Лабораторна робота №1

Програмування лінійних алгоритмів.
Стандартні класи і методи мови Java

1.Створити клас, що має методи для обчислення на ЕОМ значень змінних, що зазначені у таблиці, за даними розрахунковими формулами і наборами вхідних даних.

2. Доповнити клас методом, що виводить на екран значення вхідних даних і результати обчислень, супроводжуючи вивід найменуваннями виведених змінних.

3. Доповнити клас методом main, що є необхідним для використання класу, як автономної програми, та виконати цю програму.

Варіант завдання	*Розрахункові формули*	*Значення вхідних даних*

1	$a=\frac{2\cos(x-\pi/6)}{1/2+\sin^2 y};b=1+\frac{z^2}{3+z^2/5}$	`x=1.426`
		`y=-1.220`
		`z=3.5`

2	$c=\left\|x^{y/x}-\sqrt[3]{y/x}\right\|;f=(y-x)\frac{y-z/(y-x)}{1+(y-x)^2}$	`x=1.825`
		`y=18.225`
		`z=-3.298`

3	$s=1+x+\frac{x^2}{2}+\frac{x^3}{6}+\frac{x^4}{24};f=x(\sin x^3+\cos^2 y)$	`x=0.335` `y=0.025`

4	$y=e^{-bt}\sin(at+b)-\sqrt{\left\|bt+a\right\|};s=b\sin(at^2\cos 2t)-1$	`a=-0.5`
		`b=1.7`
		`t=0.44`

5	$w=\sqrt{x^2+b}-b^2\sin^3(x+a)/x;y=\cos^2 x^3-\frac{x}{\sqrt{a^2+b^2}}$	`a=1.5`
		`b=15.5`
		`x=-2.8`

6	$s=x^3 tg^2(x+b)^2+\frac{a}{\sqrt{x+b}};Q=\frac{bx^2-a}{e^{ax}-1}$	`a=16.5`
		`b=3.4`
		`x=0.61`

7	$R=x^2(x+1)/b-\sin^2(x+a);s=\sqrt{\frac{xb}{a}}+\cos^2(x+b)^3$	`a=0.7`
		`b=0.05`
		`x=0.5`

8	$y=\sin^3(x^2+a)^2-\sqrt{\frac{x}{b}};z=\frac{x^2}{a}+\cos(x+b)^3$	`a=1.1`
		`b=0.004`
		`x=0.2`

9	$f=\sqrt[3]{m\,tg\,t+\left\|c\sin t\right\|};z=m\cos(b\,t\sin t)+c$	`m=2;c=-1`
		`t=1.2`
		`b=0.7`

10	$y=btg^2 x-\frac{a}{\sin^2(x/a)};d=ae^{-\sqrt{a}}\cos(bx/a)$	`a=3.2`
		`b=17.5`
		`x=-4.8`

Короткі теоретичні відомості

Огляд структури Java-програми

Всі Java-програми містять в собі 4 основні різновиди будівельних блоків: класи (classes), методи (methods), змінні (variables) і пакети (packages). На якій би мові Ви не програмували раніше, Ви скоріш за все вже добре знайомі з методами, які є не що інше, ніж функції чи підпрограми, та зі змінними, в яких зберігаються дані. З іншого боку, класи представляють собою фундамент об'єктно-орієнтованих властивостей мови. Поки що, для простоти, можна вважати клас деяким цілим, що містить у собі змінні и методи. Нарешті, пакети містять в собі класи і допомагають компілятору знайти ті класи, що потрібні йому для компіляції прикладної програми.

Java-програма може містити в собі будь-яку кількість класів, але один з них завжди має особливий статус, і безпосередньо взаємодіє з оболонкою часу виконання. Таким класом завжди сприймається клас, що визначений першим у тексті програми. Цей клас називають первинним класом (primary class).

Коли програма запускається з командного рядка, системі потрібен тільки один спеціальний метод, що повинен бути присутнім у первинному класі, - метод main. Коли ми будемо розглядати програмування аплетів, ми побачимо, що первинний клас аплета повинен містити вже декілька таких спеціальних методів.

Ось приклад простої автономної програми на мові Java:

```
import java.util.*; // імпортування класів стандартного пакету java.util за вимогою

public class OurPrimaryClass {
    public static void main(String[] S) {
        System.out.println("Hello, Java!");
        Date d=new Date();        // клас Date міститься у пакеті java.util
        System.out.println("Date: "+d.toString());
    }
}
```

Ця програма виводить на екран повідомлення "Hello, Java!" та поточну системну дату.

Стандартні типи даних Java

Всі змінні та вирази у мові програмування Java можуть бути віднесені до однієї з двох великих груп типів: примітивних типів (primitive types), або посилальних типів (reference types), що містять у собі типи, визначені користувачем, і типи масивів. До примітивних типів відносяться стандартні, вбудовані в мову типи для представлення чисельних значень, одиночних символів і логічних значень. Навпаки, усі посилальні типи є динамічними типами. Головні розбіжності між двома згаданими групами типів перелічені у наступній таблиці:

Таблиця 1.1 Порівняння примітивних і посилальних типів

Характеристика	Примітивні типи	Посилальні типи
Чи визначені в самій мові Java?	Так	Ні
Чи мають визначений розмір?	Так	Ні
Чи повинна для змінних цих типів виділятися пам'ять під час роботи програми?	Ні	Так

На практиці найважливішим розходженням між примітивними і посилальними типами є те, про що свідчить останній рядок цієї таблиці, а саме - що пам'ять для змінних посилального типу повинна виділятися під час виконання програми. Використовуючи змінні посилальних типів, ми повинні явно вимагати необхідну кількість пам'яті для кожної змінної перш, ніж ми зможемо зберегти в цієї змінний деяке значення. Причина цього проста: оболонка часу виконання сама по собі не знає, яка кількість пам'яті потрібна для того чи іншого посилального типу.

Спочатку розглянемо примітивні типи:
Усього в мові Java визначено вісім примітивних типів, що перелічені в табл.1.2.

Таблиця 1.2 Примітивні типи мови Java

Тип	Розмір	Діапазон	Приклад
`byte`	1 байт	від -128 до 127	125
`short`	2 байти	від -32768 до 32767	-23
`int`	4 байти	від -2147483648 до 2147483647	2002300
`long`	8 байт	від -922372036854775808 до 922372036854775807	1243565
`float`	4 байти	Залежить від розрядності числа	1.2f
`double`	8 байт	Залежить від розрядності числа	123.4
`boolean`	1 біт	false, true	True
`char`	2 байти	Усі символи стандарту Unicode	

Посилальні типи будуть розглянуті у наступних лабораторних роботах.

Стандартні математичні функції

Усі стандартні математичні функції в мові Java є статичними методами абстрактного класу Math, який визначений з модифікатором **final**, тобто не припускає спадкування.

Основні методи класу Math наведені у наступній таблиці.

Функція – метод	Пояснення
`Math.abs(x)`	Модуль числа x
`Math.acos(x)`	Арккосинус x
`Math.asin(x)`	Арксинус x
`Math.atan(x)`	Арктангенс x
`Math.cbrt(x)`	Кубічний корінь з X
`Math.ceil(x)`	Найближче число до x, що не містить дробової частини і більше за x
`Math.cos(x)`	Косинус x
`Math.exp(x)`	Експонента від x
`Math.floor(x)`	Найближче число до x, що не містить дробової частини і менше за x
`Math.hypot(x,y)`	Гіпотенуза прямокутного трикутника з катетами x та y
`Math.log(x)`	Натуральний логарифм x
`Math.max(x,y)`	Більше з двох чисел
`Math.min(x,y)`	Менше з двох чисел
`Math.pow(x,y)`	X в степені Y
`Math.random()`	Випадкове число з проміжку [0;1)
`Math.rint(x)`	Найближче число до x, що не містить дробової частини
`Math.round(x)`	Найближче до x ціле число
`Math.sin(x)`	Синус x
`Math.sqrt(x)`	Квадратний корінь з x
`Math.tan(x)`	Тангенс x
`Math.toDegrees(x)`	Переведення кута з радіанів у градуси
`Math.toRadians(x)`	Переведення кута з градусів у радіани

Крім того, клас Math має декілька визначених констант, наведемо дві з них:

Константа	Значення
Math.PI	Число π = 3.14159…
Math.E	Число e = 2.71828…

Примітка. Починаючи з версії j2sdk 5.0 (30.09.2004) у мові Java з'явилась можливість імпорту статичних змінних та методів класу за допомогою директиви `import static` на початку програми. Наприклад:

```
import static java.lang.Math.*; // імпортування статичних змінних і методів класу Math

public class OurPrimaryClass {
    public static void main(String[] S) {
        double x;
        x = sin(PI/6);   // раніше треба було писати x=Math.sin(Math.PI/6);
        System.out.println(x);
    }
}
```

Проте, використовувати цю можливість треба обережно, оскільки іноді це призводить до небажаних колізій

Введення - виведення даних у консолі Java-програм

Для виведення інформації на консоль використовуються методи стандартного класу PrintStream:

- print
- println
- printf
- format (точна копія printf)

Кожна програма на мові Java містить стандартний об'єкт типу PrintStream – System.out. Таким чином, виведення інформації на екран буде записуватися як System.out.print(…), System.out.println(…), або System.out.printf(…).

Методи print та println повинні завжди мати один параметр – вираз будь-якого типу, що може бути автоматично приведений до рядкового типу.

Наприклад,
`System.out.println("2+2="+(2+2));` // буде виведено **2+2=4**
`System.out.println("Значення суми="+s);` // буде виведено **Значення суми=xxx** , де xxx – значення змінної S

Методи **printf** та **format** можуть мати список параметрів, що розділяються комами. Перший параметр – рядок, що містить текст для виведення і форматні шаблони для виведення значень інших параметрів.

Наприклад, якщо a=2, b=3
System.out.printf("Значення %d + %d = %d", a, b, a+b); // буде виведено **Значення 2 + 3 = 5**

Форматні шаблони для виведення звичайних, символьних та числових типів мають наступний синтаксис: `% [індекс_аргумента$][опції][ширина][.точність]перетворення`

Необов'язковий параметр *індекс_аргумента* є цілим числом, що вказує позицію в списку аргументів. Посилання на перший аргумент буде записане як "1$", на другий – "2$", і т.д.

Необов'язковий параметр *опції* – це набір символів, що змінюють формат виведення. Набір припустимих опцій залежить від типу перетворення.

Необов'язковий параметр *ширина* – це невід'ємне ціле число, що показує мінімальну кількість символів, що їх треба вивести.

Необов'язковий параметр *точність* – це невід'ємне ціле число, що зазвичай використовується для обмеження кількості символів, що будуть виведені. Його дія залежить від параметру перетворення.

Обов'язковий параметр *перетворення* – це один символ, що вказує як аргумент буде відформатований. Набір припустимих перетворень для вказаного аргументу залежить від типу даних аргументу.

Форматні шаблони для виведення типів, що означають дату і час мають такий синтаксис: `% [індекс_аргумента$][опції][ширина]перетворення`

Індекс_аргумента, опції, ширина – описані вище, а перетворення – два символи, де перший – ‘t’, або ‘T’, а другий – описує тип перетворення.
Основні символи перетворень наведені у наступній таблиці

Перетворення	Категорія аргументу	Описання
'b', 'B'	general	Якщо аргумент *arg* є `null`, тоді результатом буде "`false`". Якщо *arg* належить до типу `boolean` або **`Boolean`**, то результатом буде рядок – ”true” або ”false” в залежності від значення *arg*. У всіх інших випадках результатом буде "true".
's', 'S'	general	Якщо аргумент *arg* є `null`, тоді результатом буде "`null`". Якщо *arg* має метод **`formatTo`**, `то він буде викликаний. Інакше, результат буде отриманий через виклик arg.toString().`
'c', 'C'	character	Результатом буде символ Unicode
'd'	integral	Результат буде відформатований, як ціле десяткове число
'e', 'E'	floating point	Результат буде відформатований, як число з плаваючою точкою у ”науковому” форматі
'f'	floating point	Результат буде відформатований, як десяткове число
'g', 'G'	floating point	Результат буде відформатований, як число у ”науковому” форматі, залежно від точності та значення після округлення.
't', 'T'	date/time	Префікс для символу перетворень дати і часу.
'%'	percent	Результатом буде символ '%' ('\u0025')
'n'	line separator	Результатом буде символ, що відокремлює рядки в залежності від платформи.

Про символи перетворення для дати і часу можна дізнатися з інтерактивної документації до j2sdk.

Приклади використання System.out.printf для виведення на екран

`System.out.printf(”Hello, World!”);`		`Hello, World!`
`System.out.printf(”Hello, World!%n”);` *або* `System.out.printf("Hello, World!\n");`		`Hello, World!`
`System.out.printf("Sum %d + %d = %d", a,b,a+b);`		`Sum 15 + 2 = 17`
`System.out.printf("Const of Pi = %8.6f",Math.PI);`		`Const of Pi = 3,141593`

Введення даних з консолі

Для введення даних у програмі на мові Java використовується спеціальний об’єкт, що належить до класу Scanner:

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

public class InOutExample {
    public static void main(String[] s) {
        Scanner s = new Scanner(System.in);
        // Читання цілого числа з рядка
        int i = s.nextInt();
        // Читання дійсного числа з рядку
        double x = s.nextDouble();
    }
}
```

Створення і виконання Java-програм у середовищі NetBeans

1. Створіть новий проект, для цього:
   - Після запуску NetBeans у головному меню програми оберіть:
   File -> New Project...
   - У вікні, що відкриється, оберіть категорію **Java** та вид проекту **Java Application**, та натисніть кнопку «**Next**»
2. У наступному вікні введіть ім'я проекту (Project Name). Ім'я проекту треба обирати так, щоб було зрозуміло його призначення (наприклад **First**).
3. Оберіть місце розміщення файлів проекту (**Project Location)** та ім'я головного класу проекту (ім'я може містити ім'я пакету), наприклад, **first.Main**. Залиште відмітку у обох CheckBox'ах. Натисніть кнопку **Finish**.
4. Впишіть код вашої програми у вікно редактора коду NetBeans.
5. Для запуску програми натисніть кнопку "Run" (на ній зображено зелений трикутник).

Примітка 1. Інші інструменти середовища NetBeans будуть розглянуті у наступних лабораторних роботах.

Примітка 2. Для запуску автономної програми на мові Java можна перейти в каталог, де розміщено скомпільований код програми - файли з розширенням class (у нашому випадку каталог classes проекту) та виконати таку команду (у режимі командного рядка):

java <ім'я_класу_без_розширення>

Наприклад, ***`java first.Main`***

Приклад програми, що створена у середовищі NetBeans

```
package first;

public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        new Main().run();
    }

    private double calcSquare(double x) {
        return x*x;
    }

    private void print(double x, double y) {
        System.out.printf("x=%5.2f\n",x);
        System.out.printf("x^2=%5.2f\n",y);
    }

    private void run() {
        double x = 5;
        double y = calcSquare(x);

        print(x,y);
    }

}
```