ЛАБОРАТОРНА РОБОТА 1

Тема: Знайомство з середовищем IntelliJ IDEA та написання першої простої програми.

Мета: ознайомитися з основами технології Java. Навчитися користуватися засобами JDK та IDE для створення java-програм.

Теоретичні відомості

На сьогоднішній момент мова Java є однією з найбільш поширених і популярних мов програмування. Перша версія мови з'явилася ще в 1996 році в надрах компанії Sun Microsystems, згодом поглиненої компанією Oracle. Java замислювалася як універсальна мова програмування. Поточною версією є Java 12, яка вийшла в березні 2019 року і перетворилася

з просто універсальної мови в цілу платформу і екосистему і використовується для вирішення цілого ряду завдань: від створення десктопних додатків до написання великих веб-порталів і сервісів. Крім того, мова Java активно застосовується для створення програмного забезпечення для цілого ряду пристроїв: звичайних ПК, планшетів, смартфонів і мобільних телефонів та навіть побутової техніки. Досить згадати популярність мобільної ОС Android, більшість програм для якої пишуться саме на Java.

Мінімум необхідного для програмування на Java — це JDK (Java Developer Kit - комплект розробника Java) та звичайний текстовий редактор. Щоправда бажано, щоб редактор хоча б здійснював підсвітку синтаксису програми, здійснював компіляцію та запуск програми одним натисненням кнопки. Для цієї мети існують відносно прості безкоштовні текстові редактори, які вже інтегровані в різноманітні файлові менеджери (FAR, Total Commander, Frigate і т.д.). Такі редактори можуть бути корисні, якщо ресурси комп'ютера обмежені.

Найкращим же варіантом є спеціалізовані середовища розробки (IDE — інтегроване середовище розробки) такі як Eclipse, NetBeans або IDEA. При розробці великих проектів інтегровані середовища розробки значно полегшують роботу програмістові і скорочують час на розробку.

Для роботи програм на мові Java на цільовій машині повинна бути встановлена JRE (Java Runtime Environment). JRE представляє мінімальну реалізацію віртуальної машини, а також бібліотеку класів. Тому, якщо ми хочемо запускати програми, то нам треба встановити JRE. Для кожної конкретної платформи є своя версія JRE.

Однак, так як ми збираємося не тільки запускати програми, а й розробляти їх, нам буде потрібно спеціальний комплект для розробки JDK (Java Development Kit). JDK вже містить JRE, а також включає ряд додаткових програм і утиліт, зокрема компілятор Java.

Завантажити та встановити відповідну версію JDK можна з офіційного сайту Oracle:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Отже, з вище згаданої адреси завантажимо програму установки JDK для останньої версії Java.

Потім нас перекине на сторінку, де треба вибрати версію ОС - Windows, MacOS або Linux

Для кожної ОС є два варіанти завантаження: у вигляді інсталятора, або у вигляді архіву, який не треба встановлювати. Наприклад, якщо ваша ОС - Windows, то завантажуємо файл jdk_12_windows-x64_bin.exe, який представляє програму установки.

Хід роботи

1. Щоб перетворити код програми в додаток необхідно використати компілятор. Після

установки JDK всі файли за замовчуванням поміщаються в каталог ***C: \ Program Files \ Java \ jdk- [номер_ версії]*** (при використанні ОС Windows). У нашому випадку це каталог ***C: \ Program Files \ Java \ jdk-12***. Якщо ми відкриємо в ньому підкаталог ***bin***, то ми зможемо побачити ряд утиліт. Нас насамперед цікавить
утиліта компілятора ***javac***. Щоб скомпілювати клас програми, нам треба передати її код компілятору. Також слід зазначити іншу утиліту з цієї папки - ***java.exe***, яка дозволяє запускати скомпільовану програму.

2. Створимо на жорсткому диску який-небудь каталог, в якому будуть розташовуватися файли з вихідним кодом на мові Java. Припустимо це буде каталог ***C: / Java***. Потім створимо в цьому каталозі текстовий файл, який перейменуємо в ***Program.java***. Відкриємо цей файл в будь-якому текстовому редакторі і наберемо в ньому наступну програму:

```
public class Program {
     public static void main (String args[]){
        System.out.println("Hello Java!");
     }
  }
```

3. Тепер скомпілюємо написану програму. Відкриємо командний рядок (в Windows) або термінал в Linux / MacOS і введемо там відповідні команди.

Насамперед перейдемо в каталог, де лежить наш файл з програмою за допомогою команди: ***cd C:\Java***

4. Потім скомпілюємо програму за допомогою команди ***C:\Java>"C:\Program Files\Java\jdk-12\bin\javac" Program.java***. Зверніть увагу, що весь шлях до компілятора **javac** береться в лапки, а потім через пробіл йде назва нашого файлу, який містить клас програми.

5. Програма компілюється в байт-код, і в каталозі ***C: \ Java*** можна буде знайти новий файл ***Program.class***. Це і буде файл з байт-кодом програми. Тепер нам треба його запустити за допомогою утиліти java: ***C:\Java>"C:\Program Files\Java\ jdk-12\bin\java" Program***
6. Завантажити інсталяційний дистрибутив з офіційного сайту https://www.jetbrains.com/idea/download. За цією адресою можна знайти пакети для Windows, MacOS, Linux. Крім того, саме середовище доступне в двох версіях - **Ultimate** (платна з безкоштовним періодом) і **Community** (безкоштовна). В даному випадку виберемо безкоштовну версію **Community**.

7. Після інсталяції спробувати запустити попередню програму засобами IDE IntelliJ IDEA. Кожен крок показати в звіті у вигляді скріншотів.

ЛАБОРАТОРНА РОБОТА 2

Тема: Примітивні та референсні типи даних в Java.

Мета: ознайомитися з примітивними та референсними типами в мові програмування Java. Навчитися створювати та використовувати змінні в реальній програмі.

Теоретичні відомості

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 цій змінній деяке значення. Причина цього проста: оболонка часу виконання сама по собі не знає, яка кількість пам'яті потрібна для того чи іншого посилального типу.

Усього в мові Java визначено вісім примітивних типів, що перелічені в таблиці 1.2.

Таблиця 1.2. Примітивні типи мови Java

Тип	Розмір	Діапзон	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 922372036854775808	1243565
float	4 байти	Залежить від розрядності числа	1.2f
double	8 байт	Залежить від розрядності числа	123.4d

boolean		false, true	true
char	2 байти	Усі символи стандарту Unicode	‘z’

Введення даних з консолі

Для введення даних у мові програмування Java можна скористатися різними засобами. Один з них використовує спеціальний об'єкт, що належить до класу Scanner. Цей клас містить методи для введення найрізноманітніших типів даних. Приклад його використання наведений нижче:

```
import java.io.*;
import java.util.*;
public class InOutExample {
	public static void main(String[] s) {
		Scanner s = new Scanner(System.in);

		int i = s.nextInt();
		double x = s.nextDouble();
		.....................................................................
	}

}
```

Хід роботи

Написати програму для обчислення наступних математичних виразів:

Варіант 1: $R=x^2(x+1)/b-\boldsymbol{sin}^2(x+a)$; a=0.7, b=0.05, x=0.5

Варіант 2: $s=(xb/a)^{1/2} + \mathbf{cos}^2(x+b)^3$, a=0.8, b=0.04, x=0.6

Варіант 3: $f=(m\boldsymbol{tg}t+|c\boldsymbol{sin}t|)^{1/3}$; m=2; c=-1; t=1.2

Варіант 4: z=m***cos***(bt***sin***t)+c; m=2; t=1.2; b=0.7

Варіант 5 :y=b**tg**^{2}x-a/**sin**^{2}(x/a); a=3.2; b=17.5; x=-4.8

Варіант 6: s=1+x+x^2/2+x^3/6+x^4/24; x=0.335

Варіант 7: f=x(**sin**x^3+**cos**^{2}y); x=0.335; y=0.025

Варіант 8: s=x^3**tg**^{2}(x+b)^2+a/(x+b)$^{1/2}$; a=16.5; b=3.4; x=0.61

Варіант 9: Q=(bx^2-a)/(**e**^{ax}-1); a=16.5; b=3.4; x=0.61

Всі етапи виконання даної лабораторної роботи, а також лістінг самої програми представити в звіті.

Лабораторна робота №3
Програмування на Java алгоритмів розгалуженої структури

Мета роботи: вдосконалення навичок написання Java-коду, ознайомлення з алгоритмічною конструкцією розгалуження в мові Java та її використанням в програмах.

Короткі теоретичні відомості.

Структура Java-програми
Програма на мові Java – це проект (Project), побудований з класів та їх екземплярів (об'єктів). Об'єкти об'єднують в собі набір даних (властивостей), що характеризують їх стан та процедури для опрацювання цих даних (методи), що описують їх поведінку. Клас є загальним описом, на основі якого створюються окремі об'єкти (однакової структури). Клас містить опис назв та типів властивостей об'єктів та опис процесу виконання методів (реалізацію методів). Об'єкт, як представник класу, наділяється конкретними значеннями властивостей, та може опрацьовувати їх описаними в класі методами.
Навіть найпростіший проект (Java-програма) повинен складатися принаймні з одного класу. Цей клас повинен обов'язково містити головний метод

```
public static void main(String[] args) {
      // тут писати код методу
  }
```

Основні правила написання коду

Прості (примітивні) типи даних
В Java є вісім основних (примітивних) типів даних. П'ять із них — цілочисельні (включаючи символьний тип char), два — дійсні (float, double) і один логічний (булевий) тип даних.

Тип	Довжина (в байтах)	Діапазон або набір значень
boolean	не визначено	true, false
byte	1	$-128..127$
char	2	$0..2^{16}-1$, або 0..65535
short	2	$-2^{15}..2^{15}-1$, або −32768..32767
int	4	$-2^{31}..2^{31}-1$, або −2147483648..2147483647
long	8	$-2^{63}..2^{63}-1$, або приблизно $-9.2\cdot10^{18}..9.2\cdot10^{18}$
float	4	$-(2-2^{-23})\cdot2^{127}..(2-2^{-23})\cdot2^{127}$, або приблизно $-3.4\cdot10^{38}..3.4\cdot10^{38}$, а також $-\infty$, ∞, NaN
double	8	$-(2-2^{-52})\cdot2^{1023}..(2-2^{-52})\cdot2^{1023}$, або приблизно $-1.8\cdot10^{308}..1.8\cdot10^{308}$, а також $-\infty$, ∞, NaN

Оператори
Оператор (*англ. operator*) - це спеціальний символ, який повідомляє транслятору про те, що ви хочете виконати операцію з деякими операндами (наприклад, +,-,%, <<). Зазвичай, мови програмування визначають набір опера́торів, подібних до операторів в математиці.

Арифметичні оператори використовуються в математичних виразах так само як і в алгебрі і представлені в таблиці.

Оператор	Операція	Оператор	Операція
+	Додавання	+=	Додаваня з присвоєнням
-	віднімання (а також унарний мінус)	-=	Віднімання з присвоєнням
*	Множення	* =	Множення з присвоєнням
/	Ділення	/=	Ділення з присвоєнням
%	Залишок ділення по модулю	%=	Залишок ділення по модулю з присвоєнням
++	Інкремент (збільшення на 1)	--	Декремент (зменшення на 1)

Алгоритмічна конструкція розгалуження

Умовна інструкція в Java має форму:

```
if (логічний вираз) інструкція;
```

Якщо логічний вираз істинний (true) то буде виконана інструкція за умовою, інакше вона не буде виконана. Якщо необхідно виконати декілька інструкцій, то їх розміщають у блоці:

```
if (умова){
   інструкція 1;
   …..
   інструкція n;
}
```

Якщо ж необхідно здійснити певну дію в разі хибності логічного виразу, то застосовують умовну інструкцію наступного виду:

```
if (умова) інструкція1;
else інструкція2;
```

Логічні операції та вирази
Для побудови простих умов в розгалуженнях чи циклах використовують оператори відношення (чи операторами порівняння) які наведені в таблиці.

Оператор	Опис
==	Рівно
!=	Не рівно
>	Більше
<	Менше
>=	Більше рівне
<=	Менше рівне

Результатом операції порівняння є результат типу boolean із значеннями true або false.

Завдання Написати програму для розв'язання задачі згідно варіанту. Всі дані для роботи програми вводити з консолі. Якщо тип даних в умові не вказано явно, то вважати дані дійсними числами типу float.

1. Дано три цілих числа. Знайти кількість додатних чисел в цьому наборі.

2. Дано три дійсних числа. Знайти кількість від'ємних чисел в цьому наборі.

3. Дано дві змінні дійсного типу: A, B. Перерозподілити значення даних змінних так, щоб A виявилось меншим із заданих значень, а B - більше. Вивести нові значення змінних A і B.

4. Дано дві змінні цілого типу: A і B. Якщо їхні значення не рівні, то присвоїти кожній змінної суму цих значень, а якщо рівні, то присвоїти змінним нульові значення. Вивести нові значення змінних.

5. Дано три числа. Знайти найменше з них.

6. Дано три числа. Знайти середнє з них (тобто число, розташоване між найменшим і найбільшим).

7. Дано три числа. Вивести спочатку найменше, а потім найбільше з даних чисел.

8. Дано три числа. Знайти суму двох більших з них.

9. Дано координати точки, що не лежить на координатних осях OX та OY. Визначити номер координатної чверті, в якій знаходиться дана точка.

10. Дано цілочисельні координати трьох вершин прямокутника, сторони якого паралельні координатним осям. Знайти координати його четвертої вершини.

11. Тіло масою m з об'ємом V занурюють в рідину густиною ρ. Дослідити чи тіло плаватиме в рідині і вивети відповідне повідомлення.

12. Для введених двох чисел визначити найбільшу величину серед їх суми, різниці та добутку.

13. Для введених двох чисел визначити найбільшу величину серед їх суми, добутку та частки.

14. Дано три числа. Знайти суму двох менших з них.

15. Дано три числа. Знайти різницю між найбільшим і найменшим з них.

16. Задано натуральне число з діапазону 1 – 9999. Вивести його опис у вигляді: «одноцифрове число», «двоцифрове число» і т. д.

Приклад виконання роботи
Завдання: 16. Задано натуральне число з діапазону 1 – 9999. Вивести його опис у вигляді: «одноцифрове число», «двоцифрове число» і т. д.

Можна побудувати декілька алгоритмів розв'язання поставленої задачі. Зупинимося на варіанті, в якому спочатку перевіримочи введене число відповідає умові задачі (лежить в діапазоні від 1 до 9999). Якщо введене число дійсно належить вказаному діапазону, то подальшу перевірку можна здійснювати повною формою розгалуження за схемою:
числа менші за 10 – одноцифрові, інші, але менші за 100, – двоцифрові, інші, але менші за 1000 – трицифрові, всі решта – чотирицифрові.

Запускаємо Eclipse. Створюємо новий проект з назвою Lab2, в ньому – одноіменний класс з головним методом main, де записуємо наступний код.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Lab2 {
  public static void main(String[] args) throws IOException {
       // TODO Auto-generated method stub
       System.out.println("Введіть натуральне число від 1 до 9999");
       BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
       int n = Integer.parseInt(reader.readLine());
       if (n > 0 && n<10000) {
              if(n < 10) System.out.println("Одноцифрове число");
              else {
                     if (n < 100) System.out.println("Двоцифрове число");
                     else {
                            if (n < 1000) System.out.println("Трицифрове число");
                            else System.out.println("Чотирицифрове число");
                     }
              }
       }
       else System.out.println("Введене число лежить за межами вказаного діапазону");
  }
}
```

Лабораторна робота № 4

Програмування алгоритмів з багатьма вітками розгалуження на прикладі розв'язання алгебраїчної нерівності. Використання статичних методів класу.

Мета роботи: вдосконалення практичних навичок програмування алгоритмів розгалуженої структури, знайомство з деякими елементами об'єктно-орієнтованого програмування.

Короткі теоретичні відомості.

Структура класів, властивості і методи

Пакети, область видимості

Основні домовленості з іменування об'єктів

Завдання

1. Скласти програму для розв'язування алгебраїчної нерівності (системи нерівностей) згідно варіанту. Алгоритм знаходження розв'язку нерівності оформити у вигляді окремого методу, який для заданих значень числових коефіцієнтів a, b та c визначає розв'язок і повертає текстовий результат у вигляді проміжків на числовій осі. В головному методі класу організувати виклик методу розв'язування нерівності з різними значеннями коефіцієнтів a, b і c так, щоб продемонструвати роботу методу за усіма можливими вітками розгалуження.

Варіанти завдань

№ вар.	Завдання
1	$\frac{a}{x^2+bx+c} \le 0$
2	$ax^2 - bx > c$
3	$\frac{x^2}{ax+b} < c$
4	$\frac{x-a}{x^2+bx+c} \ge 0$
5	$\frac{x^2-ax+b}{x+c} \le 0$
6	$(x-a)(x^2+bx+c) < 0$
7	$\begin{cases} x-a<0, \\ x^2+bx+c>0 \end{cases}$
8	$\begin{cases} x-a\ge 0, \\ x^2+bx+c<0 \end{cases}$

№ вар.	Завдання
9	$\frac{a}{x^2+bx+c} > 0$
10	$ax^2 - bx \le c$
11	$\frac{x^2}{ax+b} \ge c$
12	$\frac{x-a}{x^2+bx+c} < 0$
13	$\frac{x^2-ax+b}{x+c} > 0$
14	$(x+a)(x^2-bx+c) > 0$
15	$\begin{cases} x+a>0, \\ x^2+bx+c\le 0 \end{cases}$
16	$\begin{cases} x-a\ge 0, \\ x^2+bx+c>0 \end{cases}$

Приклад виконання (варіант № 16)

$$\begin{cases} x - a \geq 0, \\ x^2 + bx + c > 0 \end{cases}$$

Результатом виконання роботи має бути програма, яка отримавши від користувача конкретні числові значення коефіцієнтів a, b, c, видавала б розв'язок відносно x нерівності у вигляді проміжків на числовій осі. Головною запорукою правильної роботи програми є ґрунтовний аналіз можливих варіантів процесу розв'язування нерівності, залежно від значень коефіцієнтів a, b та c.

Перша нерівність має розв'язок $x \geq a$, тобто проміжок $[a;+\infty)$. Розв'язок системи залежить від того яким є розв'язок другої нерівності, та як відносно цього розв'язку розташоване на числовій осі число a. Тобто програма повинна обчислити значення дискримінанта D квадратного тричлена в лівій частині другої нерівності та встановити наявність-відсутність його коренів.

Якщо дискримінант від'ємний, то квадратний тричлен не має коренів, а оскільки коефіцієнт при x^2 додатний, то друга нерівність має розв'язком усю множину дійсних чисел R. Отже розв'язком системи буде проміжок $[a;+\infty)$.

При D=0 квадратний тричлен в другій нерівності має коренем число $x_0 = \frac{-b}{2}$, а сама нерівність – розв'язок $(-\infty;x_0)\cup(x_0;+\infty)$. Тоді розв'язок системи залежить від взаємного розташування на числовій прямій чисел a та x_0:

- при $x_0 < a$ розв'язком системи залишиться проміжок $[a;+\infty)$;
- при $x_0 = a$ число a не є розв'язком системи і результатом буде $x \in (a;+\infty)$ (те ж що і $x \in (x_0;+\infty)$);
- при $x_0 > a$ число x_0 слід виключити з проміжка $[a;+\infty)$, отримаємо результат $x \in [a;x_0)\cup(x_0;+\infty)$

При $D > 0$ знайдемо корені $x_1 = \frac{-b-\sqrt{D}}{2}$ та $x_2 = \frac{-b+\sqrt{D}}{2}$ (очевидно, що $x_1 < x_2$) квадратного тричлена в другій нерівності. Нерівність матиме розв'язок $(-\infty;x_1)\cup(x_2;+\infty)$ а для розв'язку системи матимемо такі випадки:

- при $a < x_1$ розв'язок системи $x \in [a;x_1)\cup(x_2;+\infty)$;
- при $x_1 \leq a \leq x_2$ розв'язок системи $x \in (x_2;+\infty)$;
- при $a > x_2$ розв'язком системи залишиться проміжок $[a;+\infty)$.

Враховуючи проведений аналіз, код методу для розв’язування нерівності можна оформити наступним чином:

```
public static String solve(double a, double b, double c){
	String result;
	double d = b*b-4*c;
	if (d<0) result = "["+a+";+inf)";
	else{
		if (d==0){
			double x0 = -b/2;
			if (a < x0) result = "x Є ["+a+";" + x0 + ")U("+x0+";+inf)";
			else {
				if(a == x0) result = "x Є ("+a+";+inf)";
				else  result = "x Є ["+a+";+inf)";
			}
		}
		else{
			double x1 = (-b-Math.sqrt(d))/2;
			double x2 = (-b+Math.sqrt(d))/2;
			if (a < x1) result = "x Є ["+a+";" + x1 + ")U("+x2+";+inf)";
			else{
				if (a <= x2) result = "x Є ("+x2+";+inf)";
				else result = "x Є ["+a+";+inf)";
			}
		}
	}
	return result;
}
```

Створюємо новий клас з головним методом main, в тілі класу записуємо код наведеного вище методу `solve`. В тілі головного методу `main` слід реалізувати виклики методу `solve` з різними наборами числових значень коефіцієнтів a, b та c, кожен з яких відповідає одному з варіантів системи, розглянутих в аналізі. Він може виглядати так:

```
public static void main(String[] args) {
	// TODO Auto-generated method stub
	System.out.println("a=-10, b=-1, c=-6\n" + solve(-10, -1, -6));
	System.out.println("a=-2, b=-1, c=-6\n" + solve(-2, -1, -6));
	System.out.println("a=-1, b=-1, c=-6\n" + solve(-1, -1, -6));
	System.out.println("a=3, b=-1, c=-6\n" + solve(2, -1, -6));
	System.out.println("a=10, b=-1, c=-6\n" + solve(10, -1, -6));
	System.out.println("a=-10, b=2, c=1\n" + solve(-10, 2, 1));
	System.out.println("a=-1, b=2, c=1\n" + solve(-1, 2, 1));
	System.out.println("a=10, b=2, c=1\n" + solve(10, 2, 1));
	System.out.println("a=10, b=2, c=10\n" + solve(10, 2, 10));
}
```

Лабораторна робота № 5
Програмування алгоритмів циклічної структури

Мета роботи: знайомство з циклічними алгоритмічними конструкціями мови Java.

Короткі теоретичні відомості

Цикли

Цикли в програмах дозволяють організувати повторення тієї чи іншої послідовності команд. В Java реалізовані три конструкції циклу:

- цикл while ;
- цикл do ;
- цикл for.

Цикл while має таку форму:

```
while(<логічний вираз>)
   <команда або блок команд>;
```

Блок з команд утворюють за допомогою фігурних дужок. В такій конструкції команда або блок буде виконуватися доки логічний вираз матиме значення true («істина»), звідки і назва «цикл доки». Команду чи блок команд, що повторюється в циклі називають тілом циклу. Можлива ситуація коли логічний вираз хибний при першому звертанні до циклу, тоді його тіло не виконується жодного разу.

Цикл do має подібну форму та дію до циклу while але перевірка логічного виразу розташовується після тіла циклу. Тіло такого циклу виконається принаймні один раз. Його форма:

```
do
   <команда чи блок команд;>
while(логічний вираз)
```

Цикл for має вигляд:

```
for(вираз ініціалізації; умова; вираз оновлення)
<тіло циклу;>
```

Вираз ініціалізації виконується один раз - перед першим виконання тіла циклу. Умова перевіряється після кожного виконання тіла циклу. Якщо результат перевірки умови істинний то цикл повторюється, якщо хибний – повторення завершується. Вираз оновлення виконується кожного разу після чергового повторення тіла, але перед перевіркою умови. І вираз ініціалізації і вираз оновлення можуть складати з декількох дій (операторів), які розділені комою. Наприклад, обчислити суму цілих чисел від 1 до 10 можна так:

```
int s=0;
for(int i=1;i<=10;s+=i,i++){};
System.out.println(s);
```

Цикл for, зазвичай, використовують коли мова йде про перерахунок значень якоїсь величини, і виконання для кожного з них певних дій. Наприклад, для опрацювання масивів.

Всі три конструкції циклів є взаємозамінними. Зокрема записану вище загальну форму циклу for можна подати у формі while так:

```
<вираз ініціалізації;>
while (умова){
<тіло циклу;>
<вираз оновлення>
}
```

Тому деколи жартома говорять що цикл for є компактнішою формою запису циклу while.

Оператори break и continue

Оператор continue можна використовувати в циклах while, do, for для дострокового припинення чергового виконання тіла циклу. Якщо під час виконання програми зустрічається оператор continue, то виконання поточної послідовності команд припиняється і керування передається на початок блоку, що містить цей оператор.

Оператор break виконує дострокове припинення виконання усього циклу. Якщо після виконання оператора continue умова повторення істинна, то тіло циклу буде виконане наступний раз і т.д. Після виконання команди break наступних повторів тіла циклу не буде.

Завдання. Скласти блок-схему та програму розв'язання задачі відповідно до варіанта використовуючи:
a) циклічну конструкцію ***while;***
b) циклічну конструкцію ***do;***
c) циклічну конструкцію ***for.***

1. Дано ціле число N (> 0). Послідовність дійсних чисел A_k , задана за допомогою формул $A_0 = 2$, $A_k = 2 + \frac{1}{A_{k-1}}$, k = 1, 2, … . Вивести елементи $A_1, A_2, \ldots, A_N$.

2. Дано ціле число N (> 0). Послідовність дійсних чисел A_k , задана за допомогою формул $A_0 = 1$, $A_k = \frac{A_{k-1} + 1}{k}$, k = 1, 2, … . Вивести елементи $A_1, A_2, \ldots, A_N$.

3. Дано дійсне число A и ціле число N (> 0). Використовуючи один цикл, знайти суму
$$1 + A + A^2 + A^3 + \ldots + A^N .$$

4. Дано дійсне число A и ціле число N (> 0). Використовуючи один цикл і не використовуючи умовного оператора, знайти суму
$$1 - A + A^2 - A^3 + \ldots + (-1)^N A^N .$$

5. Дано два цілі числа A и B (A < B). Знайти суму всіх цілих чисел від A до B включно.

6. Для заданого натурального числа N знайти суму
$$\frac{1}{1!} + \frac{2}{2!} + \frac{3}{3!} + \ldots + \frac{N}{N!}$$
використовуючи лише один цикл.

7. Дано два цілі числа A и B (A < B). Знайти суму всіх парних цілих чисел від 2A до 3B включно.

8. Для заданого натурального числа N і дійсного числа X знайти суму

$$\frac{1}{X}+\frac{1}{X^2}+\frac{1}{X^3}+\ldots+\frac{1}{X^N}$$

використовуючи лише один цикл.

9. Дано два цілі числа A и B (A < B). Знайти добуток всіх цілих чисел від A до B включно.

10. Вивести таблицю квадратів усіх натуральних чисел, починаючи від заданого натурального K, до заданого натурального N (N>K)

11. Дано ціле число N (> 0). Послідовність натуральних чисел A_k , задана за допомогою формул $A_1=1,\ A_2=1,\ A_k=A_{k-1}+2A_{k-2}$, k = 1, 2, … . Вивести елементи $A_1, A_2, \ldots, A_N$.

12. Дано два цілі числа A и B (A < B). Знайти суму квадратів усіх цілих чисел від A до B включно.

13. Для заданого додатного дійсного числа X знайти суму

$$\frac{X}{1}-\frac{X}{2}+\frac{X}{3}-\frac{X}{4}+\ldots+\frac{X}{N}$$

де N найближче до X натуральне число, менше за X .

14. Задано ціле число N (> 0). Знайти суму

$$N^2+(N+1)^2+(N+2)^2+\ldots+(2N)^2$$

(ціле число).

15. Задано ціле число N (N > 0). Знайти добуток

$$1.1\cdot 1.2\cdot 1.3\cdot \ldots \quad 1+0.1*N$$

(N множників).

16. Задано ціле число N (N > 0). Знайти квадрат даного числа, використовуючи для його обчислення формулу:

$$N^2=1+3+5+\ldots+(2N+1).$$

Після додавання до суми наступного доданка виводити поточне значення.

Приклад (варіант 16). Зміст завдання полягає в тому, щоб програма для введеного числа N знаходила послідовно усі доданки задані формулою і додавала їх до значення деякої змінної, в якій буде зберігатися їх сума (початкове значення суми, очевидно, повинно дорівнювати 0).
Блок-схема:
Оскільки в завданні сказано реалізувати програму з використанням усіх трьох циклічних конструкцій, то слід створити три окремих проекти, або помістити всі три варіанти в одному класі у вигляді окремих методів, а в головному методі main послідовно організувати їх виклики.
Спосіб 1. Програми в окремих класах:
а) з використанням циклу ***while;***

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class lab4a {

public static void main(String[] args) throws NumberFormatException, IOException {
		System.out.println("Введіть число N");
		BufferedReader read =
			new BufferedReader(new InputStreamReader(System.in));
		int N = Integer.parseInt(read.readLine());
		int S = 0;
		int a = 1;
		System.out.println("Отримані суми:");
		while(a<=2*N-1){
			S+=a;
			a+=2;
			System.out.println(S);
		}
	}
}
```

b) з використанням циклу ***do;***

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class lab4b {

public static void main(String[] args) throws NumberFormatException, IOException {
		System.out.println("Введіть число N");
		BufferedReader read =
			new BufferedReader(new InputStreamReader(System.in));
		int N = Integer.parseInt(read.readLine());
		int S = 0;
		int a = 1;
		System.out.println("Отримані суми:");
		do{
			S+=a;
			a+=2;
			System.out.println(S);
		}
		while(a<=2*N-1);
	}

}
```

c) з використанням циклу ***for.***

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class lab4c {

public static void main(String[] args) throws NumberFormatException, IOException {
		// TODO Auto-generated method stub
		System.out.println("Введіть число N");
		BufferedReader read =
			new BufferedReader(new InputStreamReader(System.in));
		int N = Integer.parseInt(read.readLine());
		int S = 0;
		System.out.println("Отримані суми:");
		for (int a = 1;a<=2*N-1;a+=2){
			S+=a;
			System.out.println(S);
		}
	}
}
```

Спосіб 2. Програми – методи одного і того ж класу:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Lab4d {

public static void main(String[] args) throws NumberFormatException, IOException {
		System.out.println("Введіть число N");
		BufferedReader read =
			new BufferedReader(new InputStreamReader(System.in));
		int N = Integer.parseInt(read.readLine());
		System.out.println("Результат з використанням циклу while");
		withWileMethod(N);
		System.out.println("Результат з використанням циклу do");
		withWileMethod(N);
		System.out.println("Результат з використанням циклу for");
		withWileMethod(N);
	}
	static void withWileMethod(int k){
		int S = 0;
		int a = 1;
		System.out.println("Отримані суми:");
		while(a<=2*k-1){
			S+=a;
			a+=2;
			System.out.println(S);
		}
	}
```

```
	static void withDoMethod(int k){
		int S = 0;
		int a = 1;
		System.out.println("Отримані суми:");
		do{
			S+=a;
			a+=2;
			System.out.println(S);
		}
		while(a<=2*k-1);
	}
	static void withForMethod(int k){
		int S = 0;
		System.out.println("Отримані суми:");
		for (int a = 1;a<=2*k-1;a+=2){
			S+=a;
			System.out.println(S);
		}
	}
}
```

Лабораторна робота № 6

Ітераційні циклічні алгоритми. Обчислення значення функції як суми нескінченного ряду. Використання методів класу Math для математичних обчислень.

Мета роботи: оволодіння практичними навичками розробки та програмування алгоритмів ітераційної циклічної структури, створення та використання статичних методів класів, виконання математичних обчислень засобами класу Math.

Короткі теоретичні відомості

Клас Math належить до пакету java.lang, що містить усі службові класи, потрібні для роботи інших програм-класів (наприклад, клас String, що використовується для обробки текстових стрічок). Math містить статичні методи, що виконують обчислення найбільш поширених математичних функцій а також значення констант:

- public static final double Math.PI –число π;
- public static final double Math.E – основа натурального логарифма, число e.

В таблиці нижче наведено деякі методи цього класу.

Тип результату	**Назва методу і тип параметрів**	**Функція, що обчислюється**
Залежно від типу аргументу	abs(… a)	абсолютне значення (модуль) числа, тип аргумента double, float, int або long
Double	acos(double a)	Арккосинус
Double	asin(double a)	Арксинус
Double	atan(double a)	Арктангенс
Double	sin(double a)	синус кута a (в радіанах)
Double	cos(double a)	Косинус
Double	tan(double a)	Тангенс
Double	exp(double a)	e^x
Double	log(double a)	натуральний логарифм a
Double	pow(double a, double b)	a^b
Double	random()	випадкове число від 0.0 до 1.0
Double	rint(double a)	значення типу int, найближче до a (заокруглення до цілого)
Long	round(double a)	заокруглення до цілого
Int	round(float a)	заокруглення до цілого
double	sqrt(double a)	$\sqrt{a}$
double	toDegrees(double a)	перетворення аргументу з радіанної міри в градусну

double	toRadians(double a)	перетворення аргументу з градусної міри в радіанну

Оскільки всі перелічені функції є статичними методами класу Math, то перед їх викликом слід через крапку вказувати ім'я класу, наприклад, значення синуса 42-х градусів одержується так:

Math.sin(Math.toRadians(42))

Завдання Створити клас, в якому реалізувати статичні методи для обчислення значень вказаних у варіанті елементарних функцій за допомогою степеневих рядів з точністю до члена, меншого за абсолютною величиною за 0.00001. Обчислити значення заданих виразів у заданих точках з використанням методів створеного класу, та за допомогою методів класу Math. Порівняти результати (вивести модуль різниці).

Варіанти завдань

№ .	*Функції*	Вираз	Значення x
1	$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$ $arctg\, x = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{2k+1}$ $\ln(1+x) = \sum_{k=1}^{\infty} (-1)^{k-1} \frac{x^k}{k}$	$f(x) = \frac{e^{\frac{1}{\ln x}}}{e^{arctg(x-0,4)} + e^{arctg(x-0,8)}}$	0.2 0.7 1.2
2	$\sin x = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$ $\cos x = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$ $\frac{1}{(1-x)^2} = \sum_{k=0}^{\infty} (k+1)x^k$	$f(x) = \frac{\sin x + \cos x}{(1-x)^4}$	-0.5 0 0.5
3	$sh\, x = \sum_{k=0}^{\infty} \frac{x^{2k+1}}{(2k+1)!}$ $ch\, x = \sum_{k=0}^{\infty} \frac{x^{2k}}{(2k)!}$ $\ln(1+x) = \sum_{k=1}^{\infty} (-1)^{k-1} \frac{x^k}{k}$	$f(x) = \frac{sh\, x - ch x}{\log_{\frac{1}{2}}(1+x)}$	0 0.5 1
4	$\sin x = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$ $\cos x = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$ $e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$	$f(x) = e^{5x} \sin 7x + e^{7x} \cos 5x$	3 5 7
5	$arctg\, x = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{2k+1}$	$f(x) = \frac{5}{1+arctg\, x} + \frac{15}{1-arctg\, x}$	-0.2 0.1 0.4

	$\frac{1}{1+x}=\sum_{k=0}^{\infty}(-1)^k x^k$ $\frac{1}{1-x}=\sum_{k=0}^{\infty}x^k$		
6	$\sin x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{(2k+1)!}$ $arctg\,x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{2k+1}$ $\ln(1+x)=\sum_{k=1}^{\infty}(-1)^{k-1}\frac{x^k}{k}$	$f(x)=\frac{arctg\left(\frac{1+\sin 3x}{2}\right)}{\log_{\frac{1}{3}}(1+\sin 5x)}$	2 7 12
7	$\sin x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{(2k+1)!}$ $\cos x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k}}{(2k)!}$ $\frac{1}{1+x}=\sum_{k=0}^{\infty}(-1)^k x^k$	$f(x)=\frac{15}{1+\sin 3x}+\frac{1}{1+\cos 5x}$	0 3 5
8	$\sin x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{(2k+1)!}$ $ch\,x=\sum_{k=0}^{\infty}\frac{x^{2k}}{(2k)!}$ $\ln(1+x)=\sum_{k=1}^{\infty}(-1)^{k-1}\frac{x^k}{k}$	$f(x)=\frac{\ln\frac{x}{5}}{1+\sin 3x}+\frac{ch\frac{x}{3}}{1+\sin 5x}$	1 2 3
9	$sh\,x=\sum_{k=0}^{\infty}\frac{x^{2k+1}}{(2k+1)!}$ $\cos x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k}}{(2k)!}$ $\ln(1+x)=\sum_{k=1}^{\infty}(-1)^{k-1}\frac{x^k}{k}$	$f(x)=\frac{\ln\frac{x}{4}}{1+\cos 3x}+\frac{sh\frac{x}{3}}{1+\cos 5x}$	1 2 3
10	$\sin x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{(2k+1)!}$ $\cos x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k}}{(2k)!}$ $\frac{1}{1-x}=\sum_{k=0}^{\infty}x^k$	$f(x)=\frac{25}{1-\sin 33x}+\frac{33}{1-\cos 25x}$	10 23 35
11	$\sin x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{(2k+1)!}$ $arctg\,x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{2k+1}$ $\ln(1+x)=\sum_{k=1}^{\infty}(-1)^{k-1}\frac{x^k}{k}$	$f(x)=\frac{\ln(1+\sin 33x)}{arctg\left(\frac{1}{2}-\frac{1}{3}\sin 33x\right)}$	10 23 35

12	$sh\,x=\sum_{k=0}^{\infty}\frac{x^{2k+1}}{(2k+1)!}$ $arctg\,x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{2k+1}$ $\frac{1}{1+x}=\sum_{k=0}^{\infty}(-1)^k x^k$	$f(x)=\frac{sh\left(\frac{1}{1+2}\right)}{arctg\left(\frac{1}{2}-\frac{x}{3}\right)}$	0.10 0.23 0.35
13	$\cos x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k}}{(2k)!}$ $arctg\,x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{2k+1}$ $\ln(1+x)=\sum_{k=1}^{\infty}(-1)^{k-1}\frac{x^k}{k}$	$f(x)=\frac{arctg\left(\frac{1+\cos 7x}{2}\right)}{\log_{\frac{1}{5}}(1+\cos 3x)}$	-2 0.7 1.2
14	$\sin x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{(2k+1)!}$ $\cos x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k}}{(2k)!}$ $\ln(1+x)=\sum_{k=1}^{\infty}(-1)^{k-1}\frac{x^k}{k}$	$f(x)=\frac{\log_{\frac{1}{5}}(1+\cos 3x)}{\log_{\frac{1}{3}}(1+\sin 5x)}$	-12 -7 2
15	$\cos x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k}}{(2k)!}$ $\frac{1}{1+x}=\sum_{k=0}^{\infty}(-1)^k x^k$ $\frac{1}{1-x}=\sum_{k=0}^{\infty}x^k$	$f(x)=\frac{5}{1+\cos 7x}-\frac{7}{1-\cos 5x}$	20 30 40

Приклад

№	Функції	Вираз	Значення x
16	$\sin x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{(2k+1)!}$ $\frac{1}{1+x}=\sum_{k=0}^{\infty}(-1)^k x^k$ $\frac{1}{1-x}=\sum_{k=0}^{\infty}x^k$	$f(x)=\frac{15}{1+\sin 3x}+\frac{1}{1-\sin 5x}$	1 4 7

Завдання поєднує в собі декілька задач:
Запрограмувати алгоритми знаходження значення заданих функцій для заданого значення x;
Реалізацію алгоритмів знаходження функцій оформити у вигляді методів окремого класу;
Обчислити значення заданого виразу за допомогою методів створеного класу;
Обчислити значення виразу за допомогою методів класу Math і порівняти результати.

Алгоритм знаходження значення функції у вигляді суми ряду полягає в циклічному знаходженні наступного доданка та додавання його до значення деякої змінної, де міститься поточне значення суми. Обчислення загального члена ряду (доданка) безпосередньо за

формулою під знаком суми не практикується. Наприклад, в третій з заданих функцій $\frac{1}{1-x}=\sum_{k=0}^{\infty}x^k=1+x+x^2+x^3+\ldots+x^k+\ldots$ знайти k-ий член ряду x^k можна за допомогою методу pow модуля Math, чи k-кратного множення x на x. Але набагато простіше знаходячи наступний доданок взяти значення попереднього доданка і помножити його на x. Математично це виглядає так $a_{k+1}=a_k\cdot x$. Проте в програмі достатньо однієї змінної для зберігання чергового доданка, тому запишемо

a = a * x

або, з використанням спеціальної форми присвоєння

a *= x

Якщо для організації циклу скористатися оператором for, то це присвоєння можна розмістити в завершальній частині його оголошення. Враховуючи, що за умовою суму нараховуємо з точністю до члена, меншого за абсолютною величиною за 0.00001 запишемо такий код методу:

```
public static double fraction2(double x){
	double s = 0;
	for(double a = 1; Math.abs(a) >= 0.00001; a*=x){
		s+=a;
	}
	return s;
}
```

Тут `Math.abs(a)` використано для обчислення модуля числа a, за бажання повністю відмовитися від послуг стандартного модуля можна створити додатковий допоміжний метод abs в свому класі:

```
private static double abs(double x){return x>=0?x:-x;}
```

Тоді в коді методу `fraction2` звертання до класу `Math` можна уникнути.

Реалізація обчислення першого дробу відрізняється від другого лише тим, що для знаходження наступного доданка попередній слід домножувати на –x. Для обчислення функції $\sin x=\sum_{k=0}^{\infty}(-1)^k\frac{x^{2k+1}}{(2k+1)!}$ слід розрахувати доповнювальний множник діленням наступного доданка на попередній. За формулою $a_k=(-1)^k\frac{x^{2k+1}}{(2k+1)!}$, тоді $a_{k-1}=(-1)^{k-1}\frac{x^{2(k-1)+1}}{(2(k-1)+1)!}=(-1)^{k-1}\frac{x^{2k-1}}{(2k-1)!}$.

Отже $\frac{a_k}{a_{k-1}}=(-1)^k\frac{x^{2k+1}}{(2k+1)!}:(-1)^{k-1}\frac{x^{2k-1}}{(2k-1)!}=\frac{(-1)^k\cdot x^{2k+1}\cdot(2k-1)!}{(-1)^{k-1}\cdot x^{2k-1}\cdot(2k+1)!}=-\frac{x^2}{(2k+1)2k}$.

Тому $a_k=-\frac{x^2}{(2k+1)2k}\cdot a_{k-1}$ і нарахування наступного доданка можна виконувати присвоєнням

a = -x*x*a/((2*k+1)*2*k).

Остаточний код класу MyMath:

```
package lab6;

public class MyMath {

private static double abs(double x){return x>=0?x:-x;}
```

```
public static double sin(double x){
	double s = 0;
	double a = x;
	for(int k = 1; abs(a) >= 0.00001; k++ ){
		s+=a;
		a=-a*x*x/((2*k)*(2*k+1));
	}
	return s;

}

public static double fraction1(double x){
	double s = 0;
	for(double a = 1; abs(a) >= 0.00001;a*=-x){
		s+=a;
	}
	return s;
}

public static double fraction2(double x){
	double s = 0;
	for(double a = 1; abs(a) >= 0.00001; a*=x){
		s+=a;
	}
	return s;
}
}
```

Залишається створити ще однин клас. В ньому слід реалізувати метод main для перевірки функції з MyMath. Оскільки задані значення змінюються з однаковим кроком, то для їх перебору можна скористатися циклом (в інакшому випадку слід по черзі присвоїти кожне значення). Орієнтовний код класу

```
package lab6;
public class Lab6 {

	public static void main(String[] args) {

		for(double t = 1; t <= 7; t+=3){
			double f1=0,f2;
			f1 = 15*MyMath.fraction1(MyMath.sin(3*t))+
			MyMath.fraction2(MyMath.sin(5*t));
			f2 = 15/(1 + Math.sin(3*t))+1/(1 - Math.sin(5*t));
			System.out.println("Значення x = " + t);
			System.out.println("Значення виразу з використанням створених " +
			" функцій: " + f1);
			System.out.println("Значення виразу з використанням стандартних" +
			" функцій: " + f2);

		}


	}

}
```

Лабораторна робота № 7

Опрацювання масивів на Java. Одновимірні масиви. Пошук в одновимірних масивах.

Мета роботи: вивчення основних прийомів роботи з одновимірними масивами на мові Java.

Короткі теоретичні відомості.

Масив - це впорядкований набір однотипних елементів для звертання до яких використовують спільне ім'я. В Java можна створювати масиви будь-якого типу, вони можуть бути як одновимірні так і багатовимірні. До окремого елемента масиву звертаються за його індексом, індекс записують в квадратних дужках.

Загальна форма оголошення одновимірного масиву:

```
<тип елементів> <назва масиву>[];
```

Квадратні дужки після імені змінної власне і вказують, що змінна є масивом, їх можна записувати також і після типу:

```
<тип елементів>[] <назва масиву>;
```

Обидві форми є ідентичними.

В Java масиви є об'єктами, тому перед їх використанням слід створювати екземпляр масиву оператором new. При цьому вказується прогнозований розмір (кількість елементів) масиву, наприклад:

```
double[] x;
int a[];
x = new double[10];
a = new int[135];
```

Зазвичай команди опису ы створення масиву об'єднують:

```
float[] p = new float[20];
```

Розмір масиву в Java можна змінювати в процесі виконання програми за допомогою того ж оператора new. Для роботи з елементами масиву,використовують цикл for, на кшталт:

```
for(int i=0;i<20;i++)
        System.out.println(p[i]);
```

Ввід елементів масиву можна здійснити за допомогою циклу, але для зручності дозволяється ініціалізувати масив елементами відразу при його описі, наприклад:

```
int p[] = {1, 3, 4, 5, 2, 6, 5, 9};
```

В такому випадку автоматично створюється екземпляр масиву, а його розмір визначається за кількістю елементів, записаних в фігурних дужках.

Масиви є об'єктами – відповідно мають визначені поля і методи. Зокрема, поле length містить інформацію про кількість елементів масиву на даному етапі виконання програми і дозволяє опрацьовувати усі елементи масиву циклом

```
for(int i=0;i<p.length;i++)<дії з p[i]>;
```

Завдання. Скласти Java-програму розв'язання задачі відповідно до варіанта. Задано три масиви дійсних чисел A[10], B[10] та C[10], кожен містить по 10 елементів. Масив A заповнити довільно в коді програми при його ініціалізації. Масив B заповнити за вказаним правилом. Масив C утворити з елементів масивів A та B згідно варіанта. Знайти в кожному з масивів вказану величину, вивести на консоль елементи кожного масиву у порядку зростання. Алгоритми пошуку, сортування та друку масивів запрограмувати у вигляді окремих статичних методів класу.

1. Масив B утворити за правилом $B_k = \dfrac{(-1)^k\left(2k^2+1\right)}{k}$ $k = 0,1,...,9$. Масив C утворити з масиву B, замінивши в ньому всі від'ємні елементи відповідними елементами масиву A (з тими ж порядковими номерами). В кожному масиві знайти середнє арифметичне додатних елементів.

2. Масив B заповнити випадковими числами з відрізка [-2; 3] (скористатися методом random() класу Math). Масив C утворити з масиву B, замінивши в ньому всі додатні елементи максимальним елементом масиву A. В кожному масиві знайти кількість елементів, більших за їх середнє арифметичне.

3. Масив B утворити за правилом $B_k = \dfrac{99\sin k}{(k+1)^2}$, $k = 0,1,...,9$. Масив C утворити з масиву B, замінивши в ньому всі від'ємні елементи мінімальним елементом масиву A. В кожному масиві знайти кількість елементів, абсолютна величина яких більша за 5.

4. Масив B утворити за правилом $B_k = (-1)^k\sqrt{k!}$, $k = 0,1,...,9$. Масив C утворити за правилом $C_k = 2A_k - 3B_k$. В кожному масиві знайти суму додатних елементів.

5. Масив B утворити за правилом $B_k = 11\sin\left(\sqrt{(k+3)^3}\right)$, $k = 0,1,...,9$. Масив C утворити за правилом $C_k = \max\left\{\dfrac{A_k}{B_k};\dfrac{B_k}{A_k}\right\}$. В кожному масиві знайти суму елементів, абсолютна величина яких менша за 1.

6. Масив B заповнити випадковими числами з відрізка [-4; 2] (скористатися методом random() класу Math). Масив C утворити з масиву B, замінивши в ньому всі від'ємні елементи середнім значенням елементів масиву A. В кожному масиві знайти елемент, значення якого найближче до числа 1.

7. Масив B утворити за правилом $B_k = k\sin(2k) + 2k\sin k$, $k = 0,1,...,9$. Масив C утворити за правилом $C_k = \max\{A_k; B_k\}$. В кожному масиві знайти різницю максимального та мінімального елемента (розмах значень елементів масиву).

8. Масив B утворити за правилом $B_k = 20\cos k - k$, $k = 0,1,...,9$. Масив C утворити з масиву B, віднявши від кожного його елемента відповідний елемент масиву A. В кожному масиві знайти суму квадратів мінімального та максимального елементів.

9. Масив B утворити за правилом $B_k = (-1)^k \sqrt{(k+2)(k+1)}$, $k = 0,1,...,9$. Масив C утворити за правилом $C_k = \min\{A_{9-k}; B_k\}$. В кожному масиві знайти різницю між максимальним елементом та середнім арифметичним усіх елементів.

10. Масив B утворити заповнити випадковими числами з відрізка [-5; 5] (скористатися методом random() класу Math). Масив C утворити з елементів масиву A, додавши до кожного середнє арифметичне від'ємних елементів масиву B. В кожному масиві знайти відношення максимального елемента до мінімального.

11. Масив B утворити за правилом $B_k = 15\cos k - 12\sin(10-k)$, $k = 0,1,...,9$. Масив C утворити за правилом $C_k = \min\{3A_k; B_k\}$. В кожному масиві знайти відношення абсолютної величини суми від'ємних елементів до суми додатних елементів.

12. Масив B утворити за правилом $B_k = \dfrac{k\sin(2k)}{(k+1)!}$, $k = 0,1,...,9$. Масив C утворити за правилом $C_k = 2A_k + B_k$. В кожному масиві знайти суму від'ємних елементів.

13. Масив B утворити за правилом $B_k = 12\cos\left(\sqrt[k+1]{k^2}\right)$, $k = 0,1,...,9$. Масив C утворити за правилом $C_k = \max\{2A_k; 5B_k\}$. В кожному масиві знайти номер елемента, найближчого за величиною до числа 5.

14. Масив B утворити заповнити випадковими числами з відрізка [-2.5; 1.5] (скористатися методом random() класу Math). Масив C утворити за правилом $C_k = \min\{A_k; B_{9-k}\}$. В кожному масиві знайти найбільший серед від'ємних елементів.

15. Масив B утворити за правилом $B_k = \dfrac{\cos(12k)}{k!}$, $k = 0,1,...,9$. C утворити з масиву B, замінивши в ньому всі додатні елементи мінімальним елементом масиву A. В кожному масиві знайти номер елемента, найближчого за величиною до середнього арифметичного усіх елементів масиву.

16. Масив B утворити за правилом $B_k = 2\sin k + \cos k$, $k = 0,1,...,9$. Масив C утворити додаванням оберненого масиву A до масиву B, тобто $C_0 = A_9 + B_0$, $C_1 = A_8 + B_1$, $C_2 = A_7 + B_2$, і т.д. В кожному масиві знайти кількість елементів, значення яких належать інтервалу (-1; 1).

Хід виконання. Масиви будемо передавати методам друку сортування та пошуку в якості параметра. За рахунок поля length в методі завжди можна дізнатися скільки елементів містить переданий масив, наприклад, вивести на екран елементи масиву можна методом:

```
public static void printArray(double[] ar){
	String s="";
	for(int i=0; i<ar.length;i++)s+=(ar[i]+", ");
		System.out.println(s);
}
```

Для знаходження кількості елементів з інтервалу (-1; 1) створюємо метод:

```
public static int searchInArray(double[] ar){
		int k =0;
		for(int i=0; i<ar.length;i++)
			if(ar[i]>-1 & ar[i]<1)k++;
		return k;
	}
```

Для сортування масивів, скористаємося методом мінімального елемента, який полягає в наступному. Спочатку знаходимо найменший елемент масиву та його порядковий номер, всі елементи, які розташовані в масиві перед мінімальним зміщуємо на одну позицію далі. Тим самим звільняється перша позиція, куди записуємо мінімальний елемент. Далі розглядаємо масив без першого елемента. В ньому знову знаходимо найменший елемент і переносимо його на першу (для початкового масиву це друга) позицію, змістивши потрібні елементи далі. Потім розглядаємо масив без перших двох елементів і повторюємо процедуру переміщення мінімального елемента на початок масиву, і т. д. Орієнтовний код методу сортування:

```
public static void sortArray(double[] ar){
	for(int k=0; k<ar.length-1; k++){
		double min=ar[k];
		int iMin = k;
		for(int i=k; i<ar.length; i++){
			if(min>ar[i]){min=ar[i];  iMin=i;}
		}
		for(int i=iMin;i>k;i--)   ar[i]=ar[i-1];
		ar[k]=min;
	}
```

В головному методі класу залишається утворити задані масиви та по черзі викликати відповідні методи:

```
public static void main(String[] args) {
		double[] a =  {2.1,0.4,0.1,-0.5,0.6,7.1,-2.4,-1.3,-0.7,-9};
		double b[] = new double[10];
		double c[] = new double[10];
		for(int i=0; i<10;i++) b[i]=2*Math.sin(i)+Math.cos(i);
		for(int i=0; i<10;i++)
			c[i] = a[9-i]+b[i];
		System.out.println("Початкові масиви");
		System.out.println("Масив a: ");
		printArray(a);
		System.out.println("Масив b: ");
		printArray(b);
		System.out.println("Масив c: ");
		printArray(c);
		System.out.println("Сортовані масиви");
		sortArray(a);
		sortArray(b);
		sortArray(c);
		System.out.println("Масив a: ");
		printArray(a);
		System.out.println("Масив b: ");
		printArray(b);
		System.out.println("Масив c: ");
		printArray(c);
		System.out.println("Кількість значень з інтервалу (-1;1)");
		System.out.println("Масив a: " + searchInArray(a));
		System.out.println("Масив b: " + searchInArray(b));
		System.out.println("Масив c: " + searchInArray(c));
	}
```

Лабораторна робота № 8

Робота з двовимірними масивами. Читання даних з текстового файлу та запис в текстовий файл.

Мета роботи: Оволодіння практичними навичками роботи з двовимірними масивами, читання-запису текстових файлів та опрацювання виняткових ситуацій.

Короткі теоретичні відомості.

Двовимірний масив в Java є не що інше як масив одновимірних масивів. Елементи двовимірних масивів, відповідно, індексуються парою індексів – кожен записують в окремих квадратних дужках. Загальна форма оголошення двовимірного масиву:

```
<тип елементів> <назва масиву>[<розмірність1>][<розмірність2>];
```

Як і у випадку одновимірного масиву дозволяється записувати розмірність масиву відразу і після типу елементів. Приклади оголошення двовимірних масивів:

```
int[][] A = new int[5][5];
float B[][] = new float[3][2];
```

Для опрацювання двовимірних масивів використовують вкладені цикли for, наприклад:

```
int max=A[1][1];
for(int i=0;i<5;i++)
      for(int j=0;j<5;j++)
            if (max < A[i][j]) max = A[i][j];
```

Дозволяється також ініціалізація двовимірного масиву елементами під час опису, наприклад:

```
int[][]A ={
      {5, 6, 1, 3},
      {3, 4, 2, 1},
      {1, 2, 2, 2}
};
```

Для **читання даних** з текстового файлу можна використовувати клас BufferedReader, як і для зчитування даних з консолі. Для цього його слід ініціалізувати представником класу FileReader, який, в свою чергу, конструюється на основі об’єкта класу File:

```
BufferedReader input = new BufferedReader(new FileReader(new File("data.txt")))
```

Але такий спосіб передбачає використання методів read() чи readLine(), якими складно опрацьовувати стрічки тексту, що містять по декілька числових даних. Набагато зручнішим в таких випадках є використання класу java.util.Scanner, що має метод hasNext() для перевірки того чи є у вхідному потоці чергова порція даних. На випадок вводу цілих чисел використовують метод hasNextInt() (для дійсних hasNextFloat(), hasNextDouble()). Якщо наступна порція даних є, то її можна прочитати методами next(), nextInt() (nextFloat, nextDouble), відповідно. Організувати такий ввід даних можна аналогічно до попереднього:

```
Scanner input = new Scanner(new FileReader(new File("data.txt")))
```

Для **запису даних** використовують потік виводу що є екземпляром классу PrintWriter. Цей об’єкт взаємодіє з потоком виводу в файл типу FileWriter, через відповідний буфер BufferedWriter. Наприклад

```
File fB = new File("D:\\B.txt");
FileWriter fileWrite = new FileWriter(fB);
BufferedWriter writeBufer = new BufferedWriter(fileWrite);
PrintWriter printB = new PrintWriter(writeBufer);
```

Або коротше з використанням анонімних об’єктів:

```
FileWriter fileWrite = new FileWriter(new File("D:\\B.txt"));
PrintWriter printB = new PrintWriter(new BufferedWriter(fileWrite));
```

При цьому можна використовувати ті ж методи print(), println(), printf(), format(), що і при виводі даних на консоль.

Зчитування даних з файлів – процес при якому часто можуть виникати помилки, наприклад, відсутність файла з даними, чи неправильний запис даних у файлі. У класичних мовах програмування, наприклад, в C, доводилося перевіряти якусь умову, що вказувала на наявність помилки, і в залежності від цього робити ті чи інші дії. В Java з’явилося більш просте рішення - обробка виняткових ситуацій.

```
try{
   someAction();
   anotherAction();
} catch(Exception e) {
   // обробка винятку
}
```

При виникненні виняткової ситуації управління передається від коду, що викликав виняткову ситуацію, на найближчий блок catch (або вгору по стеку) і створюється об'єкт, успадкований від класу Throwable, або його нащадків який містить інформацію про виняткову ситуацію і використовується при її обробці. Власне, в блоці catch вказується саме клас оброблюваної ситуації.

Допускається створення власних класів виняткових ситуацій. Здійснюється це за допомогою механізму успадкування, тобто клас виняткової ситуації, створений користувачем повинен бути успадкований від класу Throwable, або його нащадків.

Конструкція try-catch в загальному випадку виглядає так:

```
try {
   ...
} catch(SomeExceptionClass e) {
   ...
} catch(AnotherExceptionClass e) {
   ...
}
```

Спочатку виконується код в фігурних дужки після оператора try. Якщо під час його виконання не відбувається ніяких позаштатних ситуацій, то далі управління передається

за закриваючу фігурну дужку останнього оператора catch, асоційованого з даним оператором try. Якщо в межах try виникає виняткова ситуація, то далі виконання коду проводиться по одному з перерахованих нижче сценаріїв:

- Якщо виникла виняткова ситуація, клас якої вказаний як параметр одного з блоків catch, то виконується блок коду, асоційований з даним catch. Далі, якщо код в цьому блоці завершується нормально, то і весь оператор try завершується нормально і управління передається на оператор після закриваючої фігурної дужки останнього catch.
- Якщо код в catch завершується некоректно, то і весь try завершується некоректно з тієї ж причини.
- Якщо виникла виняткова ситуація, клас якої не зазначений в якості аргументу ні в одному catch, то виконання всього try завершується некоректно.

Конструкція **try-catch-finally** дозволяє за допомогою оператора finally гарантувати виконання будь-якого фрагмента коду, незалежно від того виникла виняткова ситуація чи ні. Послідовність виконання такої конструкції наступна:

- якщо оператор try виконаний нормально, то буде виконаний блок finally. У свою чергу, якщо оператор finally виконується нормально, то і весь оператор try виконується нормально.
- Якщо під час виконання блоку try виникає виняток і існує оператор catch, який перехоплює даний тип винятку, відбувається виконання пов'язаного з catch блоку. Якщо блок catch виконується нормально, або ненормально, все одно потім виконується блок finally. Якщо блок finally завершується нормально, то оператор try завершується так само, як завершився блок catch.
- Якщо в списку операторів catch чи не знаходиться такого, який обробив б виникло виняток, то все одно виконується блок finally. У цьому випадку, якщо finally завершиться нормально, весь try завершиться ненормально з тієї ж причини, по якій було порушено виконання try.

У всіх випадках, якщо блок finally завершується ненормально, то весь try завершиться ненормально з тієї ж причини.

```
try {
   byte [] buffer = new byte[128];
   FileInputStream fis = new FileInputStream("file.txt");
   while(fis.read(buffer) > 0) {
      ... опрацювання даних ...
   }
} catch(IOException es) {
      ... опрацювання винятку ...
} finally {
   fis.flush();
   fis.close();
}
```

Якщо в даному прикладі помістити оператори очищення буфера і закриття файлу відразу після закінчення обробки даних, то при виникненні помилки вводу / виводу коректного закриття файлу не відбудеться

Завдання. Скласти Java-програму розв'язання наступної задачі: Задано цілочисельний масив (матриця) А розмірності 5х5. Елементи вхідного масиву А записані в текстовому файлі на диску (файл створити самостійно за допомогою текстового редактора, заповнити довільно – 5 рядків по 5 чисел). Потрібно:

прочитати елементи масиву А із відповідного текстового файлу;

в масиві А знайти вказані у варіанті завдання величини та вивести результат у вигляді повідомлення на консоль;

утворити новий масив В, згідно з вказівками відповідного варіанту, утворений масив вивести у новий текстовий файл на диску;

В програмі передбачити опрацювання виняткових ситуацій: відсутність файлу з масивом А, неправильний запис елементів масиву в файлі, недостатня кількість елементів у файлі та помилки при записі у файл масиву В. В якості обробки виняткових ситуацій реалізувати вивід повідомлення із текстом по відповідну помилку на консоль.

Варіанти завдань

1. В масиві А знайти кількість, суму та середнє значення додатних елементів. Масив В утворити з масиву А, замінивши в ньому всі додатні елементи числом 5, а елементи, менші за -5 – середнім значенням додатних елементів.
2. В масиві А знайти різницю середніх значень окремо взятих додатних та від'ємних елементів. Масив В утворити з масиву А, замінивши в ньому всі елементи менші за -5 на протилежні.
3. В масиві А знайти кількості елементів, значення яких лежать в діапазонах від 1 до 10, від 11 до 25 та від 1 до 20. Масив В утворити з масиву А, замінивши в ньому всі елементи на протилежні.
4. В матриці А знайти добуток мінімальних елементів кожного рядка. Матрицю В утворити транспонуванням матриці А.
5. В матриці А знайти суму середніх значень елементів рядків. Матрицю В утворити з матриці А, замінивши всі елементи під головною діагоналлю знайденою сумою.
6. В матриці А знайти відношення кількості додатних елементів до кількості від'ємних. Матрицю В утворити з матриці А, відобразивши її симетрично відносно середнього рядка.
7. В матриці А знайти кількості додатних елементів у кожному рядку. Матрицю В утворити з матриці А, видаливши з неї перший стовпчик, змістивши решту її стовпців на один вліво та додавши останнім стовпцем знайдені кількості додатних елементів рядка.
8. В матриці А знайти суму елементів, розташованих над головною діагоналлю та суму елементів, розташованих під нею. Матрицю В утворити з матриці А, замінивши елементи головної діагоналі різницею знайдених сум.
9. В матриці А знайти відношення мінімального з додатних елементів до максимального з від'ємних. Матрицю В утворити з матриці А перестановкою її стовпців в зворотному порядку.
10. В масиві А знайти відношення значень максимального та мінімального елементів. Масив В утворити з масиву А, замінивши в ньому всі елементи більші за 10 на максимальний елемент, а від'ємні елементи на 0.
11. В матриці А знайти відношення середнього значення елементів, розташованих над головною діагоналлю до середнього значення елементів цієї діагоналі. Матрицю В утворити з матриці А перестановкою її рядків в зворотному порядку.
12. В матриці А знайти середнє арифметичне елементів, розташованих над головною діагоналлю та середнє елементів, розташованих під нею. Матрицю В утворити з матриці А, відобразивши її симетрично відносно другорядної діагоналі.

13. В матриці А знайти кількості додатних елементів у кожному стовпці. Матрицю В утворити з матриці А, видаливши з неї перший рядок, змістивши решту її рядків на один вгору та додавши останнім рядком знайдені кількості додатних елементів стовпця.
14. В матриці А знайти відношення кількості нульових елементів до кількості ненульових. Матрицю В утворити з матриці А, відобразивши її симетрично відносно середнього стовпця.
15. В матриці А знайти суму середніх значень елементів стовпців. Матрицю В утворити з матриці А, замінивши всі елементи над головною діагоналлю знайденою сумою.
16. В масиві А знайти розмах значень його елементів (різницю значень максимального та мінімального елементів). Масив В утворити з масиву А, замінивши в ньому всі елементи більші за 5 на мінімальний елемент, а елементи, менші за -5 – на максимальний.

Приклад виконання лабораторної роботи.
Варіант 16. В масиві А знайти розмах значень його елементів (різницю значень максимального та мінімального елементів). Масив В утворити з масиву А, замінивши в ньому всі елементи більші за 5 на мінімальний елемент, а елементи, менші за -5 – на максимальний.

Код програми програми для розв'язання задачі може бути таким:

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.InputMismatchException;
import java.util.NoSuchElementException;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Lab8 {

public static void main(String[] args) {
	FileReader fA;
	File fB = new File("D:\\B.txt");
	int[][] A = new int[5][5];
	int[][] B = new int[5][5];
	try {
		fA = new FileReader("D:\\A.txt");
		Scanner input = new Scanner(fA);
		for(int i=0;i<5;i++)
			for(int j=0;j<5;j++)
				A[i][j] = input.nextInt();
		input.close();
	} catch (FileNotFoundException e) {
		System.out.println("Файл з елементами масиву А не створено");
		return;
	} catch (InputMismatchException e) {
		System.out.println("Перевірте запис елементів масиву А у файлі");
		return;
	} catch (NoSuchElementException e) {
		System.out.println("У файлі недостатньо елементів для масиву 5х5");
		return;
```

```
}
	int max=A[1][1];
	int min=A[1][1];
	for(int i=0;i<5;i++)
		for(int j=0;j<5;j++){
			if (max < A[i][j]) max = A[i][j];
			if (min > A[i][j]) min = A[i][j];
		}
	System.out.println("Розмах значень елементів масиву A: " + (max- min));

	for(int i=0;i<5;i++)
		for(int j=0;j<5;j++)
			if (A[i][j]<-5) B[i][j] = max;
				else {
					if(A[i][j]>5) B[i][j] = min;
					else B[i][j]= A[i][j];
				}
	try {
		FileWriter fileWrite = new FileWriter(fB);
		BufferedWriter writeBufer = new BufferedWriter(fileWrite);
		PrintWriter printB = new PrintWriter(writeBufer);
		for(int i=0;i<5;i++){
			for(int j=0;j<5;j++) printB.print(B[i][j] + " ");
			printB.println();
		}
		printB.close();
	} catch (IOException e) {
		System.out.println("Помилка при створенні файла з масивом B");
	}
	}
}
```

Основні елементи запропонованого коду:
На початку тіла методу main оголошено масиви, та файли для читання / запису даних. Файл для виводу масиву B можна оголошувати відразу, при фізичній відсутності цього файла на диску його буде створено, оскільки в цей файл буде виконуватися запис. Відсутність файла для зчитування масиву A призведе до помилки, тому створення FileReader-a fA на основі дискового файла D:\\A.txt виконується всередині блоку try разом з читанням елементів масиву A.
Для читання даних з файла використано сканер input та метод nextInt() класу Scanner, про який йшлося в теоретичних відомостях.
Три блоки catch охоплюють найбільш поширені при читанні з файлів види винятків. Послідовність в якій вони записані, очевидно, є суттєвою. Кожен catch завершується оператором return, що завершує виконання методу main – при виникненні хоча б однієї з описаних помилок подальше виконання методу є безглуздим.
У випадку успішного зчитування масиву A метод main продовжить виконання з коду, записаного після закриваючої фігурної дужки останнього catch. Там за допомогою вкладених циклів for реалізовано опрацювання масиву A та утворення масиву B, задекларованe в умові задачі.
Завершується код методом створенням потрібних для запису буферів, та власне записом елементів масиву B у файл. Перехоплення виняткової ситуації в цій частині коду не обов'язкове – помилки при записі даних у файл виникають значно рідше. Хоча і цілком можливі, наприклад запис файла на диск до якого користувач не має доступу і ін.

Лабораторна робота № 9

Створення структурованих типів даних на основі класів.

Мета роботи: оволодіння навичками опису класів та використання об'єктів для зберігання та опрацювання структурованих даних.

Завдання . Описати клас для зберігання даних про заданий тип об'єктів у заданій предметній області. При описі класу дотримуватися принципу інкапсуляції – усі поля з даними повинні бути закритими, а для роботи з даними, що містяться у цих полях, реалізувати відповідні методи. Для створення екземплярів класу реалізувати відкриті конструктори, що заповнюють поля об'єктів даними. Реалізувати перегрузку одного – двох методів класу та конструктора класу. Для демонстрації функціонування класу створити масив з 10-15 примірників класу. Для створеного масиву реалізувати:
а) вивід на консоль даних з усіх елементів масиву;
б) вивід даних лише тих елементів, які відповідають заданому у варіанті завдання критерію;
в) пошук в масиві та вивід на консоль об'єктів з вказаною властивістю (згідно до варіанта).

Варіанти завдань

1. Предметна область: дослідна лабораторія, клас: обладнання для дослідів, орієнтовний перелік полів: назва обладнання, кому видано, кількість, видано-повернуто, дата видачі. Вивести окремо список наявного та окремо список виданого обладнання. Реалізувати пошук обладнання за назвою.
2. Предметна область: оренда житла, клас: помешкання, орієнтовний перелік полів: адреса, кількість кімнат, вартість оренди, помешкання орендовано (так/ні), дата оренди, термін оренди. Вивести окремо список вільних та окремо список орендованих помешкань. Реалізувати пошук вільних помешкань з вказаною кількістю кімнат та допустимою вартістю оренди.
3. Предметна область: автостоянка, клас: автомобіль, орієнтовний перелік полів: марка автомобіля, прізвище та адреса власника, номерний знак, номер місця на стоянці, наявність на стоянці, дата та час заїзду чи виїзду зі стоянки. Вивести окремо список автомобілів, присутніх на стоянці, та автомобілів, що виїхали зі стоянки. Реалізувати пошук автомобіля за вказаним номерним знаком.
4. Предметна область: фільмотека, клас: фільм, орієнтовний перелік полів: назва фільму, рік випуску, перелік акторів у головних ролях, формат файлу, розмір в мб, можливість безкоштовного завантаження. Вивести окремо список фільмів у різних форматах. Реалізувати пошук фільмів за прізвищем актора.
5. Предметна область: інтернет-магазин, клас: товар, орієнтовний перелік полів: назва товару, категорія, опис товару, ціна за одиницю, наявність на складі, кількість, дата поставки. Вивести окремо список відсутніх товарів. Реалізувати пошук товарів за вказаною категорією.
6. Предметна область: прохідна підприємства, клас: працівник, орієнтовний перелік полів: прізвище, посада, відділ, номер посвідчення, присутній на території чи вийшов за межі, час останнього входу чи виходу. Вивести окремо список працівників, присутніх на території, та відсутніх. Реалізувати пошук даних працівника за вказаним прізвищем.
7. Предметна область: оренда автомобілів, клас: автомобіль, орієнтовний перелік полів: марка автомобіля, номерний знак, рік випуску, автомобіль орендовано (так/ні), дата оренди, термін оренди. Вивести окремо список вільних та окремо список орендованих автомобілів. Реалізувати пошук автомобілів, що звільняться з оренди у вказаному місяці.

8. Предметна область: банківські послуги, клас: картковий рахунок, орієнтовний перелік полів: номер картки, прізвище клієнта, адреса клієнта, сума на рахунку, кредит/депозит, дата видачі, термін дії. Вивести окремо список кредитних та окремо список депозитних карток. Реалізувати пошук даних за номером картки.
9. Предметна область: служба доставки, клас: замовлення, орієнтовний перелік полів: назва товару, адреса для доставки, вартість, кількість, проведення попередньої оплати, дата та час замовлення. Вивести окремо список оплачених та неоплачених замовлень. Реалізувати пошук замовлення за вказаною адресою, або її частиною.
10. Предметна область: відділ кадрів, клас: працівник, орієнтовний перелік полів: прізвище, домашня адреса, посада, рік народження, кількість дітей, дата прийому на роботу, постійний працівник/ працівник за сумісництвом. Вивести окремо список постійних працівників та окремо список сумісників. Реалізувати пошук даних працівника за вказаною посадою та допустимим стажем роботи.
11. Предметна область: готель, клас: готельний номер, орієнтовний перелік полів: опис номера, кількість місць, вартість проживання, вільний/зайнятий, дата заселення, термін проживання. Вивести окремо список вільних та окремо список зайнятих номерів. Реалізувати пошук вільних помешкань з вказаною кількістю місць та допустимою вартістю проживання.
12. Предметна область: інтернет-магазин, клас: замовлення, орієнтовний перелік полів: назва товару, адреса для доставки, ціна за одиницю, кількість, проведення попередньої оплати, дата замовлення. Вивести окремо список оплачених та неоплачених замовлень. Реалізувати пошук замовлення за вказаною адресою, або її частиною.
13. Предметна область: автотранспортне підприємство, клас: автомобіль, орієнтовний перелік полів: марка автомобіля та номерний знак, прізвище водія, вантажопідйомність, автомобіль в рейсі (так/ні), дата виїзду в рейс, тривалість рейсу в днях. Вивести окремо список автомобілів, вільних для перевезень, та автомобілів, що виїхали у рейс. Реалізувати пошук вільних автомобілів з достатньою для вказаного вантажу вантажопідйомністю .
14. Предметна область: банківські послуги, клас: операція з картковим рахунком, орієнтовний перелік полів: номер картки, номер терміналу, адреса терміналу, час, сума операції, прихід/розхід. Вивести окремо список прихідних та окремо список розхідних операцій. Реалізувати пошук даних про операції за вказаним номером терміналу.
15. Предметна область: прийом на роботу, клас: потенційний працівник, орієнтовний перелік полів: прізвище, домашня адреса, спеціальність, рік народження, рейтинг за співбесідою, запрошення надіслано (так/ні), дата запрошення. Вивести окремо список претендентів, яким надіслано запрошення. Реалізувати пошук даних претендентів за вказаною спеціальністю та рівнем рейтингу.
16. Предметна область: бібліотека, клас: книга, орієнтовний перелік полів: назва, автор, рік видання, кількість сторінок, наявна у фонді/видана читачеві дата видачі. Вивести окремо список наявних та окремо список виданих читачам книжок. Реалізувати пошук книжок за прізвищем автора.

Приклад.

Варівнт 16. Предметна область: бібліотека, клас: книга, орієнтовний перелік полів: назва, автор, рік видання, кількість сторінок, наявна у фонді/видана читачеві дата видачі. Вивести окремо список наявних та окремо список виданих читачам книжок. Реалізувати пошук книжок за прізвищем автора.

Створюємо новий проект. В нього додаємо клас Book (без головного методу main), в якому описуємо структуру та методи вказаного об’єкта (див. коментарі до коду):

```java
package lab9;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;


public class Book {

       /*
        Нижче описано поля класу, що використовуватимуться для зберігання
        даних кожного окремого екземпляра
        Дотримуючись інкапсуляції всі поля робимо закритими (private)
        їх читання чи зміни реалізуємо спеціальні методи get-ери  і set-ери
        В зміні автора, назви, року і к-сті сторінок немає сенсу, оскільки
        в такому випадку слід створювати новий екземпляр класу "книга".
        Поля у яких немає методу для зміни значення прийнято помічати
        як final
        */

      private final String title;// назва книги
    private final String author;// автор або декілька авторів
    private final int year;// рік видання
    private final int countOfPages;//кількість сторінок
    private boolean available;//наявність книги в бібліотеці
    //книга може бути видана користувачеві - false
    //або зберігатися у фонді - true
    private Date issueDate;//дата видачі, має сенс лише при available == false
    //якщо книга є у фонді issueDate = Null

    // методи-геттери для усіх полів

    public String getTitle(){
        return title;
    }
    public String getAuthor(){
        return author;
    }
    public int getYear(){
        return year;
    }
    public int getCountOfPages(){
        return countOfPages;
    }
    public boolean isAvailable(){
        return available;
    }
    public String getIssueDate() throws ParseException{
        if(issueDate!=null){
            SimpleDateFormat dateFormat = new  SimpleDateFormat("dd.MM.yy");
            return dateFormat.format(issueDate);
        }
        else return "";
    }
    // методи-сеттери змінюють лише наявність книги у фонді
    // та дату видачі користувачеві, причому зміна повинна
    // обидва поля одночасно
    public void setIssue() {
      //видача книги зареєстрована поточною датою
        if(available){
```

```
            available = false;
            issueDate = new  Date();
        }
    }
    public final void setIssue(String date) throws ParseException{
        //видача книги за вказаною датою
      if(available){
            available = false;
            issueDate = new  SimpleDateFormat("dd.MM.yy").parse(date);
        }
    }
    public void setReturn() {
        if(!available) available = false;
            issueDate = null;
    }

    //Конструктори з використанням перегрузки
    public Book(String author, String title, int year, int n){
        //основний конструктор
      this.author = author;
        this.title = title;
        this.year = year;
        countOfPages = n;
        available = true;
        issueDate = null;
    }
     public Book(String author, String title, int year, int n, String date) throws
ParseException{
        //конструктор із вказаною датою видачі книги користувачеві
      //суперечить логіці предметної області, але дозволяє спростити
      //тестування можливостей класу (див. головний клас проекту)
      this(author, title, year, n);
        this.setIssue(date);
        this.available = false;
    }

    public Book(String author, String title, int n){
        //конструктор для книг, що вийшли з друку в поточному році
      //перегрузка спрощує реєстрацію книжки
      this.author = author;
        this.title = title;
        countOfPages = n;
        available = true;
        issueDate = null;
        Date now = new Date();
        SimpleDateFormat  nowFormat = new SimpleDateFormat("yyyy");
        year = Integer.parseInt(nowFormat.format(now));
    }
       @Override
    public String toString(){
        //первизначений метод toString дозволяє зручно виводити інформацію,
      //яку зберігає екземпляр класу, на консоль
        return  author + " " + title + " - " + year + ", " + countOfPages + " с.";
    }
}
```

Для демонстрації можливостей створеного класу, та виконання додаткових завдань з варіанта слід створити принаймні ще один клас з методом main. В поданому нижче коді використано два класи: клас Library об’єднує масив книг (екземплярів класу Book)

методами для обробки цього масиву: виводом елементів та пошуком за заданим критерієм.

```
package lab9;

public class Library {

    private Book[] books;

    public Library(Book[] list){
        //відкритий конструктор приймає посилання на масив з даними
      books = list;
    }
    /* використані нижче цикли з ітератором
     * for (Book book : books)
     * можна записувати і в традиційній формі
     *  for (int i = 0; i<books.length; i++)
     * але подана нижче форма надає коду більшу гнучкість,
     * оскільки придатна для роботи з колекціями
     * такими, як, наприклад, множини і поданий код допускає
     * в перспективі заміну масиву books на множину books
     * без жодних затрат на його модифікацію
     */

    public void printList(){
        //друк всіх елементів масиву
      for (Book book : books) {
            System.out.println(book);
        }
    }

    public void printList(boolean isAvailable){
        //друк наявних/виданих книг
      for (Book book : books) {
            if (book.isAvailable() == isAvailable)System.out.println(book);
        }
    }

    public String findBookByAuthor (String author){
        //пошук елементів, що містять у полі author заданий фрагмент

      String result="";
       for (Book book : books) {
           if (book.getAuthor().contains( author))
           result += book.toString()+ "\n"+
                   (book.isAvailable()?" в наявності":" видана") + "\n";
        }
         return  result.isEmpty()?"нічого не знайдено":result;
    }
}
```

В методі main головного класу створено і проініціалізовано статичними даними масив книг, на основі якого створюється примірник класу Library. Далі продемонстровано роботу усіх реалізованих в класі Library методів.

```
package lab9;

import java.text.ParseException;
import java.util.Scanner;


public class Lab9 {
    public static void main(String[] args) throws ParseException {

    Library myLibrary;
    Book [] books = {new Book("Ткаченко  О.М.",
            "Комп'ютерне  програмування  на  мові  Java.", 2013, 147),
                        new Book("Копитко М.Ф., Іванків К.С. ",
            "Основи програмування мовою Java: Тексти лекцій.", 2002, 83, "25.12.14"),
                        new Book("Р. А. Мельник, М. М. Сенів",
                                "Програмування на JAVA: метод. " +
            "Вказівки до лаборатор. робіт з дисципліни Об'єкт.-орієнт. програмув.",
                                                                    2007,42),
                        new Book("Брюс Эккель", "Философия Java", 1168),
                        new Book("Горбань А.Г.", "Програмування в Java", 2008,
                                                                310, "03.03.15"),
                        new Book("И.Н.Блинов, В.С.Романчик.",
                                "Java. Методы программирования", 2013, 896),
                        new Book("И.Н.Блинов, В.С.Романчик.",
            "Java. Промышленное программирование", 2007, 704,"01.02.15"),
                        new Book("Герберт Шилдт",
                                "Java. Полное руководство", 2012, 1104),
                        new Book("Брнакевич І.Є., Вагін П.П.",
                               "Програмування мовою Java: використання " +
                                "фундаментальних класів: Тексти лекцій.",2002,
                                                                 75,"14.12.14"),
                        new Book("Ментинський С.М.",
            "Використання класів для опрацювання структур даних. " +
                                "Завдання до лабораторних робіт № 9-11.",36)};

    myLibrary = new Library(books);

    System.out.println("Список книг у бібліотеці:");
    myLibrary.printList();

    System.out.println("Список книг виданих у користування:");
    myLibrary.printList(false);

    System.out.println("Список книг що залишилися у книгосховищі:");
    myLibrary.printList(true);

    Scanner input = new Scanner(System.in);
    System.out.println("Пошук книг. Введіть прізвище автора");
    String author = input.next();
    input.close();

    String authorBooks = myLibrary.findBookByAuthor(author);
    System.out.println(authorBooks);

    }
  }
```

Лабораторна робота № 10

Серіалізація об’єктів, опрацювання структурованих файлів.

Мета роботи: знайомство з потоками даних та серіалізацією об’єктів.

Завдання . Вдосконалити об’єктну модель попередньої лабораторної роботи, додавши до неї наступну функціональність методи для запису об’єктів з масиву з даними у файл і навпаки.

Варіанти завдань
Варіанти завдань відповідають лабораторній роботі № 9.

Приклад.

Варіант 16. Предметна область: бібліотека, клас: книга, орієнтовний перелік полів: назва, автор, рік видання, кількість сторінок, наявна у фонді/видана читачеві дата видачі.

В класі Library створюємо метод для запису масиву книг у файл. Для цього у класі Book слід реалізувати інтерфейс `Serializable`:

```
public class Book implements Serializable{
. . .
}
```

Метод запису в класі Library можна описати так:

```
public void saveToFile(String fileName){
      File file = new File(fileName);

      try {
                  FileOutputStream fileOut =  new FileOutputStream(file);
                  ObjectOutputStream outStream = new ObjectOutputStream(fileOut);
                  for(Book book: books){
                        outStream.writeObject(book);
                  }
                  outStream.close();
            } catch (FileNotFoundException e) {
                  // методи запису у файл вимагають обробки винятків
                  System.out.println("Файл для запису даних не створено - " +
                              e.getMessage() );

            } catch (IOException e) {

                  System.out.println("Помилка запису даних - " + e.getMessage() );
            }
}
```

При цьому слід додати відповідні імпорт-вирази для використаних класів у початок коду:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
```

При читанні даних з файлу можуть виникати різні помилки, тому операції з відкриття файлів та читання даних слід поміщати в блоки try{…}cath(){…}. Для цього рекомендується скористатися підказками IDE Eclipse.

Перевірку роботи методу можна виконати на масиві книжок, створеному у попередній лабораторній роботі за рахунок ініціалізації. При вдалому виконанні методу отримаємо готовий файл з даними, який можна буде використати для тестування наступного методу.

Метод зчитування даних може виглядати так:

```
public void loadFromFile(String fileName){
      File file = new File(fileName);
      int i = 0;
      Book book=null;
      ObjectInputStream inStream = null;
      try {
                  FileInputStream fileIn =  new FileInputStream(file);
                  inStream = new ObjectInputStream(fileIn);
                  if(inStream != null){
                  while(true){
                        book = (Book) inStream.readObject();
                        if(book==null)break;
                        books[i++] = book;
                        //читання даних з файлу в масив
                  }
                  inStream.close();
            }

      } catch (FileNotFoundException e) {
            System.out.println("Не знайдено файл з даними - " + e.getMessage() );
            }catch (ClassNotFoundException e) {
            System.out.println("Помилка в структурі даних - " + e.getMessage() );
            } catch (IOException e) {
            System.out.println("Помилка читання даних - " + e.getMessage() );
            }
}
```

Очевидно, що реалізація цього методу вимагатиме імпорту відповідних класів із стандартних пакетів, та опрацювання виняткових ситуацій, що можуть траплятися при читанні з файлу.

Для перевірки читання даних з файлу слід створити порожній масив книг books, достатнього для записаних даних розміру:

```
Book [] books = new Book[15];
Library myLibrary = new Library(books);
myLibrary.loadFromFile("C:\\Books.dat");
System.out.println("Список книг у бібліотеці:");
myLibrary.printList();
```

Очевидно, що частина масиву при цьому залишиться незаповненою, оскільки масив має фіксований розмір, програма наперед не може точно визначити скільки даних міститься у файлі. В таких випадках зручніше користуватися динамічними масивами – колекціями (див завдання б) до наступної лабораторної роботи).

Лабораторна робота № 11

Використання параметризованих колекцій для роботи з об'єктами.

Мета роботи: удосконалення навичок ООП та знайомство з колекціями Java.

Завдання . Вдосконалити об'єктну модель попередньої лабораторної роботи, додавши до неї наступну функціональність:

a) Реалізувати в класі з даними метод compareTo для порівняння двох примірників класу (), за допомогою якого виконати сортування масиву об'єктів в головному класі. Правила порівняння двох об'єктів розробити самостійно, відповідно до предметної області.

b) Замінити масив об'єктів, що використовується для тимчасового зберігання даних на динамічну структуру – список ArrayList. Протестувати роботу програми прочитавши дані з файла створеного в попередній лабораторній роботі, додавши 2-3 нових об'єкти і записавши файл заново.

Варіанти завдань

Варіанти завдань відповідають лабораторній роботі № 9.

Приклад.

Варіант 16. Предметна область: бібліотека, клас: книга, орієнтовний перелік полів: назва, автор, рік видання, кількість сторінок, наявна у фонді/видана читачеві дата видачі.

а) Реалізуємо порівняння двох книжок за правилом:

більшою вважається книжка, що вийшла з друку пізніше – метод compareTo повертає ціле число, тому різниця років видання книжок може бути його результатом;

якщо ж книжки видані в одному році, то порівняємо їх за алфавітним порядком прізвищ авторів – тут скористаємося власним методом compareTo класу String, що здійснює порівняння фрагментів тексту за алфавітним порядком;

нарешті, якщо книжки видані одним і тим же автором чи групою авторів протягом року, то порівнюватимемо книжки за назвою – таке потрійне порівняння дасть результат 0 лише у випадку, якщо йдеться про одну і ту ж книжку.

Орієнтовний код методу:

```
@Override
    public int compareTo(Book another){
      /*метод порівняння для впорядкування книжок
      *більшими вважаємо книжки, видані пізніше
      *якщо книжки видані в одному році - порівнюємо
      *за алфавітом прізвища авторів, а далі - назви книжок
      */
            int result = year - another.getYear();
      if(result==0) result = author.compareTo(another.author);
      if(result==0) result = title.compareTo(another.title);
      return result;

      }
```

Оскільки метод compareTo описаний в батьківському класі усіх об'єктів Object, то перед його описом додаємо анотацію @Override, яка вказує, що ми перевизначаємо батьківський метод, а в заголовку класу додаємо реалізацію інтерфейсу Comparable:

```
public class Book implements Serializable, Comparable<Book>{
```

Для реалізації сортування масиву книг можна скористатися методом сортування масиву з лабораторної роботи № 7, замінивши назву масиву та знак “<” на метод compareTo.

b) Замість масиву

```
private Book[] books;
```

Будемо використовувати список

```
private ArrayList<Book> books;
```

Відповідну структуру даних слід підключити імпорт-виразом

```
import java.util.ArrayList;
```

Наш клас має, конструктор з параметром типу `Book[]`, його слід замінити на `ArrayList<Book>`

Цикли з ітератором виду `for (Book book : books) {. . . }` можна залишити без зміни, а от звертання до елементів масиву за індексом `books[i]`) слід замінити відповідним методом get(i) класу `ArrayList`. Наприклад, метод сортування списку книг виглядатиме так:

```
public void sort(){

      for(int k=0; k<books.size() - 1; k++){

            Book min=books.get(k);
            int iMin = k;
            for(int i=k; i<books.size(); i++){
                  if(min.compareTo(books.get(i))>0){
                        min=books.get(i);
                        iMin = i;
                  }
            }
            books.remove(iMin);
            books.add(k,min);
      }
   }
```

Початкову ініціалізацію масиву, аналогічну до лабораторної 9 слід замінити на додавання елементів до списку:

```
ArrayList<Book> books = new ArrayList<Book>();
      books.add( new Book("Ткаченко  О.М.",
            "Комп'ютерне  програмування  на  мові  Java.", 2013, 147));
      books.add(new Book("Копитко М.Ф., Іванків К.С. ",
          "Основи програмування мовою Java: Тексти лекцій.", 2002, 83, "25.12.14"));
      books.add(new Book("Р. А. Мельник, М. М. Сенів",
                "Програмування на JAVA: метод. " +
       "Вказівки до лаборатор. робіт з дисципліни Об'єкт.-орієнт. програмув.",
                                    2007,42));
```

і т. д.