close();  if (bill <= 0) {  System.out.println("Загальна сума рахунку має бути більше 0.");  return;  }  if (friends <= 0) {  System.out.println("Кількість друзів має бути більше 0.");  return;  }  double result = bill \* 1.1 / friends;  System.out.printf("Сума сплати на одного друга: %.2f\n", result);  }  } |

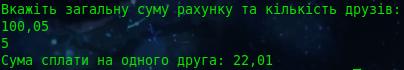


Рисунок 1.6 - Демонстрація роботи коду

**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.

Спершу я використовую конструкцію import java.util.Scanner;, щоб імпортувати клас Scanner, який дозволяє отримати введені дані від користувача. Потім я оголошую клас Task07 і його метод main, а у методі main я створюю новий об'єкт Scanner з назвою InputScanner, щоб зчитувати введення користувача з консолі. Далі програма виводить на екран повідомлення "Введіть послідовність цілих чисел та 0 для завершення:". Після цього програма оголошує змінну MaxInt та ініціалізує її значенням Integer.MIN\_VALUE, що є найменшим можливим значенням для цілого числа. Після цього програма використовує безкінечний цикл while (true), щоб продовжувати зчитування чисел, поки користувач не введе 0. У кожній ітерації циклу програма зчитує наступне введене число користувача за допомогою методу nextInt() об'єкту InputScanner та зберігає його у змінну Value. Якщо Value дорівнює 0, програма виходить з циклу за допомогою ключового слова break. Інакше, якщо Value більше, ніж поточне максимальне значення MaxInt, значення MaxInt оновлюється. Після завершення введення чисел програма закриває об'єкт InputScanner, використовуючи метод close() і після чого програма виводить на екран повідомлення "Максимальне значення: " і значення із змінної MaxInt, яке є найбільшим числом у введеній послідовності.

|  |
| --- |
| import java.util.Scanner;    public class Task07 {  public static void main(String[] args) {  Scanner InputScanner = new Scanner(System.in);  System.out.println("Введіть послідовність цілих чисел та 0 для завершення:");  int MaxInt = Integer.MIN\_VALUE;  while (true) {  int Value = InputScanner.nextInt();  if (Value == 0) {  break;  }  if (Value > MaxInt) {  MaxInt = Value;  }  }    InputScanner.close();  System.out.println("Максимальне значення: " + MaxInt);  }  } |

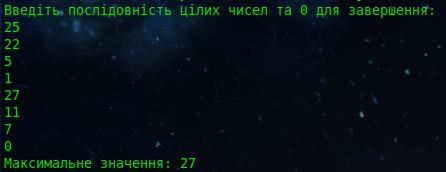


Рисунок 1.7 - Демонстрація роботи коду

**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.

Спершу я використовую конструкцію import java.util.Scanner;, щоб імпортувати клас Scanner, який дозволить отримати введені дані від користувача. Потім я оголошую клас Task08 і його метод main. У методі main я створюю новий об'єкт Scanner з назвою InputScanner, щоб зчитувати введення користувача з консолі. Далі програма виводить на екран повідомлення "Введіть послідовність цілих чисел та 0 для завершення:", після цього програма оголошує та ініціалізує дві змінні: NumbSum для зберігання суми введених чисел та NumbCount для підрахунку кількості введених чисел. Далі програма використовує безкінечний цикл while (true), щоб продовжувати зчитування чисел, поки користувач не введе 0. У кожній ітерації циклу програма зчитує наступне введене число користувача за допомогою методу nextInt() об'єкту InputScanner та зберігає його у змінну value. Якщо value дорівнює 0, програма виходить з циклу за допомогою ключового слова break, а якщо ні, то сума NumbSum збільшується на value, а кількість чисел NumbCount збільшується на 1. Після завершення введення чисел програма закриває об'єкт InputScanner, використовуючи метод close(). В кінці програма виводить на екран повідомлення "Середнє значення: " разом із значенням середнього значення, яке обчислюється як відношення суми введених чисел до їх кількості, приведене до типу double для точності.

|  |
| --- |
| import java.util.Scanner;    public class Task08 {  public static void main(String[] args) {  Scanner InputScanner = new Scanner(System.in);  System.out.println("Введіть послідовність цілих чисел та 0 для завершення:");  int NumbSum = 0;  int NumbCount = 0;  while (true) {  int value = InputScanner.nextInt();  if (value == 0) break;  NumbSum += value;  NumbCount++;  }    InputScanner.close();  System.out.println("Середнє значення: " + ((double) NumbSum / NumbCount));  }  } |

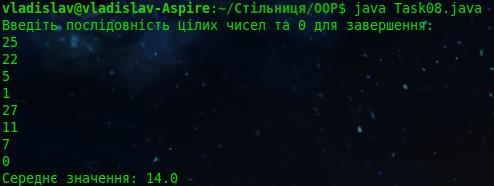


Рисунок 1.8 - Демонстрація роботи коду

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

Спочатку я використовую конструкцію import java.util.Scanner;, щоб імпортувати клас Scanner, який дозволяє отримувати введені дані від користувача. Потім я оголошую клас Task09 і його метод main, у цьому методі я створюю новий об'єкт Scanner з назвою InputScanner, щоб зчитувати введення користувача з консолі. Далі програма виводить на екран повідомлення "Введіть коефіцієнти a, b і c із квадратного рівняння ax^2 + bx + c = 0". Після цього програма просить користувача ввести коефіцієнти a, b і c квадратного рівняння. Для введення кожного коефіцієнта програма використовує метод nextDouble() об'єкту InputScanner, який дозволяє зчитати введене користувачем дійсне число. Після зчитування кожного коефіцієнта програма закріплює його значення змінними a, b і c. Після закриття об'єкта InputScanner, за допомогою методу close(), програма обчислює дискримінант рівняння за формулою b^2 – 4 \* a \* c. Після обчислення дискримінанту програма перевіряє його значення. Якщо дискримінант більше за 0, то рівняння має два дійсних корені, які обчислюються за допомогою формул Квадратного кореня. Якщо дискримінант дорівнює 0, то рівняння має один дійсний корінь, який обчислюється окремо. Якщо ж дискримінант менше за 0, то рівняння не має дійсних коренів, і програма виводить відповідне повідомлення на екран.

|  |
| --- |
| import java.util.Scanner;    public class Task09 {  public static void main(String[] args) {  Scanner InputScanner = new Scanner(System.in);  System.out.println("Введіть коефіцієнти a, b і c із квадратного рівняння ax^2 + bx + c = 0");  System.out.print("a = ");  double a = InputScanner.nextDouble();  System.out.print("b = ");  double b = InputScanner.nextDouble();  System.out.print("c = ");  double c = InputScanner.nextDouble();  InputScanner.close();    double discrim = b\*b-4\*a\*c;  if (discrim > 0) {  double xx1 = (-b+Math.sqrt(discrim))/(2\*a);  double xx2 = (-b-Math.sqrt(discrim))/(2\*a);  System.out.println("Корені рівняння: " + xx1 + ", " + xx2);  } else if (discrim == 0) {  double xx = -b / (2 \* a);  System.out.println("Знайдено один дійсний корінь рівняння: " + xx);  } else {  System.out.println("Рівняння не має дійсних коренів");  }  }  } |

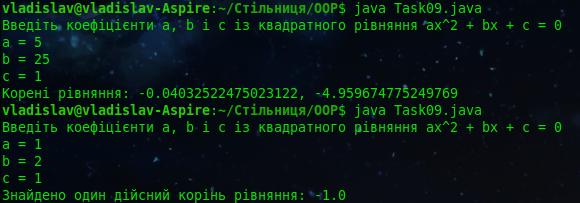


Рисунок 1.9 - Демонстрація роботи коду

**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.

Спочатку в методі main створюється масив arr з вже заданими значеннями цілих чисел, після чого викликається метод max(arr), де масив arr передається як аргумент. Результат, отриманий з методу max, зберігається у змінній result, після чого виводиться на екран разом із підписом "Максимальне значення в масиві:". Тепер перейдемо до самого масиву max, який приймає масив цілих чисел у якості параметру. У цьому методі спочатку ініціалізується змінна MaxNum значенням першого елементу масиву arr. Потім використовується цикл for, щоб пройтися по всіх елементах масиву, починаючи з індексу 1 (оскільки ми вже взяли перший елемент у змінну MaxNum). У кожній ітерації циклу перевіряється, чи поточний елемент масиву arr[xx] більший за поточне максимальне значення MaxNum. Якщо так, то MaxNum оновлюється значенням поточного елементу. По закінченню циклу метод повертає знайдене максимальне значення і таким чином, програма знаходить найбільше значення у заданому масиві та виводить його в консоль.

|  |
| --- |
| public class Task10 {  public static void main(String[] args) {  int[] arr = {5, 1, 2, 8, 43, 55, 12, 4};  int result = max(arr);  System.out.println("Максимальне значення в масиві: " + result);  }  public static int max(int[] arr) {  int MaxNum = arr[0];    for (int xx = 1; xx < arr.length; xx++) {  if (arr[xx] > MaxNum) {  MaxNum = arr[xx];  }  }    return MaxNum;  }  } |



Рисунок 1.10 - Демонстрація роботи коду

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.

Спочатку було створено метод main в якому було створено масив arr з вже заданими значеннями цілих чисел. Потім викликається метод Sum(arr), де масив arr передається як аргумент. Результат, отриманий з методу Sum, зберігається у змінній result, після чого виводиться в консоль разом із текстом "Сума парних чисел в масиві:". Тепер перейдемо до методу Sum, який приймає масив цілих чисел у якості параметру. У цьому методі спочатку ініціалізується змінна NumSum значенням 0, яка використовуватиметься для накопичення суми парних чисел. Потім використовується похідний цикл for, щоб пройтися по всіх елементах масиву arr. На кожній ітерації циклу поточне значення xx приймає значення наступного елементу масиву, у тілі циклу перевіряється, чи є поточне число парним, перевірка здійснюється за допомогою операції %, за її допомогою ми визначаємо, чи має значення залишок при діленні на 2, чи ні. Якщо число парне, то воно додається до суми NumSum. По закінченню циклу метод повертає обчислену суму парних чисел і таким чином, програма обчислює суму всіх парних чисел у масиві та виводить її.

|  |
| --- |
| public class Task11 {  public static void main(String[] args) {  int[] arr = {5, 1, 2, 8, 43, 55, 12, 4};  int result = Sum(arr);  System.out.println("Сума парних чисел в масиві: " + result);  }  public static int Sum(int[] arr) {  int NumSum = 0;  for (int xx : arr) {  if (xx % 2 == 0) {  NumSum += xx;  }  }  return NumSum;  }  } |



Рисунок 1.11 - Демонстрація роботи коду

**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.

Спочатку було створено метод main і масив arr в йому, в цей масив було внесено деякі значеннями цілих чисел. Потім викликається метод getSumCheckArray(arr), де масив arr передається як аргумент. Результат, отриманий з методу getSumCheckArray, зберігається у змінній result, після чого виводиться в консоль у вигляді рядка за допомогою методу java.util.Arrays.toString(result). Далі я створив метод getSumCheckArray, який приймає масив цілих чисел у якості параметру. У цьому методі спочатку створюється новий масив логічних значень result з тим самим розміром, що й вхідний масив arr. Потім використовується цикл for, щоб пройтися по всіх елементах масиву arr починаючи з індексу 2, бо для перевірки потрібні два попередні елементи. На кожній ітерації циклу перевіряється, чи виконується умова, що поточний елемент arr[xx] дорівнює сумі двох попередніх елементів arr[xx - 1] і arr[xx - 2]. Результат перевірки записується у відповідний елемент масиву result. По закінченню циклу метод повертає масив логічних значень result. Таким чином, програма перевіряє кожний елемент масиву на умову, що він є сумою двох попередніх елементів, і створює масив логічних значень, який відображає результат цієї перевірки для кожного елемента.

|  |
| --- |
| public class Task12 {  public static void main(String[] args) {  int[] arr = {5, 5, 10, 4, 14, 2, 1, 3};  boolean[] result = getSumCheckArray(arr);  System.out.println("Результат: " + java.util.Arrays.toString(result));  }  public static boolean[] getSumCheckArray(int[] arr) {  boolean[] result = new boolean[arr.length];  for (int xx = 2; xx < arr.length; xx++) {  result[xx] = (arr[xx] == arr[xx - 1] + arr[xx - 2]);  }  return result;  }  } |



Рисунок 1.12 - Демонстрація роботи коду

**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. **Локальний максимум** — це елемент, який більший за будь-який з його сусідніх елементів. Вам необхідно видалити елементи, які є локальними максимумами у вихідному масиві. Розмір цього масиву гарантовано більше 1. 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.\*.

Спочтку було створено метод main, в якому створюється масив arr в який було задано значеннями цілих чисел. Після цього викликається метод removeLocalMaxima(arr), де масив arr передається як аргумент, а результат, отриманий з методу removeLocalMaxima, зберігається у змінній result, після чого виводиться у вигляді рядка за допомогою методу Arrays.toString(result). Далі я виконав розробку методу removeLocalMaxima, який приймає масив цілих чисел у якості параметру. У цьому методі спочатку обчислюється кількість локальних максимумів у масиві, для цього використовується цикл for, який проходиться по всіх елементах масиву, крім першого та останнього, бо вони не можуть бути локальними максимумами, і перевіряється, чи є поточний елемент більшим за своїх сусідів. Якщо ця умова виконується, збільшується лічильник count. Далі створюється новий масив result, розмір якого дорівнює різниці між розміром вхідного масиву і кількістю локальних максимумів. Потім використовується другий цикл for, щоб пройтися по всіх елементах вхідного масиву, якщо поточний елемент не є локальним максимумом, тобто він не більший за обидва своїх сусіди, він копіюється до нового масиву result. Змінна ind використовується для відстеження поточного індексу у новому масиві result, в кінці метод повертає новий масив без локальних максимумів. Таким чином програма створює копію вихідного масиву, але без локальних максимумів і відображає цю копію користувачеві.

|  |
| --- |
| import java.util.Arrays;    public class Task13 {  public static void main(String[] args) {  int[] arr = {5, 2, 13, 8, 3, 4};  int[] result = removeLocalMaxima(arr);  System.out.println("Копія масиву: " + Arrays.toString(result));  }  public static int[] removeLocalMaxima(int[] arr) {  int count = 0;  for (int xx = 1; xx < arr.length - 1; xx++) {  if (arr[xx] > arr[xx - 1] && arr[xx] > arr[xx + 1]) {  count++;  }  }  int[] result = new int[arr.length - count];  int ind = 0;  for (int xx = 0; xx < arr.length; xx++) {  if (xx == 0 || xx == arr.length - 1 || !(arr[xx] > arr[xx - 1] && arr[xx] > arr[xx + 1])) {  result[ind++] = arr[xx];  }  }  return result;  }  } |



Рисунок 1.13 - Демонстрація роботи коду

**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.

Я почав виконання цього завдання із створення методу main, у якому задано масив arr із заданими значеннями цілих чисел. Потім викликаються методи cycleSwap(arr) та cycleSwap(arr, shift), де масив arr передається як аргумент. Перший виклик здійснює зсув масиву на одну позицію вправо, а другий - зсув на вказану кількість позицій в змінній shift, у моєму випадку на 7 вправо. Після кожного зсуву результат виводиться на екран у вигляді рядка за допомогою методу Arrays.toString(arr). Далі я створив метод cycleSwap(int[] arr), який здійснює зсув масиву на одну позицію вправо. Спочатку в методі перевіряється, чи довжина масиву більше одного елемента, якщо це не так, метод завершується без змін,а в протилежному випадку останній елемент масиву зберігається в змінній Box, а потім кожен елемент масиву пересувається на одну позицію вправо. На завершення, перший елемент масиву встановлюється значенням із змінної Box, яке ми зберегли раніше. Метод cycleSwap(int[] arr, int shift) приймає масив та кількість позицій для зсуву вправо. У цьому методі викликається метод cycleSwap(arr) стільки разів, скільки вказано в аргументі shift, що призводить до кількісного зсуву масиву на задану кількість позицій і таким чином, після виконання цього коду масив буде зсунуто вправо на одну позицію, а потім ще на 7 позицій.

|  |
| --- |
| import java.util.Arrays;    public class Task14 {  public static void main(String[] args) {  int[] arr = {5, 1, 4, 2, 7};  System.out.println("Початковий масив 1: " + Arrays.toString(arr));  cycleSwap(arr);  System.out.println("Масив після зсуву: " + Arrays.toString(arr));    int shift = 7;  cycleSwap(arr, shift);  System.out.println("Масив після зсуву на " + shift + " позицій: " + Arrays.toString(arr));  }  public static void cycleSwap(int[] arr) {  if (arr.length <= 1) {  return;  }  int Box = arr[arr.length - 1];  for (int xx = arr.length - 1; xx > 0; xx--) {  arr[xx] = arr[xx - 1];  }  arr[0] = Box;  }  public static void cycleSwap(int[] arr, int shift) {  for (int xx = 0; xx < shift; xx++) {  cycleSwap(arr);  }  }  } |



Рисунок 1.14 - Демонстрація роботи коду

**Висновок:** Під час виконання цієї лабораторної роботи я успішно виконав низку завдань, що сприяло покращенню мого розуміння та застосування теоретичних аспектів програмування на мові Java. Кожне завдання вимагало використання різноманітних концепцій, таких як умовні оператори, цикли, масиви, та роботу з введенням користувача. Ця лабораторна робота сприяла покращенню навичок програмування, розвитку логічного мислення та алгоритмічного проектування, кожне завдання відобразило важливі аспекти роботи з даними, умовами, циклами та масивами, що є фундаментальними для розуміння програмування в мові Java. В результаті виконання цієї лабораторної роботи, я покращив загальний рівень вмінь та навичок у програмуванні на мові Java.