МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА «Програмна інженерія та інтелектуальні технології управління»

ЗВІТ

з лабораторної роботи № 1

з навчальної дисципліни

“ОСНОВИ ПРОГРАМУВАННЯ JAVA (ДВВ)”

ВИКОНАЛА

Студентка групи КН-222а

Інеса РЕПЕШКО

ПЕРЕВІРИВ

доц. каф. ПІІТУ к.т.н

доцент Валерій ВОЛОВЩИКОВ

Харків 2023

# **Тема роботи**

“Архітектура та синтаксис Java”.

# **1.1 Завдання №1 “Індивідуальне завдання”**

Створити консольну програму, в якій здійснюється обчислення значень функції на певному інтервалі. У програмі треба визначити значення початку інтервалу, кінця інтервалу, а також величини кроку, з яким змінюється аргумент. Відповідні значення слід прочитати з клавіатури.

Програма повинна містити визначення та введення необхідних даних і один великий цикл, у тілі якого здійснюється:

* обчислення функції одним з варіантів, залежно від значення аргументу;
* виведення на консоль аргументу та результату на кожному кроці циклу;
* збільшення значення аргументу на величину кроку і перехід на наступне обчислення, якщо необхідно.

Програма повинна складатися з одного класу з двома статичними функціями:

* окрема статична функція для обчислення значення *y* залежно від значення аргументу *x*;
* метод main(), в якому здійснюється читання вихідних даних, а також цикл обчислення функції та виведення значень *x* та *y*.

Варіант функціональної залежності, який слід реалізувати у програмі, визначається відповідно до номеру студента у списку групи. Визначену константу *n* описати з модифікатором **final**. Для форматованого виведення результатів застосувати функцію printf().

Умова завдання для варіанту 4 (номер 24 за порядком у списку групи) наведено нижче.

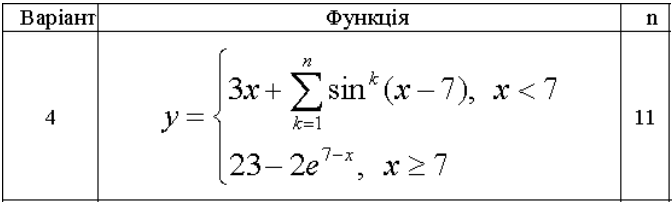


Рисунок 1.1 - Умова завдання № 1 варіант 4

**1.2 Набір тестових даних завдання № 1**

**1.2.1 Тестовий випадок № 1 “Введення валідних даних: start<end”**

Тестові дані:

* Початок інтервалу (start): “-6”.
* Кінець інтервалу (end): “10,5”.
* Крок, з яким змінюється аргумент (step): “3,0”.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами обчислення значення функції: “x = -6 y = -18,295879

x = -3 y = -7,808390

x = 0 y = -0,400397

x = 3 y = 11,966729

x = 6 y = 17,474605

x = 9 y = 22,729329”.

**1.2.2 Тестовий випадок № 2 “Введення валідних даних: start=end && step>(end-start)”**

Тестові дані:

* Початок інтервалу (start): “8”.
* Кінець інтервалу (end): “8”.
* Крок, з яким змінюється аргумент (step): “6,2”.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами обчислення значення функції: “x = 8 y = 22,264241”.

**1.2.3 Тестовий випадок № 3 “Введення невалідних даних: start>end”**

Тестові дані:

* Початок інтервалу (start): “332”.
* Кінець інтервалу (end): “15”.
* Крок, з яким змінюється аргумент (step): “8”.

Очікуваний результат: Завершення виконання програми з помилкою. Виведення у консоль повідомлення про помилку: “Invalid interval or step values were entered. Try again!”

**1.2.4 Тестовий випадок № 4 “Введення невалідних даних: step<=0”**

Тестові дані:

* Початок інтервалу (start): “1111,01234”.
* Кінець інтервалу (end): “-98,76”.
* Крок, з яким змінюється аргумент (step): “-200”.

Очікуваний результат: Завершення виконання програми з помилкою. Виведення у консоль повідомлення про помилку: “Invalid interval or step values were entered. Try again!”

**1.3 Програмний код завдання № 1**

Лістинг 1.1 - Програмний код реалізований у файлі “ComputationFunction.java”

|  |
| --- |
| package task1;  import java.util.Scanner;  public class ComputationFunction {  static final int *ITERATIONS\_COUNT* = 17;   public static double compute(double x) { |

|  |
| --- |
| double y;   if (x < 7) {  y = 3 \* x;   for (int k = 1; k <= *ITERATIONS\_COUNT*; k++) {  y += Math.*pow*(Math.*sin*(x - 7), k);  }  } else {  y = 23 - (2 \* Math.*exp*(7 - x));  }   return y;  }   public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);   System.*out*.print("Enter the value of the start of the interval:\t");  double start = scanner.nextDouble();   System.*out*.print("Enter the value of the end of the interval:\t\t");  double end = scanner.nextDouble();   System.*out*.print("Enter the value of the step size:\t");  double step = scanner.nextDouble();   if (start > end || step <= 0) {  System.*out*.println("Invalid interval or step values were entered. Try again!");  System.*exit*(1);  }   System.*out*.println("The result of calculating the function:");   for (double x = start; x <= end; x += step) {  double y = *compute*(x);  System.*out*.printf("x = %f\ty = %f\n", x, y);  }  } } |

**1.4 Екранні форми за результатами роботи програмного коду**

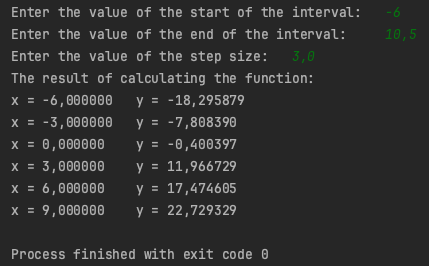


Рисунок 1.2 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Введення валідних даних: start<end”

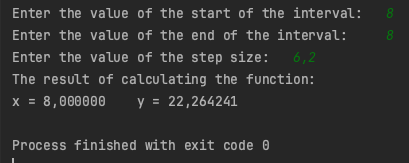


Рисунок 1.3 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Введення валідних даних: start=end && step>(end-start)”

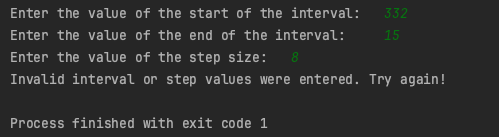


Рисунок 1.4 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Введення невалідних даних: start>end”

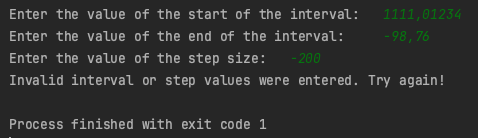


Рисунок 1.5 - Результати роботи програмного коду відповідно до тестового випадку № 4 “Введення невалідних даних: step<=0”

**2.1 Завдання № 2 “Степені числа 8”**

Увести значення n (від 0 до 10) і вивести значення степенів числа 8 до n включно. Реалізувати два підходи – з використанням арифметичних і побітових операцій.

**2.2 Набір тестових даних завдання № 2**

**2.2.1 Тестовий випадок № 1 “Введення валідних даних: n>0 && n<10”**

Тестові дані:

* Значення степенів числа 8 (n): “5”.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами обчислення значення з використанням арифметичних і побітових операцій: “Results of calculating 8 to the power of 5 (including):

Using arithmetic operations:

8 ^ 0 = 1

8 ^ 1 = 8

8 ^ 2 = 64

8 ^ 3 = 512

8 ^ 4 = 4096

8 ^ 5 = 32768

Using bitwise operations:

8 ^ 0 = 1

8 ^ 1 = 8

8 ^ 2 = 64

8 ^ 3 = 512

8 ^ 4 = 4096

8 ^ 5 = 32768”.

**2.2.2 Тестовий випадок № 2 “Введення невалідних даних: n<0”**

Тестові дані:

* Значення степенів числа 8 (n): “-25”.

Очікуваний результат: Завершення виконання програми з помилкою. Виведення у консоль повідомлення про помилку: “The value n does not belong to the valid range from 0 to 10 including.”

**2.2.3 Тестовий випадок № 3 “Введення невалідних даних: n>10”**

Тестові дані:

* Значення степенів числа 8 (n): “111”.

Очікуваний результат: Завершення виконання програми з помилкою. Виведення у консоль повідомлення про помилку: “The value n does not belong to the valid range from 0 to 10 including.”

**2.3 Програмний код завдання № 2**

Лістинг 2.1 - Програмний код реалізований у файлі “PowerOfNumber.java”

|  |
| --- |
| package task2;  import java.util.Scanner;  public class PowerOfNumber {  public static void main(String[] args) { |

|  |
| --- |
| Scanner scanner = new Scanner(System.*in*);   System.*out*.print("Enter a value for n in the range of 0 to 10 including:\t");  int n = scanner.nextInt();   if (n < 0 || n > 10) {  System.*err*.println("The value n does not belong to the valid range from 0 to 10 including.");  System.*exit*(1);  }   int base = 8;  System.*out*.printf("\nResults of calculating %d to the power of %d (including):\n", base, n);  System.*out*.println("\nUsing arithmetic operations:");   for (int i = 0; i <= n; i++) {  int result = (int) Math.*pow*(base, i);  System.*out*.printf("%d ^ %d = %d\n", base, i, result);  }   System.*out*.println("\nUsing bitwise operations:");   for (int i = 0; i <= n; i++) {  int result = 1 << 3 \* i;  System.*out*.printf("%d ^ %d = %d\n", base, i, result);  }  } } |

**2.4 Екранні форми за результатами роботи програмного коду**

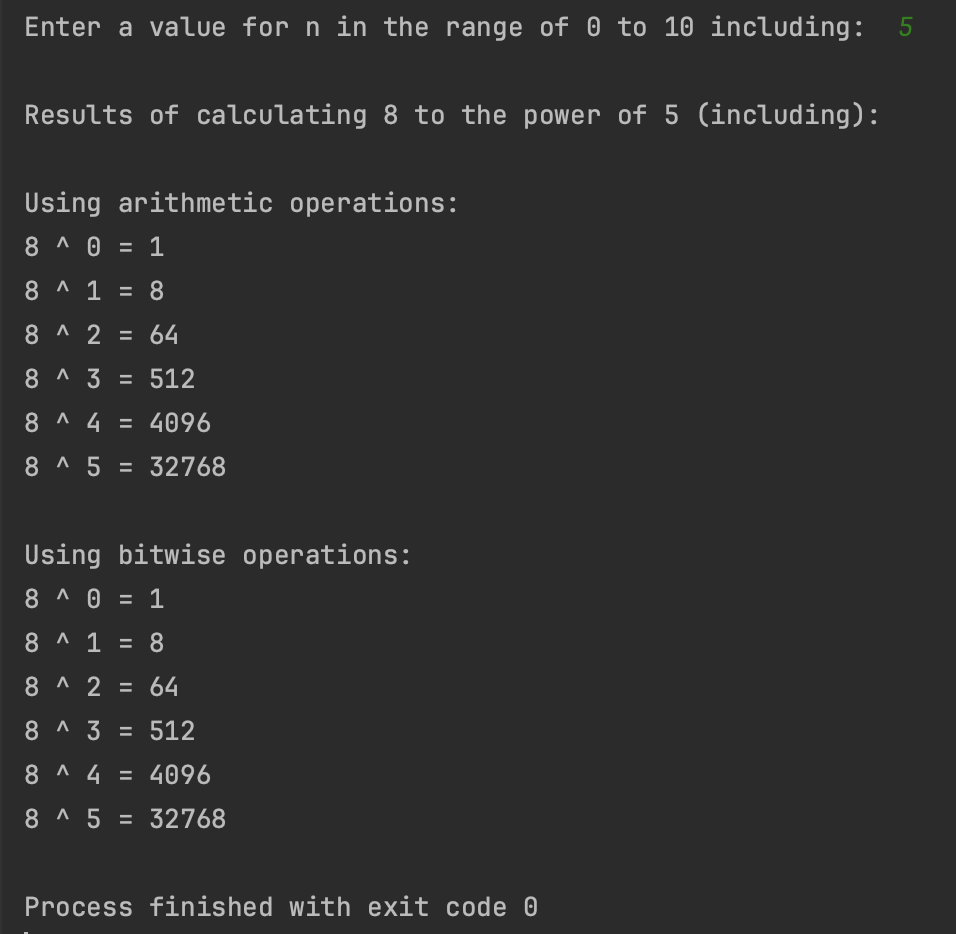


Рисунок 2.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Введення валідних даних: n>0 && n<10”

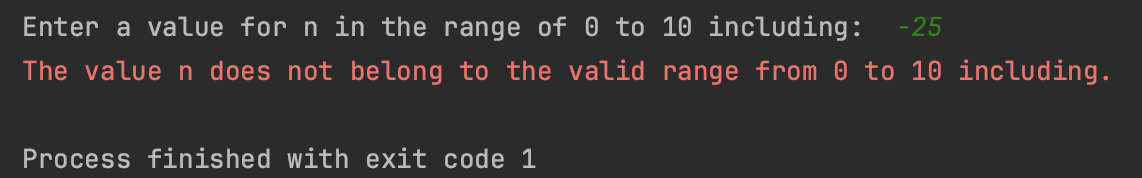


Рисунок 2.2 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Введення невалідних даних: n<0”

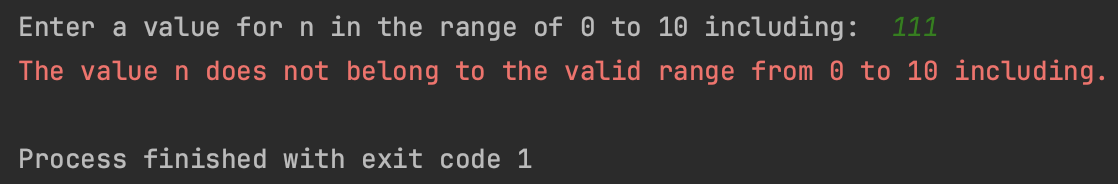


Рисунок 2.3 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Введення невалідних даних: n>10”

**3.1 Завдання № 3 “Використання break і continue з міткою”**

Увести значення *x* та *n*, обчислити та вивести результат виразу:

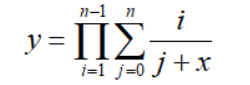


Рисунок 3.1 - Вираз функції для завдання № 3

Якщо *i* дорівнює *j + x*, перейти на наступний крок зовнішнього циклу без множення на проміжний результат. Для переходу на наступний крок застосувати **continue**з міткою.

Якщо знаменник на якійсь ітерації дорівнює 0, вийти з обох циклів з виведенням повідомлення про помилку. Для виходу застосувати **break** з міткою.

**3.2 Набір тестових даних завдання № 3**

**3.2.1 Тестовий випадок № 1 “Введення валідних даних: x>0 && n>0”**

Тестові дані:

* Значення аргументу (х): “-5,6”.
* Значення кроку (n): “4”.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами обчислення значення функції: “i = 1 sum(1) = -1,683356 y(1) = -1,683356

i = 2 sum(2) = -3,366712 y(2) = 5,667374

i = 3 sum(3) = -5,050068 y(3) = -28,620623

y = -28,620623”.

**3.2.2 Тестовий випадок № 2 “Введення валідних даних: x>0 && n>0 && (i == j + x)”**

Тестові дані:

* Значення аргументу (х): “6”.
* Значення кроку (n): “12”.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами обчислення значення функції: “i = 1 sum(1) = 1,211775 y(1) = 1,211775

i = 2 sum(2) = 2,423549 y(2) = 2,936796

i = 3 sum(3) = 3,635324 y(3) = 10,676206

i = 4 sum(4) = 4,847099 y(4) = 51,748627

i = 5 sum(5) = 6,058874 y(5) = 313,538395

y = 313,538395”.

**3.2.3 Тестовий випадок № 3 “Введення невалідних даних: x>0 && n<=0”**

Тестові дані:

* Значення аргументу (х): “1,123456”.
* Значення кроку (n): “-9”.

Очікуваний результат: Завершення виконання програми з помилкою. Виведення у консоль повідомлення про помилку: “The value of n cannot be less than 1. Try again!”.

**3.2.4 Тестовий випадок № 4 “Введення невалідних даних: x<=0 && n>0 && (j + x == 0)”**

Тестові дані:

* Значення аргументу (х): “-2”.
* Значення кроку (n): “192”.

Очікуваний результат: Завершення виконання програми з помилкою. Виведення у консоль повідомлення про помилку: “Calculation error: division by zero is not possible. Try again!”

**3.3 Програмний код завдання № 3**

Лістинг 3.1 - Програмний код реалізований у файлі “BreakAndContinue.java”

|  |
| --- |
| package task3;  import java.util.Scanner;  public class BreakAndContinue {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);   System.*out*.print("Enter value of x:\t");  double x = scanner.nextDouble();   System.*out*.print("Enter value of n:\t");  int n = scanner.nextInt();   if (n < 1) {  System.*err*.println("The value of n cannot be less than 1. Try again!");  System.*exit*(1);  }   System.*out*.println("\nResults of the function calculation:");  double y = 1;   outerLoop:  for (int i = 1; i <= n - 1; i++) {  double sum = 0;  int j;   for (j = 0; j <= n; j++) {  if (i == j + x) {  continue outerLoop;  }   if (j + x == 0) {  System.*err*.println("Calculation error: division by zero is not possible. Try again!"); |

|  |
| --- |
| break outerLoop;  }   sum += i / (j + x);  }   y \*= sum;  System.*out*.printf("i = %d\tsum(%d) = %f\ty(%d) = %f\n", i, i, sum, i, y);  }   System.*out*.printf("\ny = %f\n", y);  } } |

**3.4 Екранні форми за результатами роботи програмного коду**

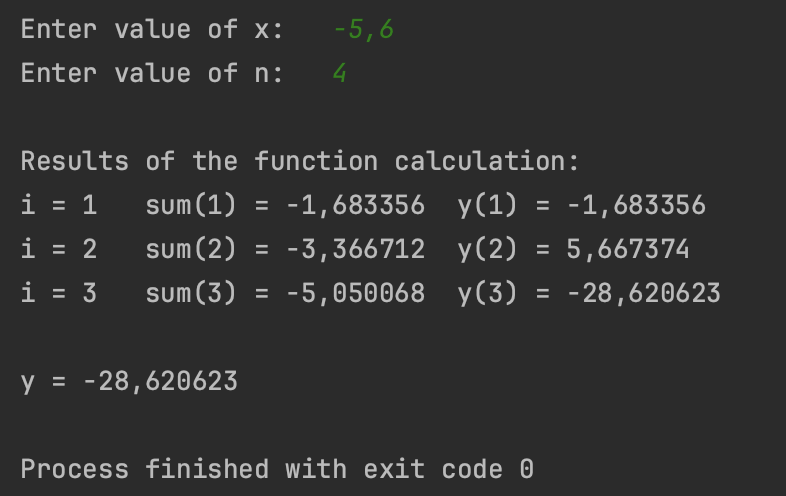


Рисунок 3.2 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Введення валідних даних: x>0 && n>0”

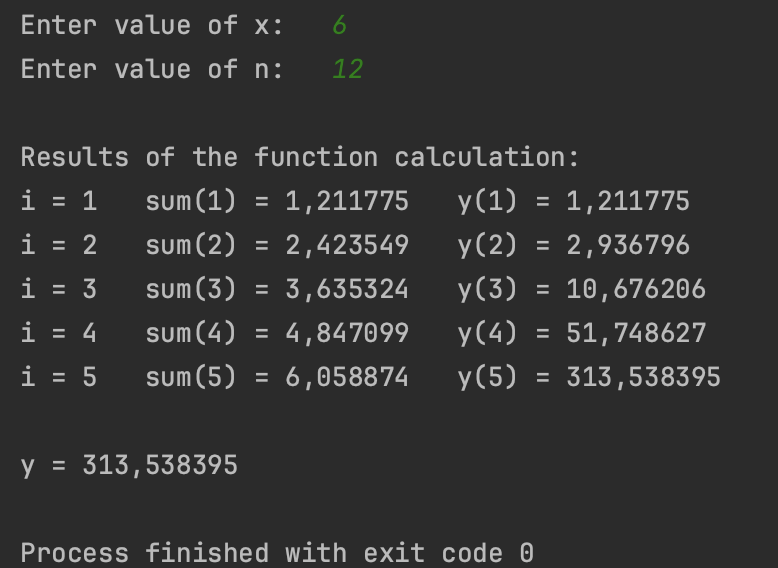


Рисунок 3.3 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Введення валідних даних: x>0 && n>0 && (i == j + x)”

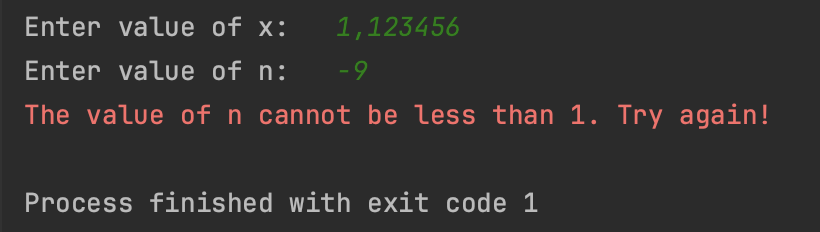


Рисунок 3.4 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Введення невалідних даних: x>0 && n<=0”

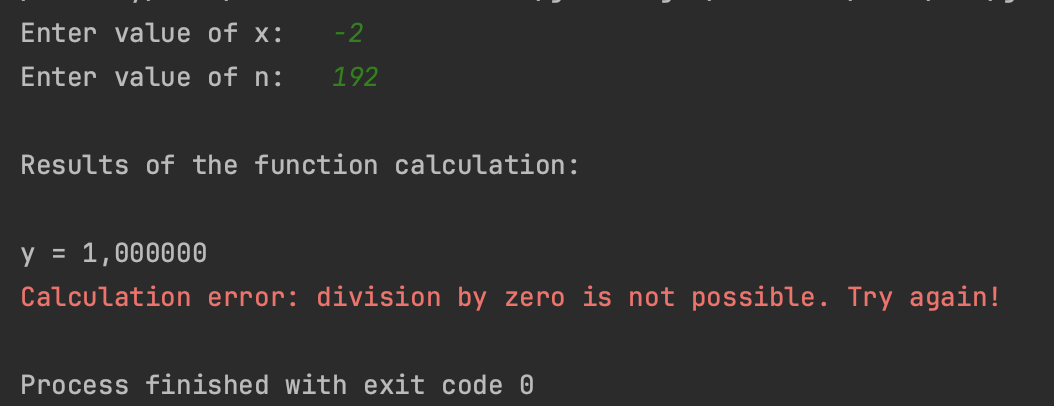


Рисунок 3.5 - Результати роботи програмного коду відповідно до тестового випадку № 4 “Введення невалідних даних: x<=0 && n>0 && (j + x == 0)”

## **4.1 Завдання № 4 “Бібліотека функцій для перетворення типів”**

Створити клас зі статичними функціями перетворення аргументу типу **boolean** у ціле значення (0 або 1) і навпаки (нуль – **false**, ненульове значення – **true**). Продемонструвати роботу функцій, викликавши їх з іншого класу іншого пакету. Застосувати звичайний і статичний імпорт.

**4.2 Набір тестових даних завдання № 4**

**4.2.1 Тестовий випадок № 1 “Конвертація значення типу integer у тип boolean (звичайний імпорт класу): 1->true”**

Тестові дані:

* Значення типу integer (intVariable): “1”.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами конвертації значення типу integer у тип boolean використовуючи звичайний імпорт іншого класу іншого пакету: “1 -> true”.

**4.2.2 Тестовий випадок № 2 “Конвертація значення типу integer у тип boolean (статичний імпорт класу): 0->false”**

Тестові дані:

* Значення типу integer (intVariable): “0”.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами конвертації значення типу integer у тип boolean використовуючи статичний імпорт іншого класу іншого пакету: “0 -> false”.

**4.2.3 Тестовий випадок № 3 “Конвертація значення типу integer у тип boolean (звичайний імпорт класу): -5->true”**

Тестові дані:

* Значення типу integer (intVariable): “-5”.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами конвертації значення типу integer у тип boolean використовуючи статичний імпорт іншого класу іншого пакету: “-5 -> true”.

**4.2.4 Тестовий випадок № 4 “Конвертація значення типу integer у тип boolean (статичний імпорт класу): 1000->true”**

Тестові дані:

* Значення типу integer (intVariable): “1000”.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами конвертації значення типу integer у тип boolean використовуючи статичний імпорт іншого класу іншого пакету: “1000 -> true”.

**4.2.5 Тестовий випадок № 5 “Конвертація значення типу bolean у тип integer (звичайний імпорт класу): true->1”**

Тестові дані:

* Значення типу boolean (boolVariable): “true”.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами конвертації значення типу boolean у тип integer використовуючи звичайний імпорт іншого класу іншого пакету: “true -> 1”.

**4.2.6 Тестовий випадок № 6 “Конвертація значення типу bolean у тип integer (статичний імпорт класу): false->0”**

Тестові дані:

* Значення типу boolean (boolVariable): “false”.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами конвертації значення типу boolean у тип integer використовуючи статичний імпорт іншого класу іншого пакету: “false -> 0”.

**4.3 Програмний код завдання № 4**

Структура файлів у проекті має наступний вигляд.

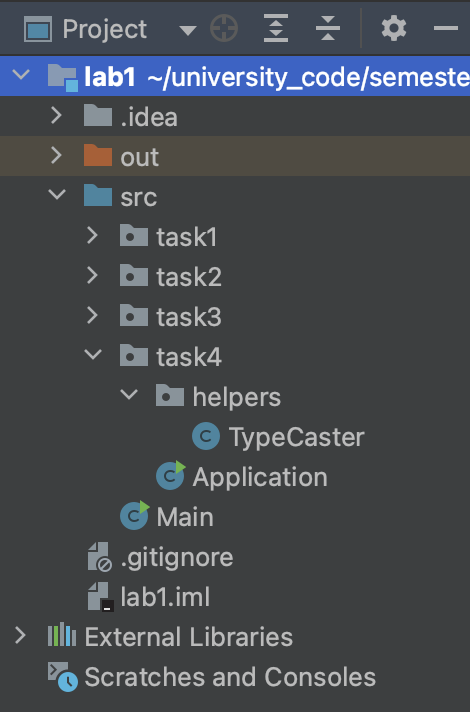


Рисунок 4.1 - Структура файлів у проекті

Лістинг 4.1 - Програмний код реалізований у файлі “TypeCaster.java”

|  |
| --- |
| package task4.helpers;  public class TypeCaster {  public static boolean cast(int value) {  return value != 0;  }  public static int cast(boolean value) {  return value ? 1 : 0;  } } |

Лістинг 4.2 - Програмний код реалізований у файлі “Application.java”

|  |
| --- |
| package task4;  import task4.helpers.TypeCaster;  import static task4.helpers.TypeCaster.\*;  public class Application {  public static void main(String[] args) {  int intVariable = 1;  System.*out*.printf("Integer to boolean:\t%d\t->\t%b\n", intVariable, TypeCaster.*cast*(intVariable));   intVariable = 0;  System.*out*.printf("Integer to boolean:\t%d\t->\t%b\n", intVariable, *cast*(intVariable));   intVariable = -5;  System.*out*.printf("Integer to boolean:\t%d\t->\t%b\n", intVariable, TypeCaster.*cast*(intVariable));   intVariable = 1000;  System.*out*.printf("Integer to boolean:\t%d\t->\t%b\n", intVariable, *cast*(intVariable));   boolean boolVariable = true;  System.*out*.printf("Boolean to integer:\t%b\t->\t%d\n", boolVariable, TypeCaster.*cast*(boolVariable));   boolVariable = false;  System.*out*.printf("Boolean to integer:\t%b\t->\t%d\n", boolVariable, *cast*(boolVariable));   } } |

**4.4 Екранні форми за результатами роботи програмного коду**

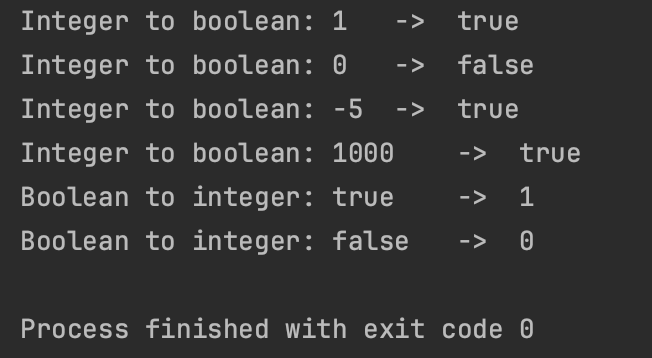


Рисунок 4.2 - Результати роботи програмного коду відповідно до тестових випадків № 1, 2, 3, 4, 5, 6

**Висновки до роботи:**

У ході виконання даної лабораторної роботи використано різні засоби, методи та технології для ефективної реалізації поставлених завдань із використанням мови програмування Java.

Під час виконання поставлених завдань опановано наступні функціональні можливості мови програмування Java:

* Введення даних з клавіатури із використанням класу Scanner для зчитування вхідних даних з консолі.
* Використання циклу for для ітерацій.
* Використання функції printf() для форматованого виведення результатів на консоль.
* Математичні операції із використанням арифметичних операцій для обчислення степенів числа.
* Використання міток для управління виконанням циклів.
* Використання операторів break і continue для контролю ходу виконання циклів в залежності від умов.
* Використання статичних методів у створенні класу для перетворення типів.
* Використання пакетів для виклику методів з іншого класу іншого пакету.
* Використання звичайного і статичного імпорту для спрощення доступу до методів класу.

Використання цих можливостей дозволяє навчитись створювати більш функціональні, ефективні та зрозумілі програми на мові Java, що є важливим для розробки якісного програмного забезпечення та розв'язання різноманітних завдань.