Лабораторна робота №1

Програмування лінійних алгоритмів.

Стандартні класи і їхні методи у мові Java

1. Створити клас, що має методи для обчислення на ЕОМ значень змінних, що зазначені у таблиці варіантів, за даними розрахунковими формулами і наборами вхідних даних.
2. Доповнити клас методом, що виводить на екран значення вхідних даних і результати обчислень, супроводжуючи вивід найменуваннями виведених змінних.
3. Додати в клас метод, що друкує поточну дату і час у вказаному форматі.
4. Доповнити клас методом введення початкових значень.
5. Створити метод, що вводить дані, обчислює потрібні значення за вказаними формулами, та друкує потрібні результати.
6. Доповнити клас методом main, що є необхідним для використання класу, як автономної програми, та виконати цю програму.

Короткі теоретичні відомості

Огляд структури Java-програми

Всі Java-програми містять в собі 4 основні різновиди будівельних блоків: класи (classes), методи (methods), змінні (variables) і пакети (packages). На якій би мові Ви не програмували раніше, Ви скоріш за все вже добре знайомі з методами, які є не що інше, ніж функції чи підпрограми, та зі змінними, в яких зберігаються дані. З іншого боку, класи пре дставляють собою фундамент об'єктно-орієнтованих властивостей мови. Поки що, для простоти, можна вважати клас деяким цілим, що містить у собі змінні и методи. Нарешті, пакети містять в собі класи і допомагають компілятору знайти ті класи, що потрібні йому для компіляції прикладної програми. Java-програма може містити в собі будь-яку кількість класів, але один з них завжди має особливий статус, і безпосередньо взаємодіє з оболонкою часу виконання. Цей клас називають первинним класом (primary class). Коли програма запускається з командного рядка, системі потрібен тільки один спеціальний метод, що повинен бути присутнім у первинному класі, - метод main. Розглянемо приклад програми мовою Java:

JAVA

```
// імпортування класу LocalDate зі стандартного пакету java.time
import java.time.LocalDate;

public class Main {
    public static void main(String[] S) {
        System.out.println("Hello, Java!");
        LocalDate d = LocalDate.now();
        System.out.println("Date: "+d.toString());
    }
}
```

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

Стандартні типи даних Java

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

Table 1. Примітивні та посилальні типи

Характеристика	Примітивні типи	Посилальні типи
Чи визначені в самій мові Java?	Так	Ні

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

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

Table 2. Примітивні типи мови Java

Тип	Розмір	Діапазон	Приклад
byte	1 байт	від -128 до 127	125
short	2 байти	від -32768 до 32767	-42
int	4 байти	від -2147483648 до 2147483647	19834
long	8 байтів	від -922372036854775808 до 922372036854775807	12345678991
float	4 байти	Залежить від розрядності числа	1.2f
double	8 байтів	Залежить від розрядності числа	123.4
boolean		false, true	true
char	2 байти	Усі символи стандарту Unicode	'z'

Стандартні математичні функції

Оскільки мова Java є об'єктно-орієнтованою, то математичні функції повинні належати до деякого класу. Фактично існують два класи, що визначають математичні операції: Math та StrictMath, Останній призначений для виконання обчислень із "підвищеною точністю", але через поширення вбудованих у процесори математичних модулів, "звичайна" і "підвищена" точність у сучасній Java не розрізняються. Тому найчастіше використовується саме клас Math.

Усі стандартні математичні функції в мові Java є статичними методами класу Math, який визначений з модифікатором `final`, тобто не припускає спадкування. Крім того, клас Math має декілька визначених констант, наведемо дві з них:

Table 3. Константи класу Math

Константа	**Значення**
Math.PI	3.1415926…
Math.E	2.7182818

Основні статичні методи класу Math наведені у наступній таблиці:

Table 4. Математичні функції класу Math

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

Функція – метод	Пояснення
Math.tan(x)	Тангенс x
Math.toDegrees(x)	Переведення кута з радіанів у градуси
Math.toRadians(x)	Переведення кута з градусів у радіани

Примітка. У мові Java є можливість імпорту статичних змінних та методів класу за допомогою директиви `import static` на початку програми. Наприклад:

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

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

Виведення даних у консолі Java-програм

Для виведення інформації на консоль використовуються методи стандартного класу PrintStream:

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

Кожна програма мовою Java містить стандартний об'єкт типу `PrintStream` – `System.out`. Таким чином, виведення інформації на екран буде записуватися як `System.out.print(…)`, `System.out.println(…)`, або `System.out.printf(…)`. Методи `print` та `println` повинні завжди мати один параметр – вираз будь-якого типу, що може бути автоматично приведений до рядкового типу.

Наприклад,

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

Методи `printf` та `format` можуть мати список параметрів, що розділяються комами. Перший параметр – рядок, що містить текст для виведення і форматні шаблони для виведення значень інших параметрів. Наприклад, якщо a=2, b=3

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

Форматні шаблони для виведення звичайних, символьних та числових типів мають наступний синтаксис:

`%[індекс_аргумента$][опції][ширина][.точність]перетворення`

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

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

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

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

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

Table 5. Приклади використання методу printf

Приклади використання `System.out.printf`	**Результат**
System.out.printf("Hello, students");	Hello, students
System.out.printf("Hello, students!\n"); або System.out.printf("Hello, students!%n");	Hello, students!
System.out.printf("Sum %d + %d = %d", a,b,a+b);	Sum 7 + 64 = 71
System.out.printf("Const of E = %16.14f", Math.E);	Const of E = 2,71828182845905

Table 6. Основні типи - символи перетворень

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

Перетворення	Категорія	Опис
'%'	percent	Результатом буде символ '%' ('\u0025')
'n'	line separator	Результатом буде символ, що відокремлює рядки в залежності від платформи.

Дата та час

java.time.LocalDate: LocalDate містить дату без часового поясу в форматі ISO-86011. LocalDate має стандартний формат "РРРР-ММ-ДД", наприклад "2021-12-12".

java.time.LocalTime: LocalTime містить час у форматі ISO-8601 без жодних даних про дату чи часовий пояс. Зазвичай такий формат - "НН: mm: ss", наприклад "11: 12: 25". LocalTime може бути з точністю до наносекунд (після останньої секунди) у форматі "НН: mm: ss.nnnnnnnnn", як у "11: 12: 25.123456789".

java.time.LocalDateTime: представляє дату та час без часового поясу у форматі ISO-8601. Його типовий формат "РРРР-ММ-ДДТГ: мм: сс". (Зверніть увагу на «Т», що відокремлює дні від годин), як у "2021-12-12T11: 12: 25". LocalDateTime також може мати компонент наносекунди, як у LocalTime.

Усі три класи дати-часу мають статичний метод `now()`, який виводить дату (для LocalDate), час (для LocalTime) та дату-час (для LocalDateTime) із системних годинників, на яких запускається JVM.

Все три класса даты и времени предоставляют статический метод now (), который извлекает дату (для LocalDate), время (для LocalTime) и дату-время (для LocalDateTime) из системных часов машины, на которой запущена JVM.

Метод **LocalXXX.of()** приймає як вхідні дані окремі значення, що становлять дату-час, як параметри int. Це значення, що відповідають даті, місяцю, року, годинам, хвилинам, секундам і нагносекундам у форматі int.

Клас `DateTimeFormatter` використовується в Java 8 при форматуванні і розборі датах.

Для створення об'єкта цього класу використовується статичний метод ofPattern (), на вхід якого передається рядок і об'єкт класу `Locale`:

```java
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("MMMM, dd, yyyy HH:mm:ss", Locale.UK);
System.out.println(date.format(formatter));
```

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:

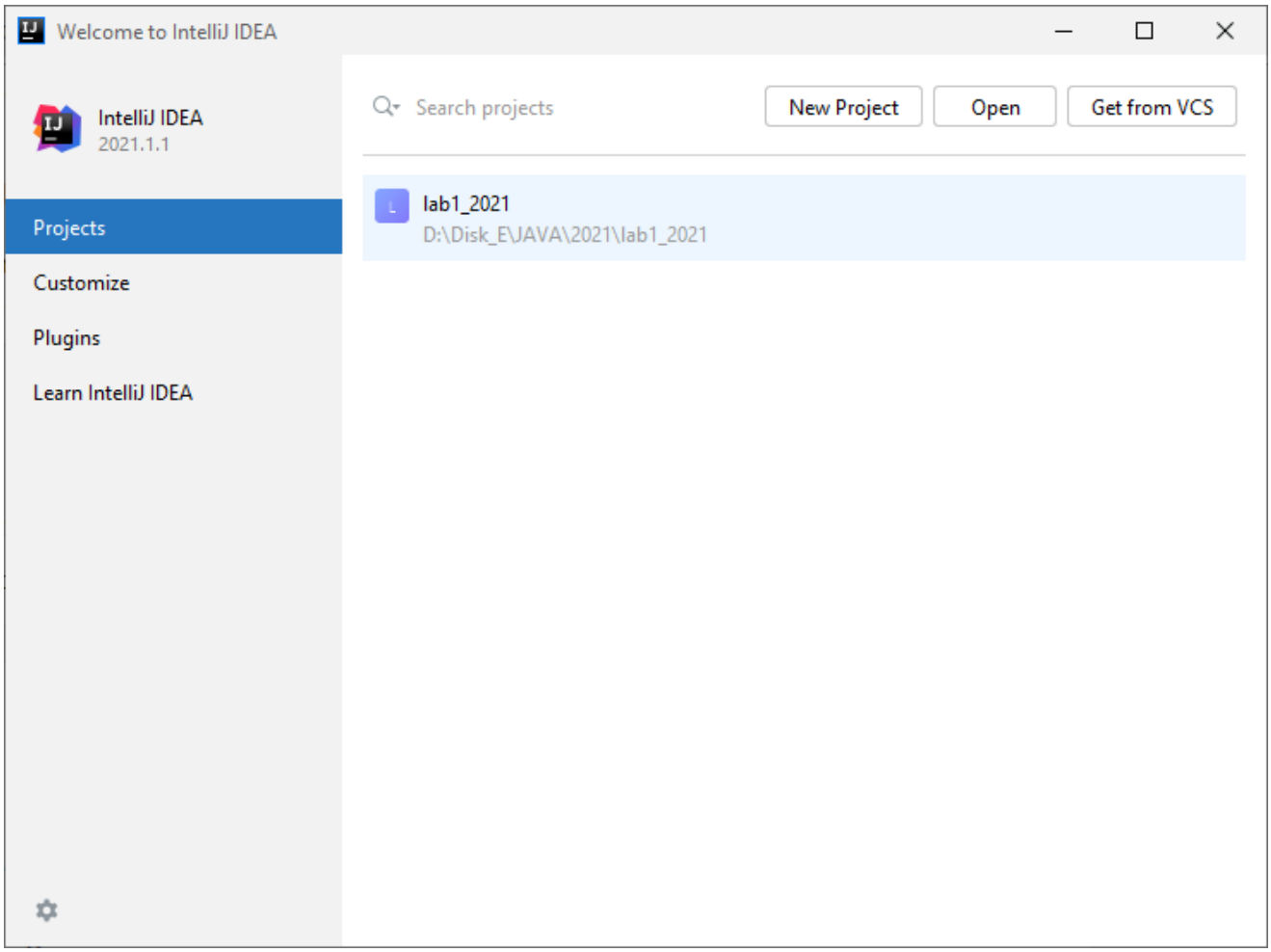
```java
import java.io.*;
import java.util.*;

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

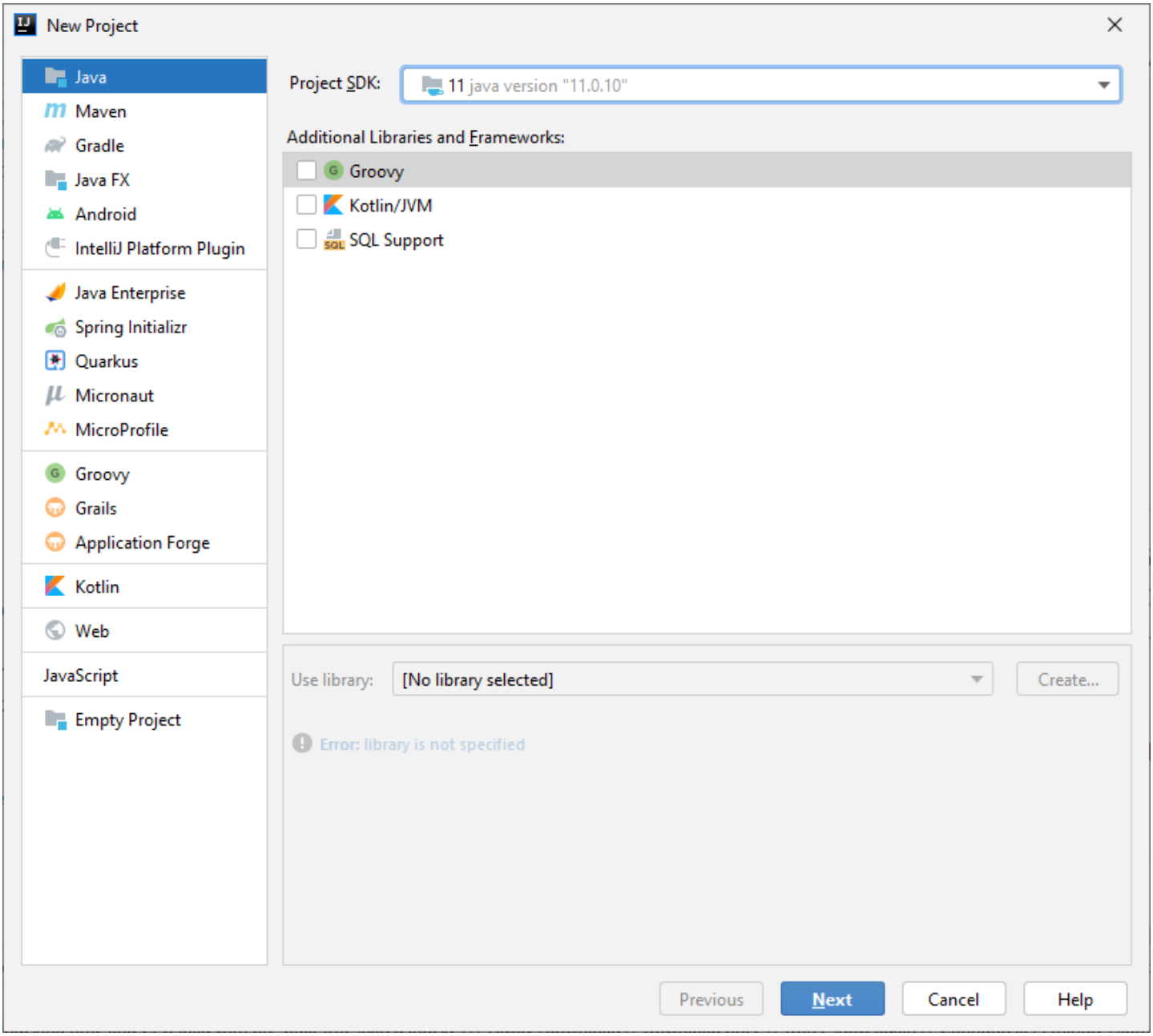
Створення і виконання Java-програм у середовищі IntelliJ IDEA

1. Створіть новий проект, для цього: (якщо у IDE відкрито інший проект, закрийте його – **File → Close Project**)

2. У вікні ”Welcome to IntelliJ IDEA”, що відкриється, оберіть **New Project**

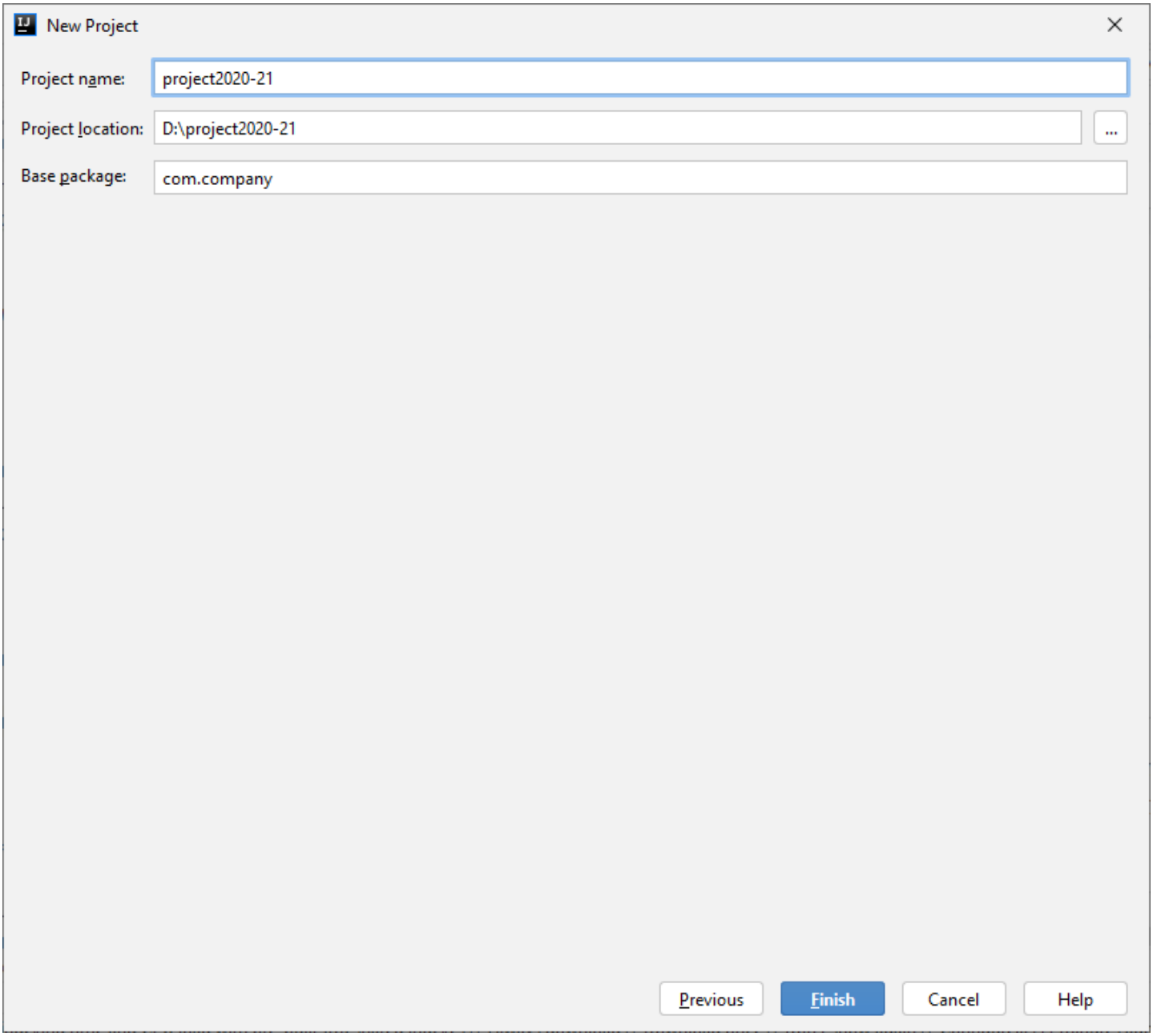


4. У вікні, що відкриється, оберіть категорію **”Java”**, вкажіть потрібний JDK (JDK 11 – якщо потрібно, вкажіть його розташування на диску) та натисніть **Next**.



5. У вікні вибору шаблону проекту просто натисніть **Next**.

6. У вікні вибору місця розташування та ім’я проекту вкажіть необхідні дані, та натисніть **Finish**:



7. У дереві проекту розгорніть вузол з іменем проекту, ПКМ на вузлі **"src"** та у меню, що розкриється виберіть **New → Java class**
8. Вкажіть його ім'я та натисніть **OK**
9. У полі текстового редактора відредагуйте код Вашої програми.

Для першого запуску використайте ПКМ на імені класу у дереві проекту і виберіть **Run <ім'я вашого класу>.main()** або натисніть **Ctrl+Shift+F10**

Для наступних запусків програми можна скористатись сполученнями клавіш **Shift+F10 (Run Main)** або **Ctrl+F5 (Rerun Main)**

Приклад програми на Java

JAVA

```java
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class Main {

    public static void main(String[] args) {
        Main prog = new Main();
        prog.run();
    }

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

    private void print(int x, int y) {
        System.out.println("x="+x);
        System.out.println("x^2="+y);

    }

    private void printDate() {
        LocalDateTime date=LocalDateTime.now();
        System.out.println(date);
        System.out.println(date.format(DateTimeFormatter.ofPattern("dd  MMMM uuuu HH:mm")));
    }

    private void run() {
        int x = 5;
        int y = calcSquare(x);
        print(x,y);
        printDate();
    }

}
```

Варіант	Розрахункові формули	Значення вхідних даних	Формат дати і часу
1	$R = x^3(x+1)/b - sin^2(x+a);\ s = \sqrt{\frac{\lvert xb\rvert}{a^{2/5}}} + cos(x+b)^2$	a=0.7 b=0.05 x=0.5	Дата у форматі рр-мм-дд
2	$c = \lvert e^{y/x} - \sqrt[5]{y/x}\rvert; f = (y-x)\frac{y - z/(y-x)}{1+(y-x)^2}$	x=1.825 y=18.025 z=-3.298	Місяць, день, рік та день тижня
3	$f = \sqrt[3]{m \cdot tgt + \lvert csint\rvert};\ z = mcos(\frac{b}{t}sint) - c$	m=2;c=-1 t=1.2 b=0.7	Дата і час з точністю до мілісекунд
4	$y = btg^2x + ae^{-\sqrt{a}};\ d = cos(bx/a) - \frac{a}{sin^2(x/a)}$	a=3.2 b=17.5 x=-4.8	Час у форматі гг:хх:сс

Варіант	Розрахункові формули	Значення вхідних даних	Формат дати і часу
5	$a = \frac{2cos(x+\pi/6)}{3/2 - sin^2 y};\ b = 1/e^z + \frac{z^2}{3 + z^2/5}$	x=1.426 y=-1.220 z=3.5	Дата у форматі дд-мм-рр
6	$s = 1 - x + \frac{x^2}{2} - \frac{x^3}{6} + \frac{x^4}{24};\ f = x(sinx^3 + cos^2 y)$	x=0.135 y=0.225	День тижня та час
7	$s = x^3 tg^2(x+b) + \frac{a}{\sqrt{x+b}};\ Q = \frac{bx^2 - \sqrt[3]{a}}{e^{ax} - 1}$	a=16.5 b=3.4 x=0.61	Дата у форматі дд міс рррр
8	$y = e^{-bt} sin^2(at - b) + \sqrt{\lvert bt - a\rvert};\ s = sin(a^2 t^2 cos3t) - \frac{b}{2}$	a=-0.5 b=1.7 t=0.44	День тижня число і місяць
9	$y = sin^3(x^2 + a)^2 - \sqrt{\frac{x}{b}};\ z = \frac{x^2}{a} + cos(x+b)^3$	a=1.1 b=0.004 x=0.2	Дата у форматі дд місяць рррр
10	$w = \sqrt{x^2 + b} - b^2 sin^3(x+a)/x;\ y = cos^2 x^3 - \frac{x}{\sqrt{a^2 + b^2}}$	a=1.5 b=15.5 x=-2.8	Час у форматі гг:хх та дата місяць рр