



Управління освіти
Миколаївської міської ради
Палац творчості учнів

Вивчення мови Java в середовищі NetBeans

Навчально-методичний посібник

Розробила:

Беркунська Наталія Михайлівна
керівник «Комп'ютерної школи»



Миколаїв-2010

Зміст

Вступ.....	4
Введення в розробку графічного інтерфейсу.....	6
Крок 1: Створення проекту.....	6
Крок 2: Створення зовнішнього інтерфейсу.....	7
Крок 3 : Додавання функціональності.....	9
Крок 4: Виконання програми	12
Завдання:	13
Примітивні типи даних та оператори для роботи з ними.....	13
Приклад:	16
Завдання:	17
Дійсні типи та клас Math.....	18
Приклад:	21
Завдання:	22
Булевський (логічний) тип.....	24
Логічні операції	24
Операції відношення	25
Умовний оператор if.....	26
Приклад:	29
Завдання:	31
Оператор вибору switch	31
Оператори інкремента ++ и декремента --	32
Оператор циклу for.....	33
Приклад:	34
Завдання:	36
Оператор циклу while – цикл з передумовою.....	37
Оператор циклу do while – цикл з постумовою.....	38
Приклад:	39
Завдання:	39
Оператори переривання continue, break, return, System.exit	40
Приклад:	41
Додаток. Установка та налаштування засобів програмування мовою Java.	43
Література	46

Вступ

Суспільство висуває усе більші вимоги до рівня освіченості громадян. Однією з них є комп'ютерна грамотність. Сьогодні активне використання комп'ютерних технологій є актуальним в багатьох сферах діяльності. І не випадково, що підвищився об'єм соціального замовлення на навчання дітей цьому виду діяльності. Реалізацію замовлення покладено на шкільні та позашкільні заклади.

Всі учні мають на меті стати вправними користувачами сучасних комп'ютерних і інформаційних засобів, грамотно застосовувати отримані вміння і навички у повсякденному житті та у подальшій діяльності в інформаційному просторі та його «цифровому» середовищі. Але є вихованці, які зацікавлені стати більш ніж користувачі, а саме займатися WEB-дизайном або програмуванням та ін.

Одна з потужних систем об'єктно-орієнтованої технології програмування, що завоювала світове визнання і розвивається швидкими темпами на сучасному етапі, є програмування мовою Java.

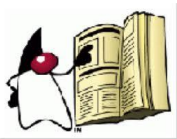
Метою вивчення мови Java є організація систематизованої наскрізної підготовки учнів - студентів – фахівців-професіоналів у галузі сучасних інформаційних і комп'ютерних технологій. Знання ООП на базі мови Java, надасть вихованцям можливість:

- спілкування на міжнародному рівні;
- доступу до різноманітних баз даних;
- високого рівня підготовки до діяльності в інформаційному просторі сучасного суспільства;
- бути конкурентоспроможною людиною на ринку праці;
- безпосередньо працювати на тих підприємствах і фірмах, які потребують кваліфікації грамотного програміста;
- шанс продовжувати навчання з метою отримання міжнародного диплому професійного фахівця у таких сферах, як електрона

комерція, банківська справа, комп'ютерна анімація, розробка проектів, розробка баз даних та ін.

Матеріал даного посібника допоможе керівникам комп'ютерних гуртків в роботі з обдарованими вихованцями (вищий рівень) навчити самостійно складати програмні продукти мовою Java.

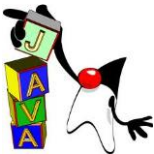
Умовні позначки:



- Теоретичні відомості



- Приклад



- Завдання для опрацювання



Введення в розробку графічного інтерфейсу

У даній темі розглядається створення простого графічного інтерфейсу користувача і додавання до нього нескладної серверної функціональності. Зокрема, буде розглянуто код, що визначає поведінку кнопок і полів у формі **Swing**.

У наданому курсі демонструється розробка проекту графічного інтерфейсу з додаванням ряду кнопок і текстових полів. Текстові поля призначені для отримання вводиться користувачем, і виведення результату роботи програми. Кнопка буде ініціювати роботу функцій, вбудованих в клієнтську частину програми. Створена програма являє собою простий, але повнофункціональний калькулятор.

Для роботи нам потрібно наступне програмне забезпечення.

Програмне забезпечення

Середовище IDE NetBeans

Комплект для розробника мовою Java (JDK)

Крок 1: Створення проекту

Першою дією є створення проекту середовища IDE для програми що розробляється. Дамо проекту ім'я "*NumberAddition*".

1. Виберіть "*File> New Project*". Також можна клацнути значок "*New Project*" на панелі інструментів середовища IDE.
2. В області "*Categories*" оберіть вузол "*Java*". В області "*Projects*" оберіть "*JavaApplication*". Натисніть кнопку "*Next*".
3. Введіть *NumberAddition* в поле "*ProjectName*" та вкажіть путь к місце розташуванню проекту, наприклад, в поточному каталозі.
4. Встановіть прапорець "*UseDedicatedFolderforStoringLibraries*" і вкажіть місце розташування папки бібліотек.
5. Перевірте, що встановлений прапорець "*SetasMainProject*".

6. Видаліть прапорець "*CreateMainClass*", якщо він встановлений.
7. Натисніть кнопку "*Finish*".

Крок 2: Створення зовнішнього інтерфейсу

Для продовження процесу створення інтерфейсу необхідно створити контейнер Java, в котрий будуть поміщені інші необхідні елементи графічного інтерфейсу. У цій дії контейнер буде створений за допомогою елементу *JFrame*. Контейнер буде переміщений в новий пакет, який буде відображатись в вузлі "*SourcePackages*".

Створення контейнера JFrame

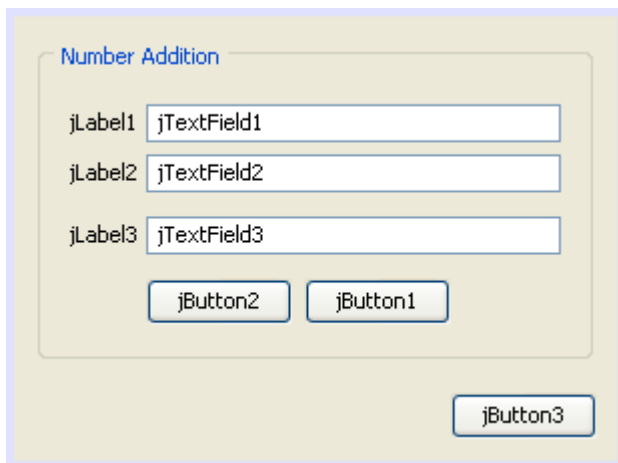
1. У вікні "*Projects*" клацніть правою кнопкою миші вузол "*NumberAddition*" та оберіть "*New>JFrameForm*".
2. Введіть ім'я класу "*NumberAdditionUI*".
3. Виберіть пакет *my.numberaddition*.
4. Натисніть кнопку "*Finish*".

Середовище IDE створює форму *NumberAdditionUI* и клас *NumberAdditionUI* в програмі *NumberAddition* та відкриває форму *NumberAdditionUI* в GUI Builder. Пакет *my.NumberAddition* замінює собою пакет за замовчуванням.

Додавання елементів: створення зовнішнього інтерфейсу

Далі за допомогою вікна "*Palette*" зовнішній інтерфейс програми заповнюється панеллю *JPanel*. Після цього додаються три елемента *JLabel* (текстові написи), три елемента *TextField* (текстові поля) та три елемента *Button* (кнопки).

Після розміщення вказаних вище елементів елемент *JFrame* повинен виглядати так, як показано на малюнку нижче.



Якщо палітра в верхньому правому куті середовища IDE відсутня, оберіть "Windows>Palette".

1. Спочатку оберіть панель *JPanel* на палітрі та перетягніть її в елемент *JFrame*.
2. Панель *JPanel* буде виділена. Перейдіть до вікна "Properties" та натисніть кнопку з трьома крапками (...) поруч з полем "Border" для вибору стилю бордюру.
3. В діалоговому вікні "Border" оберіть "TitledBorder" зі списку та введіть *NumberAddition* в поле "Title". Для збереження змін і закриття діалогового вікна натисніть кнопку "OK".
4. Тепер на екрані повинен відображатись пустий елемент "JFrame" з заголовком "NumberAddition", як показано на малюнку. Згідно малюнка, додайте до нього три підписа *JLabel*, три текстових поля *JTextField* та три кнопки *JButton*.

Перейменування елементів

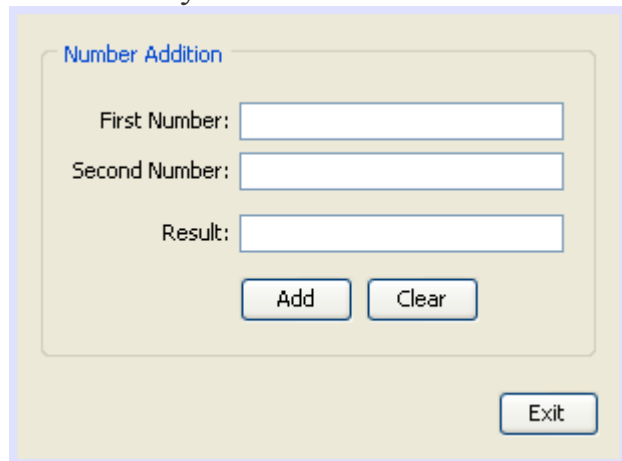
На цьому етапі буде виконано перейменування елементів, що були додані до елементу *JFrame*.

1. Двічі клацніть *jLabel1* та змініть властивість тексту на *First Number*.
2. Двічі клацніть *jLabel2* та змініть текст на *SecondNumber*.
3. Двічі клацніть *jLabel3* та змініть текст на *Result*.
4. Видаліть стандартний текст із *jTextField1*. Для змінення відображаемого тексту спочатку клацніть текстове поле і через деякий час клацніть його вдруге. При цьому може знадобитись відновити попередній розмір поля *jTextField1*. Повторіть цю дію для полів *jTextField2* та *jTextField3*.
5. Змініть відображаємий текст *jButton1* на *Clear*. (Для змінення тексту кнопки клацніть кнопку правою кнопкою миші і оберіть "EditText". В

якості альтернативи можна клацнути кнопку, витримати паузу і клацнути ще раз.)

6. Змініть відображаємий текст *jButton2* на *Add*.
7. Змініть відображаємий текст *jButton3* на *Exit*.

Тепер готовий графічний інтерфейс повинен виглядати так, як показано на малюнку нижче:



Крок 3 : Додавання функціональності

В цій вправі буде додана необхідна функціональність до кнопок "Add", "Clear" та "Exit". Поля *jTextField1* і *jTextField2* будуть використовуватись для вводу значень користувачем, а *jTextField3* – для виводу результату роботи програми. Програма що створюється є найпростіший калькулятор. Отже, приступимо!

Додавання функціональності до кнопки "Exit"

Для того щоб кнопки стали функціональними, кожній з них необхідно присвоїти обробник подій, який відповідатиме за реагування на події. У нашому випадку потрібно ідентифікувати подію натискання кнопки - шляхом клацання мишею або за допомогою клавіатури. Тому буде використовуватись інтерфейс "ActionListener", призначений для обробки подій "ActionEvent".

1. Клацніть правою кнопкою миші кнопку "Exit". У меню що з'явилося оберіть "Events>Action>ActionPerformed". Врахуйте, що меню містить безліч інших подій, на які може реагувати програма! При виборі події "actionPerformed" середовище IDE автоматично додає інтерфейс "ActionListener" до кнопки "Exit" и створює метод-обробник, який буде відповідати за обробку метода "actionPerformed".

2. В середовищі IDE автоматично відкривається вікно "*SourceCode*", де відображається місце вставки дії, яке повинно виконуватись кнопкою при її натисканні (за допомогою миші або клавіатури). Вікно "*SourceCode*" повинно містити наступні строки:

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
//TODO: Add your handling code here:  
}
```

3. Тепер додамо код дії, яку повинна виконувати кнопка "*Exit*". В вищенаведеному коді слід змінити строку "//TODO: Add your handling code here:" текстом "System.exit(0);". Готовий код кнопки "*Exit*" повинен виглядати наступним образом:

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
System.exit(0);  
}
```

Додавання функціональності до кнопки "*Clear*"

1. Клацніть вкладку "*Design*" в верхній частині робочої області для повернення до екрану "*FormDesign*".
2. Клацніть правою кнопкою миші кнопку "*Clear*" (*jButton1*). В меню що з'явилося оберіть "*Events>Action>actionPerformed*".
3. Натискання кнопки "*Clear*" повинно приводити до видалення усього тексту з усіх текстових полів "*JTextField*". Для цього слід додати код, аналогічний наведеному вище. Готовий вихідний код повинен виглядати наступним образом:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt){  
jTextField1.setText("");  
jTextField2.setText("");  
jTextField3.setText("");  
}
```

Цей код видаляє текст з усіх трьох полів *JTextField*, залишив їх порожніми.

Додавання функціональності до кнопки "Add"

Кнопка "Add" повинна виконувати три дії.

1. Спочатку вона приймає дані, що ввів користувач в полях *textField1* і *textField2*, и перетворює їх з типу "String" до типу "Float".
2. Потім має відбутися складання двох чисел.
3. Отримана сума повинна бути перетворена в тип "String" и поміщена в поле *textField3*.

Почнемо!

1. Клацніть вкладку "Design" в верхній частині робочої області для повернення до екрану "Form Design".
2. Клацніть правою кнопкою миші кнопку "Add" (*button2*). В меню що з'явилося виберіть "Events > Action > actionPerformed".
3. Додайте код дій, котрі повинна виконати кнопка "Add".



Готовий вихідний код повинен виглядати наступним образом:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent
evt){
    // Спочатку визначимо дійсні змінні.
    double num1, num2, result;
    // Перетворюємо текст до дійсного типу.
    num1 = Double.parseDouble (textField1.getText());
    num2 = Double.parseDouble(textField2.getText());
    // Тепер ми можемо виконати додавання.
    result = num1+num2;
    // Ми виведемо значення результату у textField3.
    // Одночасно ми перетворюємо
    // значення результату із double до типу string.
    textField3.setText(String.valueOf(result));
}
```

Тепер програма повністю готова, и можна приступити до її складання та виконання.

Крок 4: Виконання програми

Для виконання програми в середовищі IDE виконайте наступні дії:

1. Виберіть "Run > Run Main Project"(F6).
2. При появі вікна з повідомленням про те, що для проекту "NumberAddition" не встановлений головний клас, оберіть в якості головного класу "my.NumberAddition.NumberAdditionUI" та натисніть кнопку "OK".

Для запуску програми поза середовищем IDE виконайте наступні дії:

1. Для складання архіву JAR програми виберіть "Run > Clean and Build Main Project" (Shift-F11).
2. За допомогою провідника по файловій системі або диспетчера файлів перейдіть в каталог NumberAddition/dist.
3. Двічі клацніть файл NumberAddition.jar.

Через декілька секунд програма запуститься.

Примітка:Якщо не вдається запустити програму шляхом подвійного клацання архіву JAR, необхідно змінити налаштування файлових асоціацій JAR в операційній системі.

Можна також запустити програму з командної строки.

Для запуску програми з командної строки виконайте наступні дії:

1. Викличте командну строку або вікно терміналу.
2. В командній строчці змініть поточний каталог на каталог NumberAddition/dist.
3. В командній строчці введіть наступний оператор:

```
java -jar NumberAddition.jar
```



Завдання:

Додайте до програми кнопки, які б обчислювали: різницю, добуток та частку.



Примітивні типи даних та оператори для роботи з ними

Цілі типи, змінні, константи.

Тип	Діапазон значень
byte	-128..127
short	$-2^{15}..2^{15}-1 =$ - 32768.. 32767
char	\u0000..\uFFFF=0.. 65535
int	$-2^{31}..2^{31}-1 =$ - 2.147483648*10 ⁹ .. 2.147483647*10 ⁹
long	$-2^{63}..2^{63}-1 =$ -9.22337203685478*10 ¹⁸ ..9.22337203685478*10 ¹⁸

Для задання в тексті програми чисельних літерних констант типу long, що виходять за межі діапазону чисел типу int, після написання числа необхідно ставити постфікс – букву L. Наприклад, 6000000000000000L. Можна ставити і рядкову l, але її гірше видно, особливо - на роздруківках програми (можна переплутати з одиницею). В інших випадках для всіх цілочисельних типів значення вказується в звичайному вигляді, і воно вважається за тип int - але при присвоєнні число типу int автоматично перетвориться в значення типу long.

Як вже говорилося, оголошення змінних може здійснюватися або в класі, або в методі.

У класі їх можна оголошувати як без явного присвоювання початкового значення, так і з вказівкою цього значення. Якщо початкове значення не вказано, величина ініціалізується нульовим значенням. Якщо оголошується кілька змінних одного типу, після зазначення імені типу дозволяється перераховувати кілька змінних, в тому числі - з присвоюванням їм початкових значень.

У методі всі змінні перед використанням обов'язково треба ініціалізувати - автоматичної ініціалізації для них не відбувається. Це треба робити або при оголошенні, або до спроби використання значення змінної в підпрограмі. Спроба використання в підпрограмі значення неініціалізованої змінної приводить до помилки під час компіляції.

Розглянемо приклади завдання змінних в класі.

```
int i,j,k;  
int j1;  
byte i1,i2=-5;  
short i3=-15600;  
long m1=1,m2,m3=-100;
```

Зауважимо, що після зазначених оголошень змінні i,j,k,j1,i1,m2 мають значення 0. Для i1 це неочевидно - можна подумати, що обидві змінні ініціалізуються значенням -5. Тому краще писати так:

```
byte i1=0, i2=-5;
```

Використання в виразах:

```
i=5;  
j=i*i + 1  
m1=j  
m2=255;  
m1=m1 + m2*2;
```

Константами називаються іменовані комірки пам'яті с незмінним вмістом.

Оголошення констант здійснюється в класі, при цьому перед ім'ям типу константи ставиться комбінація зарезервованих слів **public** і **final**:

```
public final int MAX1=255;  
public final int MILLENIUM=1000;
```

Константами можна користуватися як змінними, доступними тільки з читання.

Основні оператори для роботи з цілочисельними величинами

Всі перераховані нижче оператори діють на дві цілочисельні величини, які називаються операндами. У результаті дії оператора повертається цілочисельний результат. У всіх перерахованих далі прикладах i та j позначають цілочисельні вирази, а v - цілочисельну змінну.

Оператор	Назва	Приклад	Примітка
+	Оператор суми	$i+j$	В разі, коли i та j мають різні типи або типи <code>byte</code> , <code>short</code> або <code>char</code> , діють правила автоматичного перетворення типів.
-	Оператор віднімання	$i-j$	
*	Оператор множення	$i*j$	
/	Оператор ділення	i/j	Результат округлюється до цілого шляхом відкидання дробової частини як для додатних, так і для від'ємних чисел.
%	Остача від ділення	$i \% j$	Повертається остача від цілочисельного ділення
=	Оператор присвоювання	$v=x$	Спочатку обчислюється вираз x , після чого отриманий результат копіюється в комірку v .
++	Оператор інкременту (збільшення на 1)	$v++$	$v++$ еквівалентно $v=v+1$
--	Оператор декременту (зменшення на 1)	$v--$	$v--$ еквівалентно $v=v-1$



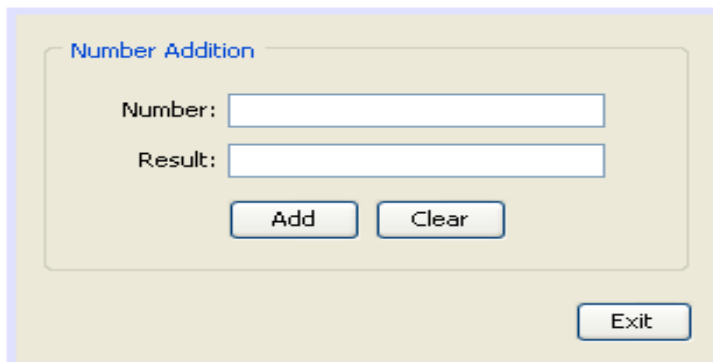
Приклад:

Написати програму підрахунку суми цифр двозначного цілого числа.

1. Створюємо проект.
2. Далі за допомогою вікна "Palette" зовнішній інтерфейс програми заповнюється панеллю JPanel. Після цього додаються два елементи JLabel (текстові написи), два елементи JTextField (текстові поля) та три елементи JButton (кнопки). Після розміщення вказаних вище елементів елемент JFrame повинен виглядати так, як показано на малюнку нижче.



3. Перейменуйте елементи вікна згідно малюнка нижче:



4. Додайте функціональність кнопкам «*Clear*» та «*Exit*» як в попередній темі.
5. Додайте код дій, котрі повинна виконати кнопка "*Add*". Готовий вихідний код повинен виглядати наступним чином:


```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt){
    // Спочатку визначимо цілі змінні.
    int num, result;
    //Перетворюємо текст до цілого типу.
    num = Integer.parseInt(jTextField1.getText());
    // Тепер є можливість виділити розряди одиниць та десятків
    //в окремі змінні
    int num1 = num / 10;
    int num2 = num % 10;
    // Тепер ми можемо виконати додавання.
    result = num1+num2;
    // Ми виведемо значення результату у jTextField3.
    // Одночасно ми перетворюємо
    // значення результату із int до типу string.
    jTextField3.setText(String.valueOf(result));
}
```



Завдання:

1. Написати програму підрахунку суми цифр тризначного числа
2. Написати програму, що за показниками лічильника розраховує оплату за спожиту електрику.



Дійсні типи та клас Math

Тип	Діапазон значень
float	$1.5 \cdot 10^{-45} \dots 3.4 \cdot 10^{38}$
double	$-5 \cdot 10^{-3245} \dots 1.7 \cdot 10^{308}$

Оператор	Назва	Приклад	Примітка
+	Оператор суми	$x + y$	В разі, коли x та y мають різні типи або типи <code>byte</code> , <code>short</code> або <code>char</code> , діють правила автоматичного перетворення типів.
-	Оператор віднімання	$x - y$	
*	Оператор множення	$x * y$	
/	Оператор ділення	x / y	
%	Остача від цілочисельного ділення	$x \% y$	Результат є дійсним. В разі, коли x та y мають різні типи діють правила автоматичного перетворення типів. Повертається остача від цілочисельного ділення. В разі, коли x та y мають різні типи діють правила автоматичного перетворення типів..
=	Оператор присвоювання	$v = x$	Спочатку обчислюється вираз x , після чого отриманий результат копіюється в комірку v .
++	Оператор інкременту (збільшення на 1)	$v++$	$v++$ еквівалентно $v = v + 1$
--	Оператор декременту (зменшення на 1)	$v--$	$v--$ еквівалентно $v = v - 1$

Клас `Math` складається з набору статичних методів, які виробляють найбільш популярні математичні обчислення, і двох констант, що мають особливе значення в математиці, - це число π і основа натурального логарифма. Часто цей клас ще називають класом-утилітою (`Utilityclass`). Так як всі методи класу статичні, немає необхідності створювати екземпляр даного класу, тому він і не має відкритого конструктора. Не можна також і успадковуватися від цього класу, так як він оголошений з модифікатором *final*.

Отож, константи визначені наступним чином:

```
public static final double Math.PI – задає число  $\pi$  ("пи");  
public static final double Math.E – основа натурального логарифма.
```

Модифікатор `static` означає, що це змінна класу; `final` визначає, що в класі-спадкоємці перевизначати це значення неможна.

В таблиці нижче наведена більшість методів класу и дано їх короткий опис.

Значення результату	Ім'я метода та параметри	Опис
...	<code>abs(... a)</code>	абсолютне значення (модуль) для типів <code>double</code> , <code>float</code> , <code>int</code> , <code>long</code>
<code>double</code>	<code>acos(double a)</code>	арккосинус
<code>double</code>	<code>asin(double a)</code>	арксинус
<code>double</code>	<code>atan(double a)</code>	арктангенс
<code>double</code>	<code>ceil(double a)</code>	найменше ціле число, більше за <code>a</code>
<code>double</code>	<code>floor(double a)</code>	ціле число, менше за <code>a</code>
<code>double</code>	<code>IEEEremainder (double a, double b)</code>	остача по стандарту IEEE 754
<code>double</code>	<code>sin(double a)</code>	синус (тут та далі: аргумент повинен бути в радіанах)
<code>double</code>	<code>cos(double a)</code>	косинус
<code>double</code>	<code>tan(double a)</code>	тангенс
<code>double</code>	<code>exp(double a)</code>	<code>e</code> в степені <code>a</code>
<code>double</code>	<code>log(double a)</code>	натуральний логарифм <code>a</code>
...	<code>max(... a, ... b)</code>	більше з двох чисел (для типів <code>double</code> , <code>float</code> , <code>long</code> , <code>int</code>)
...	<code>min(... a, ... b)</code>	менше з двох чисел (для типів <code>double</code> , <code>float</code> , <code>long</code> , <code>int</code>)
<code>double</code>	<code>pow(double a, double b)</code>	<code>a</code> в степені <code>b</code>
<code>double</code>	<code>random()</code>	випадкове число від 0.0 до 1.0
<code>double</code>	<code>rint(double a)</code>	Значення <code>int</code> , ближнє к <code>a</code>
...	<code>round(... a)</code>	значення <code>long</code> для <code>double</code> (<code>int</code> для <code>float</code>), ближнє к <code>a</code>
<code>double</code>	<code>sqrt(double a)</code>	квадратний корінь числа <code>a</code>
<code>double</code>	<code>toDegrees(double a)</code>	перетворення із радіан в градуси
<code>double</code>	<code>toRadians(double a)</code>	перетворення із градусів в радіани

Розгляньте приклади використання функції класу Math

```
//Знаходження найбільшого з чисел a та b  
double c= Math.max(a,b);  
//Піднесення значення змінної a до 5 степеня  
double c= Math.pow(a,5);
```

Та починаючи з jdk 1.5, для цих цілей можна скористатись статичним імпортом.

```
import static java.lang.Math.*;
```

І тепер можна вказувати математичні функції без кваліфікатора Math. Але на початковому етапі імпортом ми користуватися не будемо.

Пріоритет операторів

При обчисленні виразів важливий пріоритет операторів. Для операторів складання, віднімання, множення і ділення він "природний": множення і ділення володіють однаковим найвищим пріоритетом, а додавання і віднімання - однаковим пріоритетом, який нижче.

Таким чином, наприклад,

$a*b/c+d$

це те саме, що

$((a*b)/c)+d$

Круглі дужки дозволяють групувати елементи виразів, при цьому вираз в дужках обчислюється до того, як бере участь в обчисленні решти вираження. Тобто дужки забезпечують більший пріоритет, ніж всі інші оператори. Тому $(a + b) * c$ буде обчислюватися так: спочатку обчислити суму $a + b$, після чого отриманий результат буде помножений на значення c .

Крім перерахованих в Java є велика кількість інших правил, що визначають пріоритети різних операторів. але програму слід писати так, щоб всі

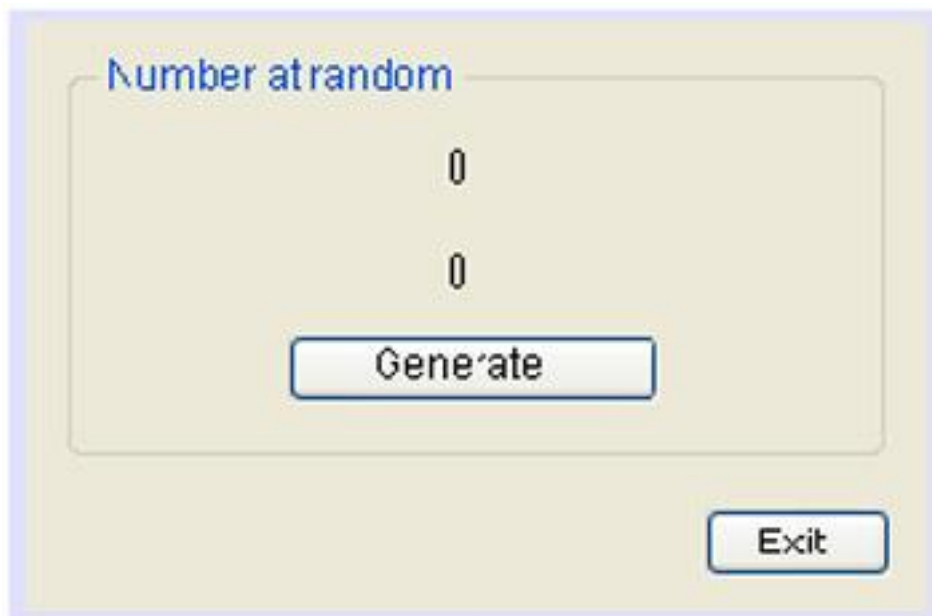
послідовності дій були очевидні і не могли викликати складнощів у розумінні тексту програми і привести до логічної помилки. Тому слід розставляти дужки навіть у тих випадках, коли вони теоретично не потрібні, але роблять очевидною послідовність дій. Відзначимо, такі дії часто допомагають заодно вирішити набагато більш складні проблеми, пов'язані з арифметичним переповненням.



Приклад:

Написати програму, що генерує випадкове число, та підносить його до другого степеня.

1. Створюємо проект.
2. Далі за допомогою вікна "Palette" зовнішній інтерфейс програми заповнюється панеллю JPanel. Після цього додаються два елемента JLabel (текстові написи) та два елемента JButton (кнопки). Після розміщення вказаних вище елементів перейменуйте елементи вікна згідно малюнка нижче:



3. Додайте функціональність кнопці «Exit» як в першій темі.
4. Додайте код дій, котрі повинна виконати кнопка "Generate".

Готовий вихідний код повинен виглядати наступним чином:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt){
    // Спочатку визначимо дійсні змінні.
    double num, result;
    //Генеруємо випадкове число та поміщаємо його в змінну num.
    num = Math.random();
    //Обчислюємо num в квадраті
    result =Math.pow(num,2);
    //Виводимо значення змінної num у jLabel1,
    // а значення змінної result у jLabel2. Одночасно ми перетворюємо
    // значення результату типу string.
    jLabel1.setText(String.valueOf(num));
    jLabel2.setText(String.valueOf(result));
}
```



Завдання:

Скласти програму обчислення значень змінних, вказаних в таблиці за обраним варіантом, по наданим розрахунковим формулам и наборам вхідних даних. На екрані повинні бути підписані поля вхідних, та результати обчислень супроводжуються найменуваннями виведених змінних.

Варіант задання	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	$c = x^{y/x} - \sqrt[3]{y/x} ; f = (y-x) \frac{y-z/(y-x)}{1+(y-x)^2}$	$x=1.825$ $y=18.225$ $z=-3.298$
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	$y = e^{-bt} \sin(at + b) - \sqrt{bt + a}; s = b \sin(at^2 \cos 2t) - 1$	$a=-0.5$ $b=1.7$ $t=0.44$
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	$s = x^3 \operatorname{tg}^2(x + b)^2 + \frac{a}{\sqrt{x + b}}; Q = \frac{bx^2 - a}{e^{ax} - 1}$	$a=16.5$ $b=3.4$ $x=0.61$
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	$f = \sqrt[3]{2 \operatorname{tg} t + c \sin t }; z = 2 \cos(bt \sin t) + c$	$c=-1$ $t=1.2$ $b=0.7$
10	$y = b \operatorname{tg}^2 x - \frac{a}{\sin^2(x/a)}; d = a e^{-\sqrt{a}} \cos(bx/a)$	$a=3.2$ $b=17.5$ $x=-4.8$
11	$f = \ln(a + x^2) + \sin^2(x/b); z = e^{-x} \frac{x + \sqrt{x+a}}{x - \sqrt{ x-b }}$	$a=10.2$ $b=9.2$ $x=2.2$
12	$y = \frac{a^{2x} + b^{-x} \cos(a + b)x}{x + 1}; R = \sqrt{x^2 + b - b^2 \sin^3(x + a) / x}$	$a=0.3$ $b=0.9$ $x=0.61$



Булевський (логічний) тип

Величини типу `boolean` приймають значення `true` або `false`.

Об'ява булевських змінних:

```
boolean a;  
boolean b;
```

Використання в виразах при присвоюванні:

```
a=true;  
b=a;
```

Булевські величини зазвичай використовуються в логічних операторах та в операціях відношення.

Логічні операції

Оператор	Назва	Приклад
<code>&&</code>	логічне "І" (and)	<code>a && b</code>
<code> </code>	логічне "АБО" (or)	<code>a b</code>
<code>^</code>	логічне "виключне АБО" (xor)	<code>a ^ b</code>
<code>!</code>	логічне "НЕ" (not)	<code>! a</code>

Значенням логічного виразу є `true` та `false`. Наприклад, якщо `a=true`, `b=true`, то `a && b` має значення `true`. А при `a=false` або `b=false` вираз `a && b` приймає значення `false`.

Виконання булевських операторів відбувається на апаратному рівні, а значить, дуже швидко. Реально процесор оперує числами 0 і 1, хоча в Java це здійснюється прихованим від програміста чином.

Для роботи з логічними виразами часто застосовують так звані таблиці істинності. У них замість логічної одиниці (`true`) пишуть 1, а замість логічного нуля (`false`) пишуть 0. У наведеній нижче таблиці вказані значення логічних виразів при всіх можливих комбінаціях значень `a` і `b`.

Вираз	Значення			
a	0	0	1	1
b	0	1	0	1
a && b	0	0	0	1
a b	0	1	1	1
a ^ b	0	1	1	0
! a	1	1	0	0

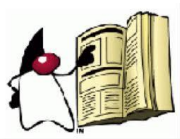
Логічні вирази в Java обчислюються відповідно за так званим укороченим оцінюванням: з наведеної вище таблиці видно, що якщо a має значення false, то значення оператора a && b дорівнюватиме false незалежно від значення b. Тому якщо b є Булевським виразом, його можна не обчислювати. Аналогічно, якщо a має значення true, то значення оператора a || b дорівнюватиме true незалежно від значення b.

Операції відношення

Це оператори порівняння та приналежності. Вони мають результат типу boolean.

Оператори порівняння можна застосувати до будь-яких величин a і b одного типу, а також до довільних числових величин a і b, які не обов'язково мають один тип.

Оператор	Назва	Приклад
==	равно	a==b
!=	не равно	a!=b
>	більше	a>b
<	менше	a=	більше чи равно	a>=b
<=	менше чи равно	a<=b



Умовний оператор if

У умовного оператора if є дві форми: if та if- else.

Англійською if означає “якщо”, else - “ у противному випадку ”. Таким чином, ці оператори можуть бути переведені як “якщо...то...” та “якщо...то...у противному випадку...”.

Перша форма:

```
if(умова)
    оператор1;
```

Якщо умова дорівнює true, виконується оператор1. Якщо ж умова==false, в операторі не виконується ніяких дій.

Друга форма:

```
if(умова)
    оператор1;
else
    оператор2;
```

В такому варіанті оператора if якщо умова==false, то виконується оператор2.

Зверніть особливу увагу на форматування тексту. Не розташовуйте усі частини оператора if на одній строчці – це характерно для новачків!

Приклад:

```
if(a<b)
    a=a+1;
else if(a==b)
    a=a+1;
else{
    a=a+1;
    b=b+1;
};
```

Треба відмітити, що в операторі if в області виконання, що йде після умови, а також в області else, повинен стояти тільки один оператор, а не послідовність операторів. Тому запис оператора в вигляді

```
if(умова)
    оператор1;
    оператор2;
else
    оператор3;
```

недопустима. В таких випадках застосовують складений оператор, обмежений фігурними дужками{}. Між ними може стояти довільне число операторів:

```
if(умова){
    оператор1;
    оператор2;
}
else
    оператор3;
```

Якщо ж ми напишемо

```
if(умова)
    оператор1;
else
    оператор2;
    оператор3;
```

- ніякої діагностики помилки компілятор не видасть! Оператор3 в цьому випадку ніякого відношення до умови else мати не буде – подібне форматування тексту буде підштовхувати до логічної помилки. При наступному форматуванні тексту програми, такому ж як попередній при компіляції, вже більш вочевидь, що оператор3 не стосується до частини else:

```
if(умова)
    оператор1;
else
    оператор2;
оператор3;
```

Для того, щоб оператор3 відносився до частини else, слід використовувати складений оператор:

```
if(умова)
    оператор1;
else{
    оператор2;
    оператор3;
};
```

В випадку послідовності операторів типу:

```
if(умова1)if(умова2)оператор1 else оператор2;
```

наявний else стосується останнього if, тому краще відформатувати текст так:

```
if(умова1)
    if(умова2)
        оператор1;
    else
        оператор2;
```

Таким чином, якщо писати відповідні if та else друг під другом, логіка роботи програми стає вочевидь.

Типовою є ситуація з забутим об'єднанням послідовності операторів в складений за допомогою фігурних дужок.

Наприклад, пишуть

```
if(условие)
    оператор1;
    оператор2;
```

замість

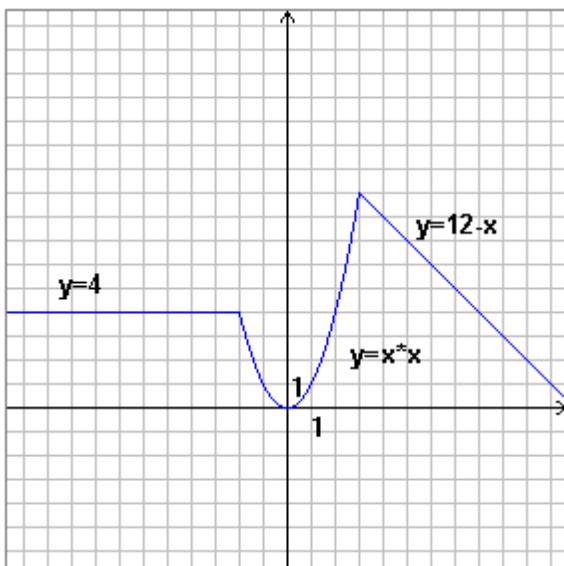
```
if(условие)
{оператор1;
  оператор2;
};
```

До того ж таку помилку час от часу допускають навіть досвідчені програмісти.



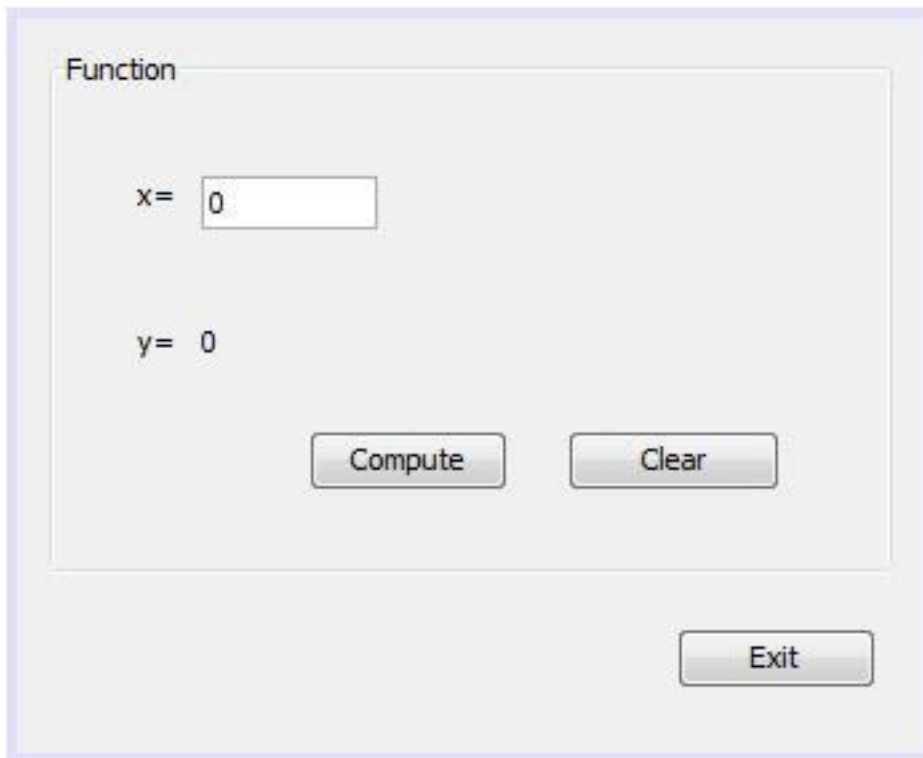
Приклад:

Написати програму обчислення значення функції згідно графіка.



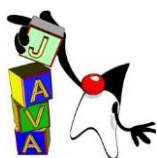
1. Створюємо проект.
2. Нам знадобляться два елементи JLabel (текстові написи), один елемент JTextField (текстове поле) та три елемента JButton (кнопки).

Після розміщення вказаних вище елементів, та їх перейменування елемент JFrame повинен виглядати так, як показано на малюнку нижче.



3. Додайте функціональність кнопкам «Exit» та «Clear» як в першій темі.
 4. Додайте код дій, котрі повинна виконати кнопка "Compute".
- Готовий вихідний код повинен виглядати наступним чином:

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    //Перетворюємо текст до дійсного типу.  
    double x=Double.valueOf(jTextField1.getText());  
    double y;  
    if (x<-2) { y=4;}  
    else {  
        if (x<3) {y=x*x;}  
        else {  
            y=12-x;  
        }  
    }  
    // Виводимо значення результату у jLabel2, одночасно перетворюємо  
    // значення результату із double до типу string.  
    jLabel2.setText("y="+y);  
}
```



Завдання:

1. Скласти програму, що знаходить корені квадратного рівняння $ax^2+bx+c=0$, ($a \neq 0$):
2. Дано три дійсних числа. Обрати з них ті, що лежать в проміжку(1,3).
3. Дано три дійсних числа. Піднести в квадрат ті з них, значення яких від'ємні.
4. (*) Дано дійсні додатні числа x, y, z . З'ясувати, чи існує трикутник с довжинами сторін x, y, z .
5. Написати гру «Вгадай число від 1 до 6».



Оператор вибору switch

Цей оператор є аналогом if для декількох умов вибору.

Синтаксис оператора наступний:

```
switch(вираз){  
    case значення1: оператори1;  
    .....  
    case значенняN: оператори N;  
    default: оператори;  
}
```

Правда, вкрай незручно, що не можна ні вказувати діапазон значень, ні перераховувати через кому значення, яким відповідають однакові оператори.

Тип вираження повинен бути яким-небудь з цілих типів. Зокрема, неприпустимі дійсні типи.

Працює оператор наступним чином: спочатку обчислюється вираз. Потім розраховане значення порівнюється зі значеннями варіантів, які повинні бути визначені ще на етапі компіляції програми. Якщо знайдено варіант, якому задовольняє значення виразу, виконується відповідна цьому варіанту послідовність операторів, після чого **НЕ ВІДБУВАЄТЬСЯ** виходу з оператора

case, що було б природно.- Для такого виходу треба поставити оператор **break**. Ця неприємна особливість Java успадкована від мови C.

Частина з *default* є необов'язковою і виконується, якщо жоден варіант не знайдений.

Приклад:

```
switch(i/j){  
  case 1:  
    i=0;  
    break;  
  case 2:  
    i=2;  
    break;  
  case 10:  
    i=3;  
    j=j/10;  
    break;  
  default:  
    i=4;  
};
```

У оператора *switch* є дві особливості:

- Можна писати довільне число операторів для кожного варіанту *case*, що вельми зручно, але зовсім випадає з логіки операторів мови Java.
- Вихід із виконання послідовності операторів здійснюється за допомогою оператора *break*. Якщо він відсутній, відбувається "провалювання" в блок операторів, відповідних наступному варіанту за тим, з яким співпало значення виразу.



Оператори інкремента ++ и декремента --

Оператор “++” називається інкрементним (“збільшення”), а “--” декрементним (“зменшення”). У цих операторів є дві форми, постфіксна (найбільш поширена, коли оператор ставиться після операнда) і префіксна (використовується дуже рідко, в ній оператор ставиться перед операндом).

Для будь-якої чисельної величини x вираз $x++$ або $++x$ означає збільшення x на 1, а вираз $x--$ або $--x$ означає зменшення x на 1.

Різниця двох форм пов'язана з тим, коли відбувається зміна величини - після обчислення виразу, в якому використовується оператор, для постфіксної форми, або до цього обчислення - для префіксної.

Наприклад, присвоювання $j=i++$ та $j=++i$ дадуть різні результати. Якщо спочатку $i = 0$, то перше присвоювання дасть 0, так як i збільшиться на 1 після виконання присвоювання. А друге дасть 1, так як спочатку виконається інкремент, і лише потім буде обчислюватися вираз i виконуватися присвоювання. При цьому в обох випадках після виконання присвоювання i стане дорівнювати 1.



Оператор циклу for

```
for(блок ініціалізації; умова виконання тіла циклу;  
    блок зміни лічильників)  
    оператор;
```

У **блоці ініціалізації** через кому перераховуються оператори завдання локальних змінних, область існування яких обмежується оператором `for`. Також можуть бути присвоєні значення змінним, заданим поза циклом. Але ініціалізація може відбуватися тільки для змінних одного типу.

У **блоці умови** продовження циклу перевіряється виконання умови, і якщо воно виконується, йде виконання тіла циклу, в якості якого виступає оператор. Якщо ж не виконується - цикл припиняється, і йде перехід до оператора програми, наступного за оператором `for`.

Після кожного виконання тіла циклу (чергового кроку циклу) виконуються оператори **блоку зміни лічильників**. Вони повинні розділятися комами.

Приклад:

```
for(int i=1,j=5; i+j<100; i++,j=i+2*j){  
...  
};
```

Кожен з блоків оператора for є необов'язковим, але при цьому розділювальні ";" потрібно писати.

Найбільш уживане використання оператора for - для перебору значень деякої змінної, що збільшуються або зменшуються на 1, і виконання послідовності операторів, які використовують ці значення. Змінна називається лічильником циклу, а послідовності операторів - тілом циклу .



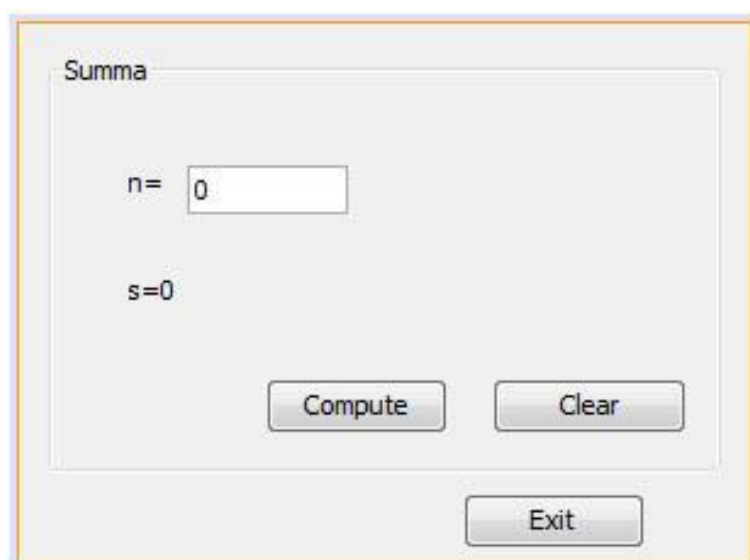
Приклад:

Обчислити суму чисел що йдуть послідовно.

Напишемо цикл, в якому йде підсумовування всіх чисел від 1 до x. Результат будемо зберігати у змінній result.

1. Створюємо проект.
2. Нам знадобляться два елементи JLabel (текстові написи), один елемент JTextField (текстове поле) та три елемента JButton (кнопки).

Після розміщення вказаних вище елементів, та їх перейменування елемент JFrame повинен виглядати так, як показано на малюнку нижче.



3. Додайте функціональність кнопкам «Exit» та «Clear» як в першій темі.
4. Додайте код дій, котрі повинна виконати кнопка "Compute".

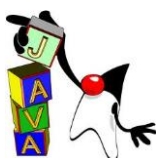
Готовий вихідний код повинен виглядати наступним чином:

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    //Перетворюємо текст до цілого типу.  
    int n=Integer.valueOf(jTextField1.getText());  
    int result=0;  
    for(int i=1; i<=n; i++){  
        result=result+i;  
    };  
    // Виводимо значення змінної result у jLabel2  
    jLabel2.setText("S="+result);  
}
```

Цикл (повторне виконання одних і тих же дій) виконується таким чином:

- ✓ До початку циклу створюється змінна `result`, в якій ми будемо зберігати результат. Одночасно виконується ініціалізація - присвоюється початкове значення 0.
- ✓ Починається цикл. Спочатку виконується блок ініціалізації - лічильнику циклу і присвоюється значення 1. Блок ініціалізації виконується тільки один раз на самому початку циклу.
- ✓ Починається перший крок циклу. Перевіряється умова виконання циклу. Значення `i` порівнюється з `n`.
- ✓ Оскільки порівняння `1 <= n` повертає `true`, виконується тіло циклу. У змінній `result` зберігається 0, а значення `i` дорівнює 1, тому присвоювання `result = result+i` еквівалентно `result = 1`. Таким чином, після першого кроку циклу у змінній `result` буде зберігатися значення 1.
- ✓ Після виконання тіла циклу виконується секція зміни лічильника циклу, тобто оператор `i`, збільшує `i` на 1. Значення `i` стає рівним 2.

- ✓ Починається другий крок циклу. Перевіряється умова виконання тіла циклу. Оскільки порівняння $2 \leq n$ повертає true, йде чергове виконання тіла циклу, а потім - збільшення лічильника циклу.
- ✓ Кроки циклу продовжуються до тих пір, поки лічильник циклу не стане рівним $n+1$. У цьому випадку умова виконання тіла циклу $n+1 \leq n$ повертає false, і відбувається вихід з циклу. Останнє присвоювання $result = result+i$, проведене в циклі, це $result = result+n$.



Завдання:

<p>1. Задано натуральне число n. Обчислити:</p> $\left(1 + \frac{1}{1^2}\right) \left(1 + \frac{1}{2^2}\right) \dots \left(1 + \frac{1}{n^2}\right)$	<p>2. Задано натуральне число n. Обчислити:</p> $\frac{1}{\sin 1} + \frac{1}{\sin 2} + \dots + \frac{1}{\sin n}$
<p>3. Задано дійсне a, натуральне число n. Обчислити:</p> $a(a+1) \dots (a+n-1)$	<p>4. Задано натуральное n, действительное x. Обчислити:</p> $\sin x + \sin^2 x + \dots + \sin^n x$
<p>5. Задано натуральне n, дійсне x. Обчислити:</p> $\sin x + \sin x^2 + \dots + \sin x^n$	<p>6. Задано дійсне a, натуральне число n. Обчислити:</p> $\frac{1}{a} + \frac{1}{a^2} + \frac{1}{a^4} + \dots + \frac{1}{a^{2^n}}$



Оператор циклу **while** – цикл з передумовою

```
while(умова)
```

```
    оператор;
```

Поки умова зберігає значення true - в циклі виконується оператор, інакше - дія циклу припиняється. Якщо умова з самого початку false, цикл відразу припиняється, і тіло циклу не виконається жодного разу.

Цикл while зазвичай застосовують замість циклу for в тому випадку, якщо умови продовження досить складні. На відміну від циклу for в цьому випадку немає формально заданого лічильника циклу, і не проводиться його автоматичної зміни. За це відповідає програміст. Хоча цілком можливо використання як циклу for замість while, так і навпаки.

Приклад:

```
i=1;  
x=0;  
while(i<=n){  
    x+=i;//еквівалентно x=x+i;  
    i*=2;//еквівалентно i=2*i;  
};
```



Оператор циклу **do while** – цикл з постумовою

Іноді виникає необхідність забезпечити циклічне виконання оператора якнайменше один раз. З цією метою використовується конструкція **do ... while**, синтаксис якої може бути описаний таким чином:

```
do
    оператор;
while (умова);
```

В цьому випадку перевірка істинності умови виконується після виконання тіла циклу. Цикл повторюється до тих пір, поки в результаті перевірки виразу не буде отримане значення **false**. В якості тіла циклу **do ... while** найчастіше використовується блок операторів.

Приклад:

```
int i=0;
double x=1;
do{
    i++; // i=i+1;
    x*=i; // x=x*i;
}
while(i<n);
```

При необхідності організувати нескінченний цикл (з виходом зсередини тіла циклу за допомогою оператора переривання) часто використовують наступний варіант:

```
do{
    ...
}
while(false);
```

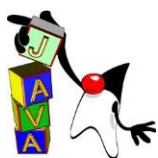
У операторі while дуже часто роблять помилки, що призводять до нестійкості алгоритмів через порівняння чисел з плаваючою точкою на нерівність. Порівнювати їх на рівність у переважній більшості випадків некоректно через помилки подання таких чисел в комп'ютері. Тому треба порівнювати їх різницю з дуже маленьким числом.



Приклад:

Обчислити значення функції $y=14x^3+5x-12$ для значень x , що змінюються від -3 до -1, з кроком 0.1.

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    //Ініціюємо змінні x та y.  
    double x=-3;  
    double y;  
    //Ініціюємо строкову змінну s та задамо їй значення пустий рядок  
    String s="";  
    While (x-(-1)<=0.000001){  
        y=14*x*x*x+5*x-12;  
        x=x+0.1;  
        s=s+" "+y  
    };  
    // Виводимо значення змінної s у jLabel2  
    jLabel2.setText("S="+s);  
}
```



Завдання:

1. Обчислити значення функції $y=3x^2-12x+9$ для значень x , що змінюються від -2 до 0, з кроком 0.1.
2. Обчислити значення функції $y=\sin x$ для значень x , що змінюються від 0 до π , з кроком $\pi/6$.



Оператори переривання **continue**, **break**, **return**, **System.exit**

1. **Мітки.** Оператори програми можуть бути позначені мітками (labels).

Мітка являє собою змістовне ім'я, що дозволяє посилатися на відповідний оператор:

```
Мітка: Оператор
```

Звертатися до мітки дозволено тільки за допомогою команд **break** та **continue** (вони розглядатимуться далі).

2. **break.** Інструкція **break** застосовується для завершення виконання коду будь-якого блоку. Існують дві форми інструкції – безіменна:

```
break;
```

та іменована:

```
break мітка;
```

Безіменна команда **break** перериває виконання коду конструкцій **switch**, **for**, **while** або **do** і може використовуватися лише всередині цих конструкцій. Команда **break** у іменованій формі може перервати виконання будь-якої інструкції, що помічена відповідною міткою.

Команда **break** найчастіше використовується для примусового виходу з тіла циклу. Для виходу із вкладеного циклу чи блоку, достатньо позначити міткою зовнішній блок і вказати її в інструкції **break** як показано в наступному прикладі:



Приклад:

Використання поміченого **break**

```
private float[][] matrix;
public boolean workOnFlag(float flag) {
    int y, x;
    boolean found = false;
    search:
    for (y = 0; y < matrix.length; y++) {
        for (x = 0; x < matrix[y].length; x++) {
            if (matrix[y][x] == flag) {
                found = true;
                break search;
            }
        }
    }
    if (!found) {
        return false;
    }
    // А тут знайдено значення matrix[y][x]
    // деяким чином обробляється
    return true;
}
```

Відмітимо, що іменований оператор **break** – це зовсім не те ж саме, що й сумнозвісна команда **goto**. Оператор **goto** дозволяє ”стрибати” по коду без жодних обмежень, переплутуючи порядок обчислень і збиваючи читача з глузду. Команди же **break** і **continue**, що посилаються на мітку, дозволяють лише акуратно залишити відповідний блок і забезпечити його повторення, при цьому потік обчислень залишається цілком очевидним.

3. **continue**. Команда **continue** застосовується лише у контексті циклічних конструкцій і передає управління на кінець тіла циклу. В ситуації з **while** і **do** це призводить до виконання перевірки умови циклу, а при використанні в тілі **for** інструкція **continue** провокує передавання управління секції змін значень змінних циклу.

Як і **break**, команда **continue** дозволяє використання в двох формах – без імені: **continue**; і іменованій: **continue** мітка. Команда **continue** у формі без імені мітки передає управління в кінець поточного циклу, а іменована – в кінець циклу, позначеного відповідною міткою. Мітка повинна відноситися до циклічного виразу.

4. **return**. Оператор **return** завершує виконання методу і передає управління до коду-ініціатору. Якщо метод не повертає значень, оператор виглядає просто: **return**. Якщо ж в оголошенні методу вказано тип параметра, що повертається, до складу команди **return** повинен бути включеним вираз, який може бути присвоєний змінній оголошеного типу. Наприклад, якщо метод повертає значення типу **double**, в

операторі **return** дозволяється використовувати вирази, що відносяться до типів **double**, **float** або будь-якого цілих типів.

Додаток. Установка та налаштування засобів програмування мовою Java.

Засоби програмування мовою Java традиційно складаються з двох великих частин:

- Компілятор, середовище виконання, утиліти командного рядка, файли електронної документації (Java Software Development Kit – Java SDK)
- Середовища розробки

Для установки більшості з сучасних засобів розробки треба спочатку встановити Java SDK, після цього треба встановити середовище розробки.

1. Для установки Java SDK v 6.0 в операційних системах родини Windows запустіть на виконання інсталяційний пакет **jdk-6u3-windows-i586-p.exe** (його можна отримати з файлового архіву університету, або завантажити з сайту розробника – Sun Microsystems мережі Internet: <http://java.sun.com>) Для установки Java SDK в операційних системах родини Linux треба скористатися інсталяційним пакетом **jdk-6u3-linux-i586.bin** (отримати його можна там же)
2. Для комфортної роботи з Java SDK розпакуйте файли електронної документації з архіву **jdk-1_6_0-doc.zip** його можна також отримати з сайту розробника, чи з файлового архіву університету (Бажано розпаковувати в той самий каталог, котрий було створено при інсталяції компілятора, наприклад, якщо компілятор встановлено в каталог *C:\Program Files\Java\jdk1.6.0*, то розпаковувати архів треба в нього, при цьому там буде створено підкаталог *docs*)
3. Після того треба встановити одне з середовищ програмування. Для Windows можна використовувати середовище NetBeans (про роботу з ним див. початок посібника) Також можна використовувати інші середовища

BlueJ, або JBuilder. В процесі інсталяції, якщо з'явиться запит, вкажіть папку в якій встановлено компілятор і система довідки.

Після інсталяції пакета Java SDK треба зробити ще один крок: додати ім'я каталогу `jdk/bin` до списку каталогів, де операційна система може знайти виконувані файли.

На платформі Windows 9x додайте рядок, що наведений нижче в кінець файлу `AUTOEXEC.BAT`:

```
SET PATH="C:\Program Files\Java\jdk1.6.0\bin";%PATH%
```

На платформі Windows NT/200/XP відкрийте панель управління, оберіть піктограму "Система" там у вкладці "Дополнительно" натисніть кнопку "Переменные среды". Оберіть `PATH` у вікні User Variables (Переменные пользователя) та додайте у початок цього рядка ім'я каталогу `jdk/bin`, наприклад так:

```
"C:\Program Files\Java\jdk1.6.0\bin";інше-те що було раніше
```

Збережіть свої установки.

В операційних системах Linux треба виконати аналогічні дії в залежності від встановленої оболонки.

Для перевірки правильності виконаних дій зробіть таке:

Відкрийте вікно оболонки (WinXP – Пуск-Выполнить-cmd.exe), у ньому введіть такий рядок:

```
java -version
```

та натисніть <ENTER> На екрані повинно з'явитися таке:

```
java version "1.6.0_03"
```

```
Java(TM) SE Runtime Environment (build 1.6.0_03-b05)
```

```
Java HotSpot(TM) Client VM (build 1.6.0_03-b05, mixed mode)
```

якщо замість цього повідомлення з'явився рядок типу "java:command not found" треба повернутися і ще раз перевірити, чи правильно виконано інсталяцію.

Література

1. Арнольд К., Гослинг Дж., Холмс Д. Язык программирования Java. 3-е изд. – М.: Издательский дом "Вильямс", 2002 – 624 с.
2. Бадд Т. Объектно-ориентированное программирование в действии. – СПб.: Питер, 1997. – 464 с.
3. Бишоп Д. Эффективная работа: Java 2. –СПб.: Питер; К.: Издательская группа BHV, 2002. – 592 с.
4. Монахов В.В., Язык программирования Java и среда NetBeans. 2-е изд.- СПб.: БХВ-Петербург, 2009 – 710 с.
5. Синтес А. Освой самостоятельно объектно-ориентированное программирование за 21 день. – М.: Издательский дом "Вильямс", 2002. - 672 с.
6. Хабибуллин И. Самоучитель Java 2. СПб.: BHV, 2007. – 720 с.
7. <http://netbeans.org/> – матеріали офіційного сайту середовища NetBeans
8. <http://www.oracle.com/technetwork/java> – матеріали сайту компанії Oracle, що стосуються мови програмування Java.