МІЖРЕГІОНАЛЬНИЙ ЦЕНТР ПРОФЕСІЙНОЇ ПЕРЕПІДГОТОВКИ
ЗВІЛЬНЕНИХ У ЗАПАС ВІЙСЬКОВОСЛУЖБОВЦІВ М. КРИВОГО РОГУ
ДНІПРОПЕТРОВСЬКОЇ ОБЛАСТІ

Дистанційний курс

“ОСНОВИ ПРОГРАМУВАННЯ”

Номінація

“Кращий електронний освітній ресурс в професіях електротехнічного профілю та зв’язку”

Автор:
Гладир Андрій Володимирович

м. Кривий Ріг
2024

Автор: Гладир Андрій Володимирович, викладач професійно-теоретичної підготовки МЦППВ, спеціаліст І категорії.
Рецензент: Одінцова Анастасія Миколаївна, методист МЦППВ м. Кривого Рогу.
Протокол засідання педагогічної ради МЦППВ №4 від 25.03.2024.

Цей курс “Основи програмування” розроблений для людей, які не мають досвіду програмування або мають мінімальні знання в цій сфері. Він підходить для тих, хто хоче навчитися основам програмування на мові Java та розпочати кар'єру розробника програмного забезпечення.

Переваги курсу: практичний підхід - курс фокусується на практичних завданнях, які допоможуть здобувачам освіти закріпити теоретичні знання та навчитися застосовувати їх на практиці; доступність - виклад навчального матеріалу подається зрозумілою мовою, з використанням простих та життєвих прикладів.

Актуальність: матеріал курсу відповідає сучасним стандартам програмування на Java.

Теми, що висвітлюються: Поняття алгоритму та програми; Типи даних; Оператори; Змінні; Синтаксис мови Java; Структури керування (if, else, for, while); Масиви.

Цей курс Java для початківців дасть вам все необхідне, щоб розпочати свою кар'єру розробника програмного забезпечення. Ви навчитеся основам програмування на Java, а також отримаєте практичний досвід роботи з цією мовою.

РЕЦЕНЗІЯ

Дистанційний курс “Основи програмування” відповідає освітній програмі для підготовки кваліфікованих робітників за професією “Оператор з обробки інформації та програмного забезпечення”. Цей дистанційний курс є відмінним вибором для тих, хто прагне опнувати програмування на Java з нуля. Зміст курсу ретельно планується з урахуванням потреб початківців, які не мають попереднього досвіду в програмуванні або мають мінімальні знання в цій галузі.

Однією з ключових переваг цього курсу є практичний підхід до навчання. Курс акцентує увагу на практичних завданнях, які допомагають здобувачам освіти закріпити теоретичні знання та навчитися застосовувати їх у реальних сценаріях. Такий підхід сприяє кращому розумінню матеріалу та розвитку навичок програмування.

Крім того, курс викладений зрозумілою мовою, що сприяє легкому засвоєнню матеріалу здобувачами освіти. Використання простих та зрозумілих прикладів дозволяє краще усвідомити ключові поняття програмування на Java.

Надзвичайно актуальний матеріал курсу відповідає сучасним стандартам програмування на Java. Здобувачі освіти отримують доступ до інформації, яка є релевантною і корисною для їхньої подальшої кар'єри у сфері розробки програмного забезпечення.

Теми, що розглядаються в курсі, широко охоплюють основи програмування на Java, об'єктно-орієнтоване програмування, алгоритми та структури даних, винятки та базові API Java. Розглянуті теми дозволяють здобувачам освіти отримати всебічні компетенції, необхідні для успішного старту в програмуванні на Java.

Перевагою курсу є система домашніх завдань після кожної теми. Ці завдання допомагають здобувачам освіти закріпити отримані знання та практично застосувати їх у вирішенні конкретних задач. Приклади тем для домашніх завдань підтверджують практичний та зрозумілий підхід курсу.

Навчання програмуванню - це складний процес, який вимагає безперервного навчання та самовдосконалення. Натомість ЕОР “Основи програмування” є ефективним практичним ресурсом для тих, хто бажає вивчити програмування на Java з нуля. Він надає необхідні знання та навички, щоб успішно розпочати кар’єру в сфері програмного забезпечення.

Для успішного проходження курсу рекомендується мати комп'ютер із встановленим JDK (Java Development Kit). Додаткові ресурси для вивчення Java,запропоновані джерела, можуть допомогти в розширенні знань та вирішенні складніших завдань.

Завдяки доступності та логіці викладу, структурованості змісту ЕОР, дистанційний курс “Основи програмування” буде корисним як для здобувачів освіти, які опановують професії, пов’язані з ІТ-галуззю, так і для тих громадян, які мають наміри отримати нову сучасну професію в дорослому віці (в тому числі перекваліфікація військових ЗСУ).

Методист Анастасія ОДІНЦОВА

ЗМІСТ

ОПИС КОНТЕНТУ ЕОР

Цей курс відповідає освітній програмі з підготовки кваліфікованих робітників за професією “Оператор з обробки інформації та програмного забезпечення”.

На сьогодні рівень сприйняття та засвоєння інформації кожного року погіршується по всьому світу, діти не можуть довгий час концентрувати увагу на новому матеріалі. Крім того, гаджети впливають на емоційний стан дітей. Саме тому при складанні курсу я намагався розділити матеріал на мікротеми, пов’язуючи матеріал кожного заняття для кращого розуміння із ситуаціями з повсякденного життя. З цією ж метою емоційно насичую заняття цікавими прикладами та гумором, це набагато підвищує захоплення здобувачів освіти.

Актуальність вивчення програмування на сьогодні беззаперечна, стрімкий розвиток ІТ потребує великої кількості фахівців, з часом ця потреба буде тільки зростати. Також беззаперечним є вплив ІТ галузі на економіку України, так за результатами 2022 року ІТ-індустрія забезпечила валютні надходження до української економіки у $7,34 млрд. Обсяг експорту збільшився на $400 млн у порівнянні з довоєнним 2021 роком. Ми спостерігали зростання щороку приблизно на 20%. У першому півріччі 2023 року експорт становив $3,38 млрд. Тому вивчення програмування, яке стає можливим для кожного завдяки цьому курсу, зробить величезний внесок у розвиток України.

З досвіду педагогічної діяльності застосування саме такого ЕОР призводить до зростання показників якості освіти з даної професії.

Дистанційний курс “Основи програмування” містить 9 відеоуроків. На кожному наступному відеоуроці подається детальне роз’яснення домашнього завдання, оголошеного наприкінці попереднього уроку. До кожного такого відеоуроку додатково надається опорний конспект із домашнім завданням на завершення мікротеми, а також практичні завдання. Зважаючи на важливість самоконтролю під час навчання основам програмування, в дистанційному курсі розміщено три поточних тестових завдання та один підсумковий тест.

ДЖЕРЕЛА

1. Наказ Міністерства освіти і науки України від 27.08.2010 р. №834 «Про затвердження Типових навчальних планів загальноосвітніх навчальних закладів ІІІ ступеню».
2. Постанова Кабінету Міністрів України від 14.01.2004 № 24 “Про затвердження Державного стандарту базової і повної загальної середньої освіти”.
3. Руденко В.Д. Жугастров О.О.Вивчаємо Java у школі: навчальний посібник у 2 частинах. Частина 1. Синтаксис мови.
4. Herbert Schildt Java: The Complete Reference, Twelfth Edition 12th Edition, 2022. - 1280 p.
5. Jonathan Middaugh. 200 Oracle Certified Associate Java SE7 (OCAJP) Study Questions: Detailed Questions and Answers + Free Beginner's Study Guide. –Amazon Services, Inc., 2014.- 722 p.
6. web-джерело з документацією Java: https://docs.oracle.com/javase/8/docs/api/

ЕЛЕКТРОННЕ ПОСИЛАННЯ НА ДИСТАНЦІЙНИЙ КУРС

GoogleClassroom: https://classroom.google.com/c/NjcxODAzNDEwMzYx?cjc=5ymf4al

ГЛОСАРІЙ ДО ДИСТАНЦІЙНОГО КУРСУ “ОСНОВИ ПРОГРАМУВАННЯ”

Алгоритм - чітка послідовність певних дій.

АЛП - аріфметико-логічний пристрій є складовою процесора

Амперсанд - знак & який призначений для виконання порівнянь з багатьма умовами.

Декремент - зменшення змінної на одиницю, яке виконується безпосередньо АЛП без виділення додаткової пам’яті для одиниці.

Дійсне число - число з крапкою.

Змінна - назва ділянки пам’яті для зберігання інформації

Інкремент - збільшення змінної на одиницю, яке виконується безпосередньо АЛП без виділення додаткової пам’яті для одиниці.

Кастинг - тимчасове змінення типу даних

Клас - один файл з кодом.

Код - текст написаної програми.

Команда - спеціальне слово, яке виконує певні дії.

Консоль - вікно для виводу результатів роботи програми.

Кратність числа - число на яке можна поділити інше число без остачі.

Масив - окремий тип даних, який дозволяє в одній змінній зберігати багато інших змінних.

Оператор - службове слово, яке виконує певні дії.

Пам’ять (оперативна пам’ять) - пристрій комп’ютера призначений для зберігання інформації.

Проект - набір класів, поєднаних між собою.

Процесор - головна складова комп’ютера призначеня для обчислення та порівняння даних.

Середа розробки - програма, яка дозволяє писати код програми, та тестувати роботу написаного коду.

Тип даних - позначення для процесора скільки пам’яті треба виділити для зберігання даної інформації

Цикл - набір команд, які повторюються декілька разів.

Ціле число - число без крапки.

SDK - (Software Development Kit) це набір інструментів, які необхідні для розробки програмного забезпечення на Java.

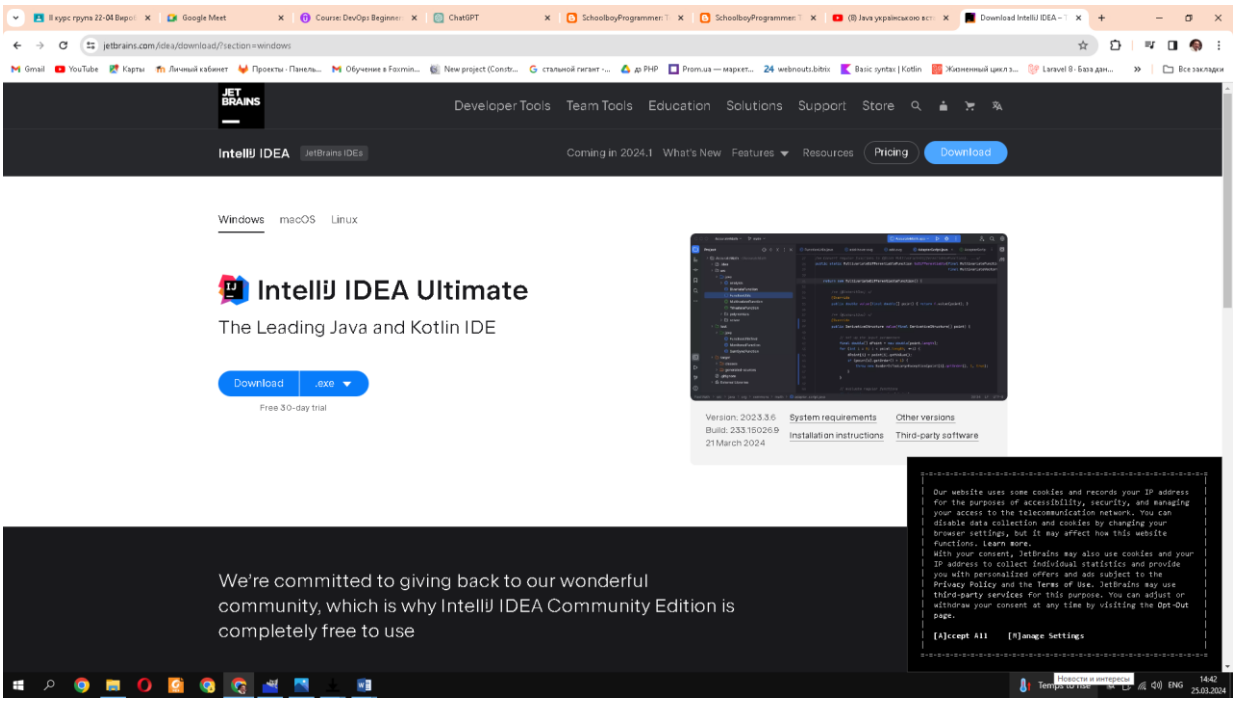
ОПОРНІ КОНСПЕКТИ ДО ВІДЕОЛЕКЦІЙ
ДИСТАНЦІЙНОГО КУРСУ “ОСНОВИ ПРОГРАМУВАННЯ”

Тема 1: Встановлення середи розробки IntelliJ IDEA та запуск першої програми.

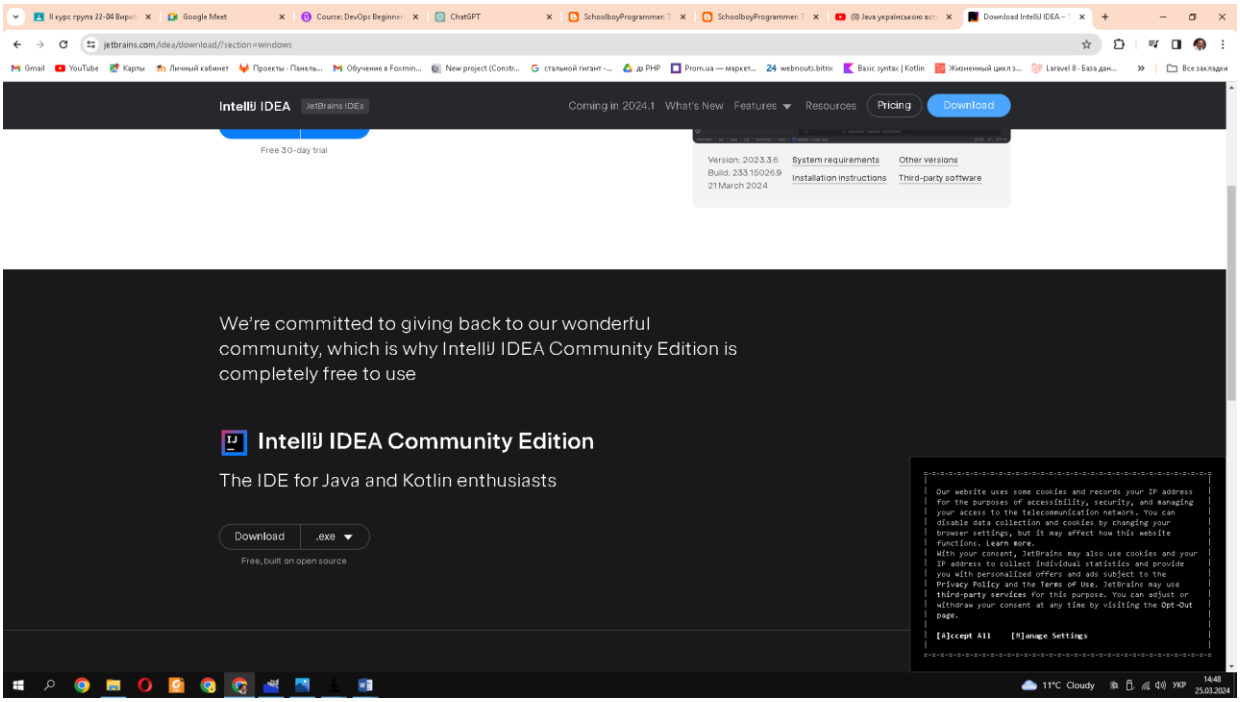
Сьогодні встановимо середу розробки та створимо першу програму.

Спочатку треба зайти на сайт https://www.jetbrains.com/idea/download

Перший запропонований на сайті варіант не треба встановлювати, він платний та призначений для корпоративної розробки.

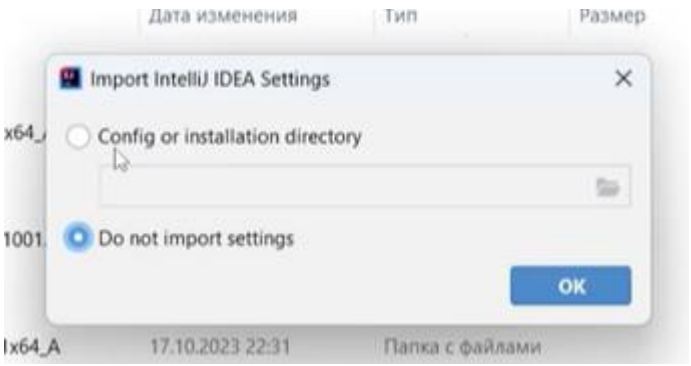


Для навчання нам буде достатньо безкоштовної версії IntelliJ IDEA Community Edition , яка знаходиться трохи нижче.



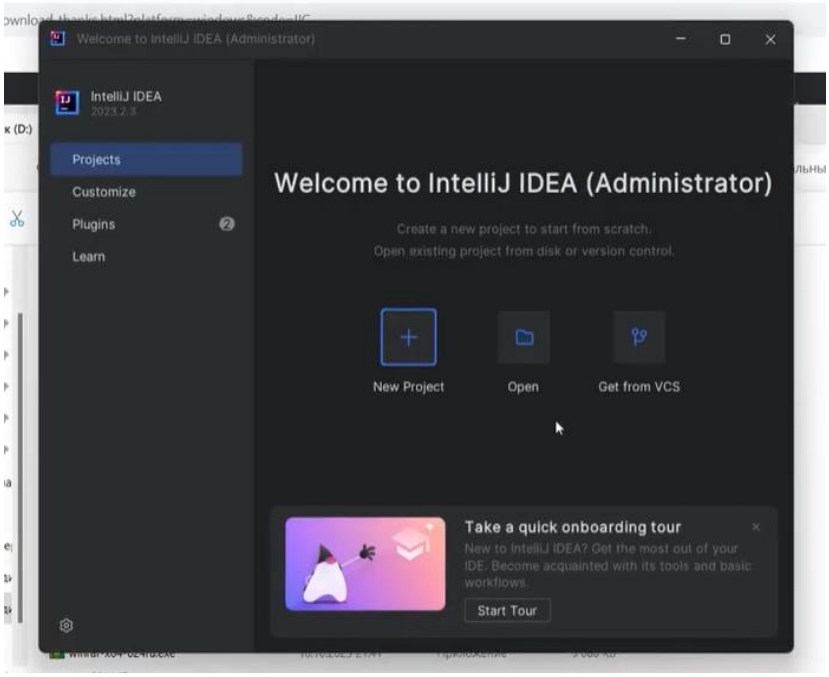
Натискаємо кнопку Download, та чекаємо завантаження. Зверніть увагу: файл досить великий - більше 500 Мб, тому, якщо використовуєте мобільний інтернет, зважайте на це.

Далі запускаємо від імені адміністратора завантажений файл. Для цього треба клікнути правою кнопкою миші по завантаженому файлу, та в розкривному меню обрати «Запустити від імені адміністратора», і натискаємо “next” у всіх вікнах. По закінченні інсталяції програми з’явиться таке вікно:

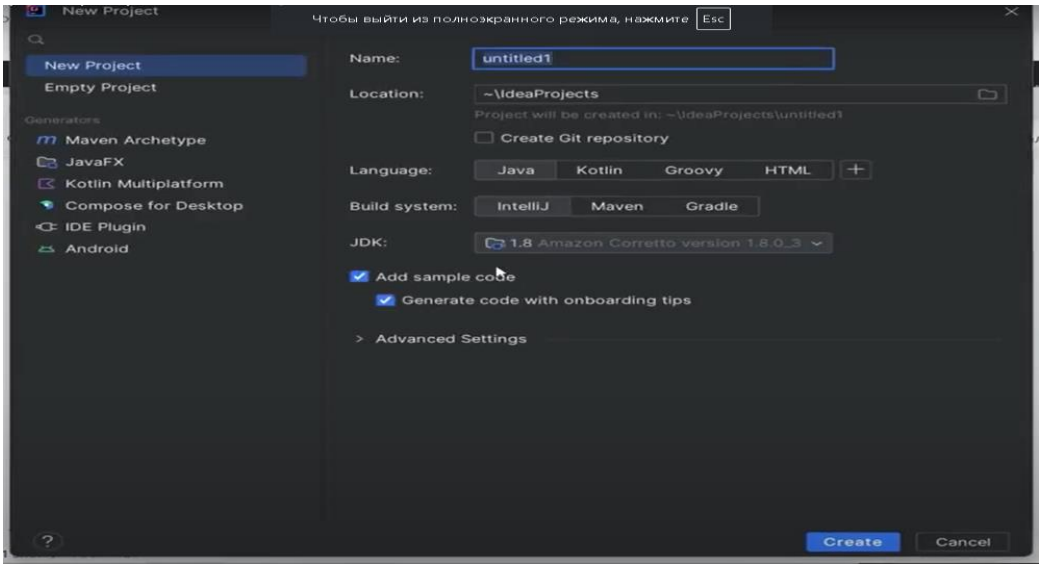


В цьому вікні інсталятор запитує, чи треба імпортувати налаштування. Обираємо пункт “Do not import settings” та натискаємо “OK”.

Далі з’явиться вікно для створення нового проєкту.



Натискаємо на кнопку + “New Project”.

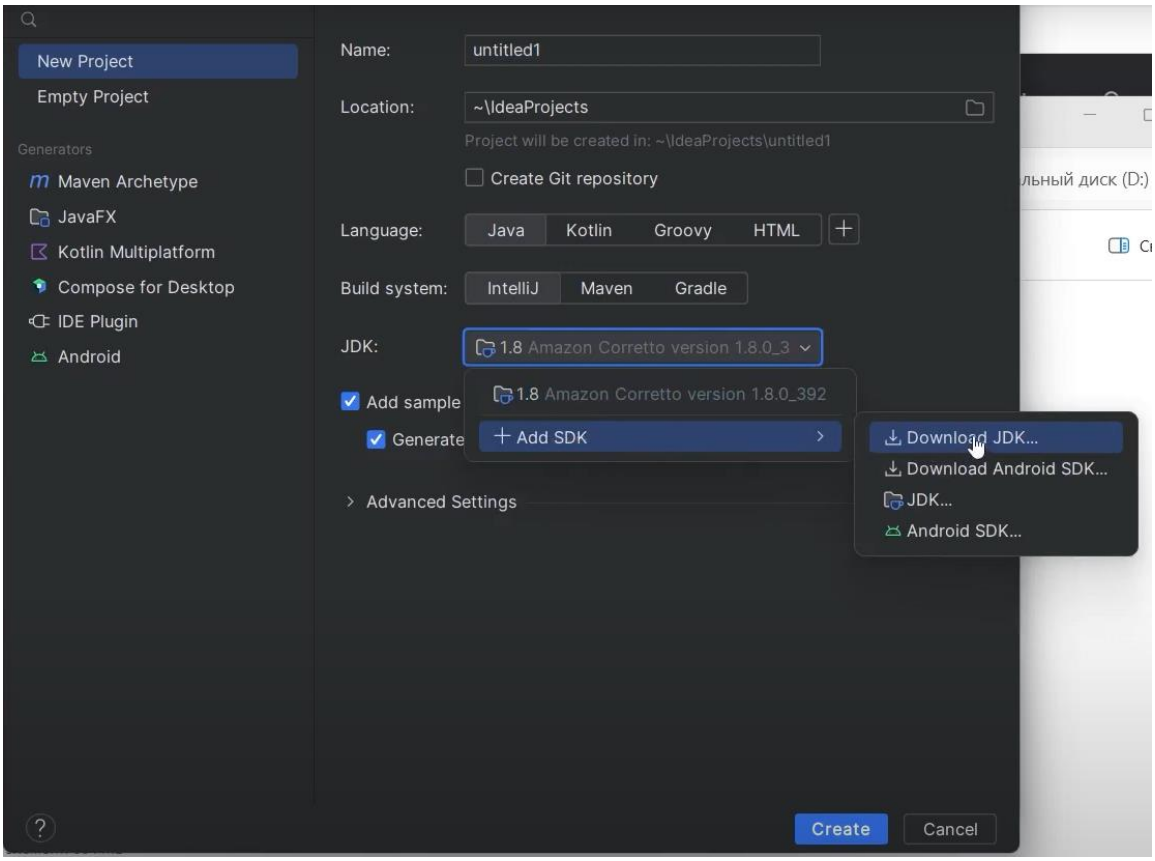


В поле Name вводимо назву проєкту, наприклад lesson1, назва повинна бути латиницею та починатися з маленької літери. Така домовленість між програмістами.

В поле Language треба обрати мову Java.

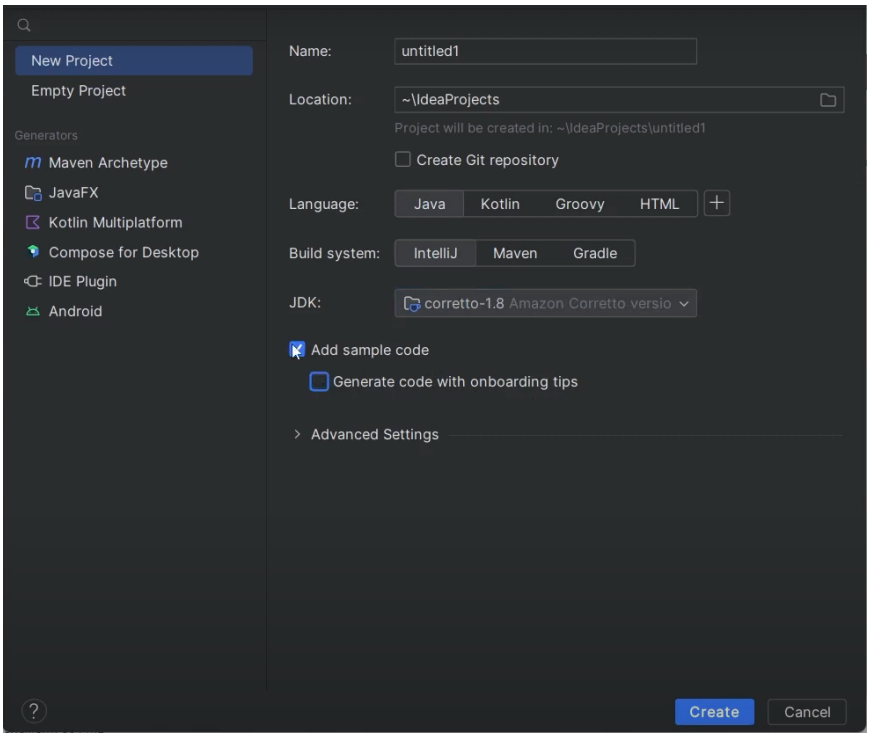
Поле Build system залишаємо все як є, за замовчуванням обрано IntelliJ.

Поле JDK скоріш за все буде мати напис червоного кольору “no sdk”. Це означатиме, що на комп’ютері не встановлений набір Java-розробника, його можна завантажити та встановити саме за допомогою цього вікна, для цього треба натиснути на випадний список та обрати.

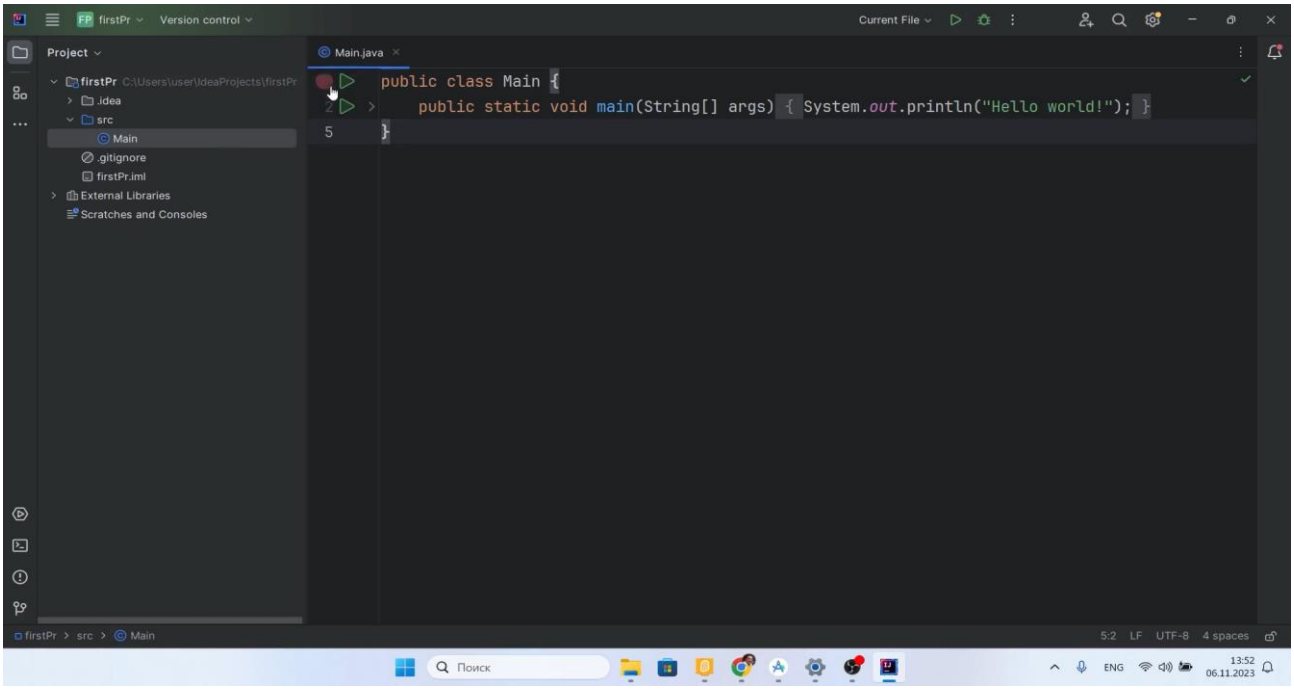


Add SDK -> Download JDK. Тепер обираємо версію jdk. Обираємо 1.8, саме її ми будемо вивчати.

Далі залишаємо галочку “add sample code”, з “generate code with onboarding tips” знімаємо виділення. Якщо не зняти виділення, то буде згенеровано багато непотрібного нам коду.



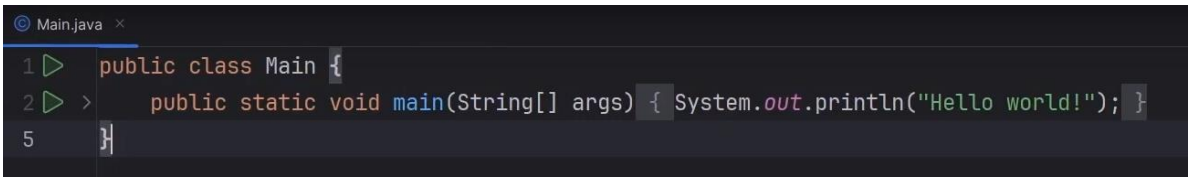
Далі натискаємо кнопку Create – і в нас відкривається вікно з нашим проєктом.



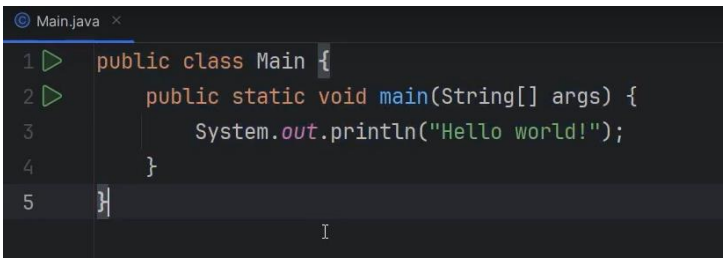
Вікно поділяється на 2 частини: 1) зліва більш вузьке вікно, це менеджер проєкту, щось на зразок провідника Windows, де ми можемо переглядати файли та папки нашого проєкту; 2) більшу частину екрану займає вікно з кодом програми, де вже є декілька рядків, які були створені завдяки тому, що ми з вами поставили галочку під час створення проєкту навпроти “add sample code”.

Розглянемо головне вікно середи розробки.

Ми бачимо наступний код.



Зліва є нумерація рядків, але пропущені 3 та 4 рядок. Вони згорнуті, щоб їх розгорнути, треба натиснути в другому рядку на кнопку “>”, після чого код розкриється і вже буде виглядати таким чином:



Тепер розглянемо окремо кожен рядок.

Рядок 1. Більшість файлів у java називаються «класс». Слово public означає, що цей файл публічний, тобто ним можна користуватися з будь-якого місця нашого проєкту. Слово “Main” – це назва нашого класу.

Рядок 2. public static void main(String[ ] args) – поки що, вам буде достатньо розуміння, що так позначається головний файл у проєкті і саме з цього рядка починається виконуватися код.

Рядок 3. Досить довга команда System.out.println(“Hello world”); просто виведе у консоль текст “Hello world”. Зверніть увагу на крапку з комою в кінці рядка, вона означає, що це кінець команди, і всі команди в Java в кінці рядка повинні мати крапку з комою, інакше програма не запуститься.

Рядок 4. Закриваюча фігурна дужка. Всі команди, які ми хочемо написати, повинні знаходитися між відкриваючою фігурною дужкою в другому рядку та закриваючою фігурною дужкою в четвертому рядку. Якщо ви продублюєте команду System.out.println(“Hello world”), але напишете її після закриваючою дужки, то буде помилка, програма не запуститься.

Рядок 5. Закриваюча фігурна дужка класу закриває відкриваючу дужку в першому рядку, тим самим говорить нам, що це кінець класу.

Кількість відкриваючих та закриваючих дужок повинна завжди бути рівною (це дуже поширена помилка у початківців).

Ну, і нарешті, щоб запустити нашу з вами першу програму треба натиснути на зелений трикутник, зверху в головному меню. Іще він є у першому та другому рядку. Після цього знизу відкриється консоль, де ми побачимо результат роботи нашої першої програми.

Вітаю, Ви встановили середу розробки та запустили свою першу програму. Можете собою пишатися! 😊

Домашнє завдання:

Вивести на екран улюблений вірш за допомогою команди System.out.println(“…”);

Тема 2: Типи даних та арифметичні дії з ними.

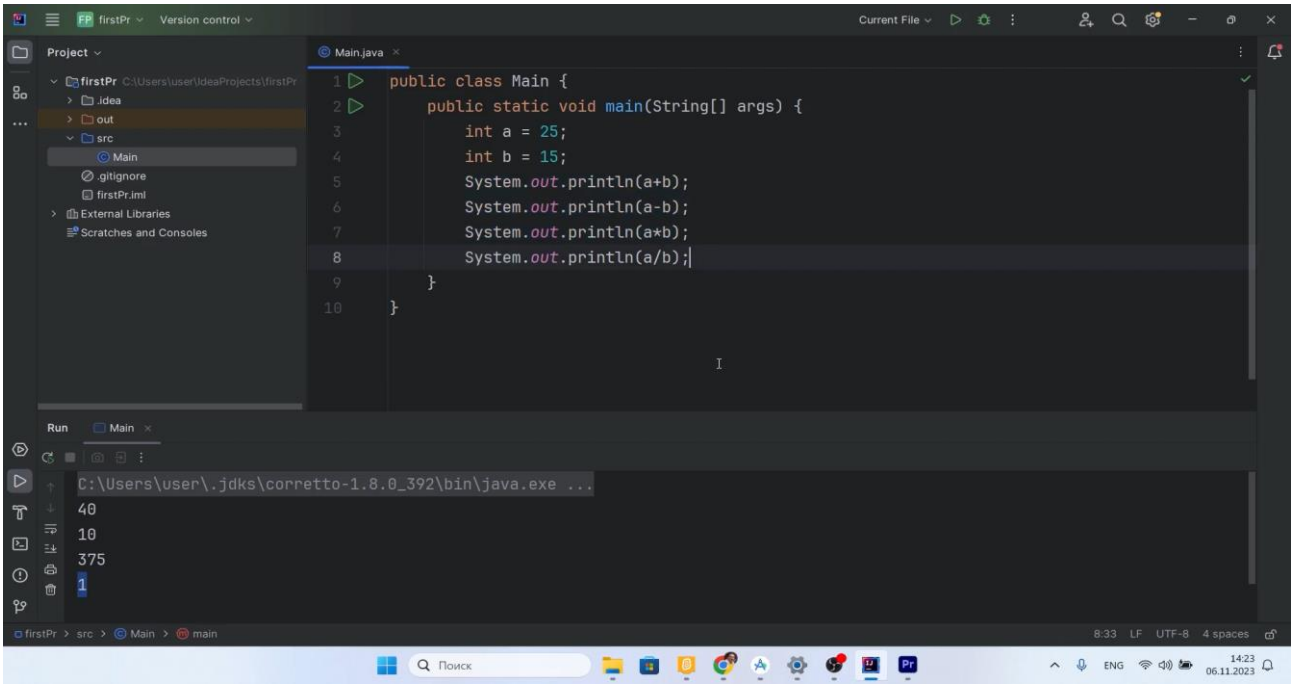
Будь-яка програма повинна обробляти інформацію. Для зберігання інформації потрібно виділити пам’ять. Завдання програміста – раціонально використовувати пам’ять комп’ютера, тому в Java створили декілька типів даних: для невеликих чисел (від -128 до 127) достатньо одного байта, для більших чисел (від -32768 до 32767) треба вже 2 байти, а в 4 байтах можна зберегти числа аж від -2147483648 до 2147483647. Крім того, існують дійсні числа (з крапкою, наприклад 3.14), для них існує свій тип даних.

Тип даних визначає обсяг пам'яті, який використовується для зберігання значення, а також можливі діапазони допустимих значень.

У Java існують наступні примітивні типи даних:

- byte: 8-бітне ціле число. Діапазон від -128 до 127.
- short: 16-бітне ціле число. Діапазон від -32768 до 32767.
- int: 32-бітне ціле число. Діапазон від -2147483648 до 2147483647.
- long: 64-бітне ціле число. Діапазон від -9223372036854775808 до 9223372036854775807.
- float: 32-бітне дійсне число з плаваючою комою. Діапазон від приблизно ±1.4x10^(-45) до приблизно ±3.4x10^38.
- double: 64-бітне дійсне число з плаваючою комою. Діапазон від приблизно ±4.9x10^(-324) до приблизно ±1.7x10^308.
- boolean: логічний тип, може приймати значення true або false.
- char: 16-бітний Unicode символ.

Розглянемо приклад.

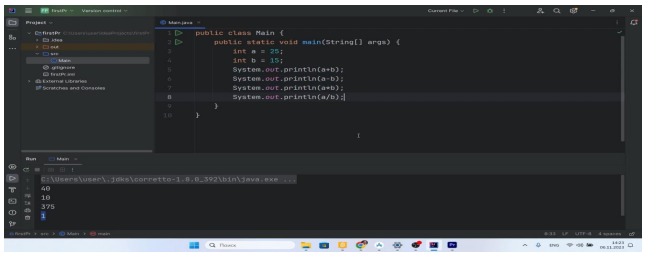


Створюємо змінну *a* цілого типу int, та надаємо їй значення 25.

Створюємо змінну *b* цілого типу int, та надаємо їй значення 15.

Далі виводимо на екран арифметичні дії додавання, віднімання, множення та ділення.

Після запуску програми на виконання ми отримуємо такий результат у консолі:



Перші три дії обчислено правильно, але з останньою дією ділення виникли проблеми, тому що

25/15 = 1,667, але аж ніяк не 1.

Така помилка відбулась через те, що у процесорі існують два різних пристрої для обчислень, які називаються арифметико-логічний пристрій (АЛП). Один дуже швидкий, але може обчислювати лише цілі числа, другий обчислює повільніше у 1000 разів, але вміє обчислювати дійсні числа (дробові).

При оголошенні змінних *a* та *b* ми вказали цілий тип int, тому процесор вирішив використовувати швидкий АЛП, який працює з цілими числами, але в ньому немає місця для збереження даних після коми.

Ми можемо змінити тип даних int на double, всі обчислення будуть правильними. Але в такому разі в усіх прикладах ми будемо використовувати повільний АЛП, що теж не є гарним рішенням.

Правильно буде у діленні в дужках вказати тип (double) перед будь-якою змінною. Цей прийом називається кастинг, таким чином ми вказуємо процесору, що саме зараз треба використати АЛП з більш високою точністю, а всі інші приклади будуть обчисленні швидким АЛП.

Домашнє завдання:

1. обчислити площу прямокутника, для цього треба створити 2 змінні типу int
2. обчислити площу круга, для цього треба створити 2 змінні типу double (радіус та PI)

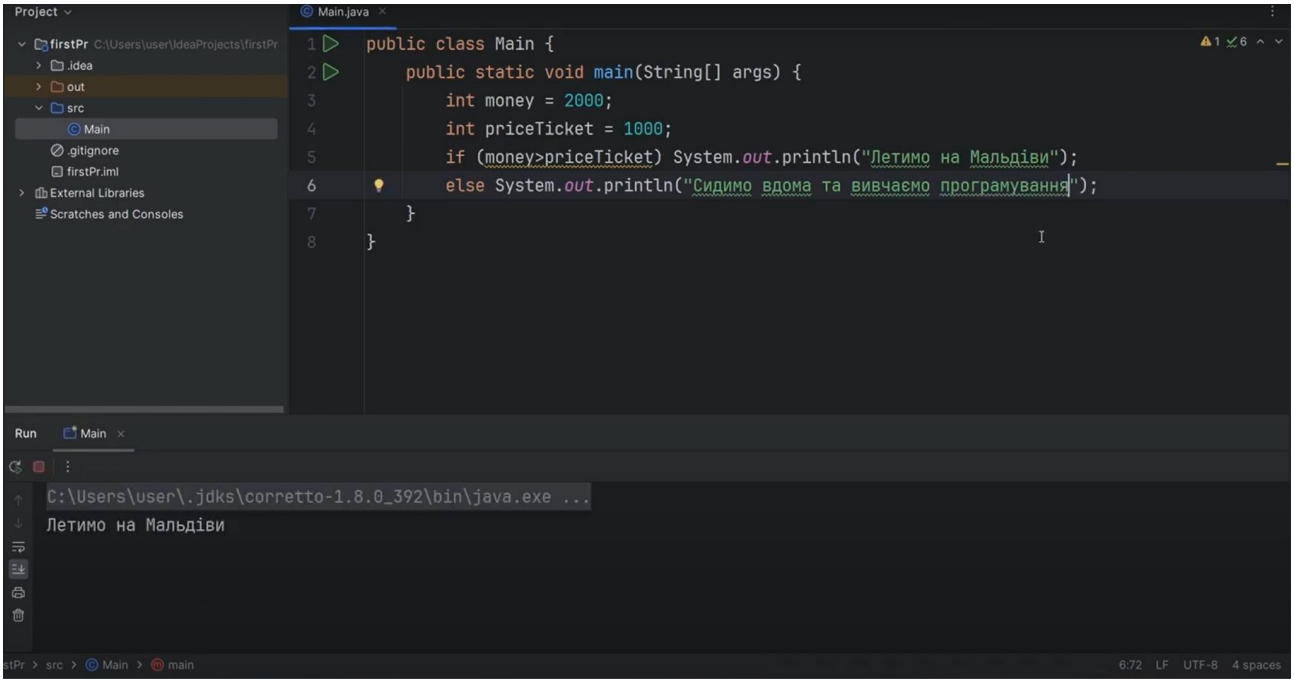
Тема 3: Умовний оператор if.

Кожну мить у житті ми робимо вибір, наприклад: якщо хочемо їсти, йдемо до холодильник; якщо хочемо спати, лягаємо в ліжко. Тобто весь наш день ми можемо поділити на певну кількість подій та відповідних реакцій на ці події. Із цього складається все наше життя.

Тому оператор if, який перекладається саме як «якщо» можна вважати головним оператором програмування, тому що програма повинна аналізувати якісь дані і відповідно реагувати тим, чи іншим шляхом.

Давайте розглянемо використання цього оператора на прикладі наступної задачі:

Ми вирішили полетіти у відпустку на Мальдіви. Для цього нам потрібно мати гроші, причому бажано, щоб їх було більше, ніж коштує квиток. Отже, нам треба оголосити дві змінні цілого типу, думаю, копійки не треба враховувати.



У третьому рядку оголошуємо змінну money та зберігаємо в ній 2000, тобто ми маємо 2000.

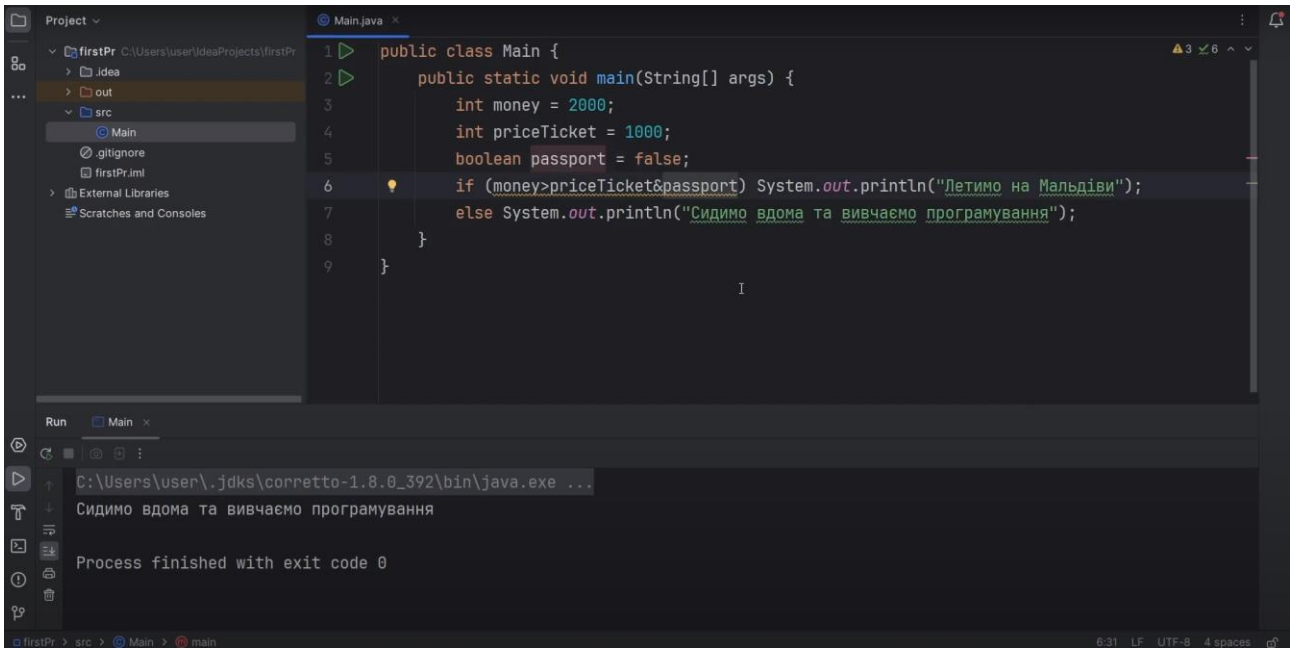
У четвертому рядку оглошуємо змінну priceTicket, тобто ціна квитка – 1000.

Зверніть увагу на назву змінної priceTicket. У Java прийнято давати назви змінним в стилі “Camel case”, тобто як верблюд має горби, так і назва змінної, якщо має більше одного слова, то наступне слово пишемо з великої літери. Таким чином назва стає схожою на верблюда. Справа в тому, що не можна ставити пробіл у назві змінної, а без пробілу стає складніше прочитати назву змінної. Ще слід зазначити, що назва змінної повинна бути написана так, щоб будь-хто відразу зрозумів, для чого ця змінна створена. Можна було б скоротити priceTicket до pt, але тоді через деякий час, ви не пригадаєте вже, для чого створили цю змінну.

У п’ятому рядку ми використовуємо умовний оператор if. Цей рядок, якщо перекласти українською, можна прочитати так: якщо грошей більше, ніж коштує квиток, то… виводимо на екран ***“Летимо на Мальдіви”***, інакше виводимо на екран “***Сидимо вдома та вивчаємо програмування***”.

Якщо умова в дужках після оператора if виконується, тоді працює System.out.println(“***Летимо на Мальдіви***”); якщо не виконується, тобто грошей менше, ніж коштує квиток, тоді працює команда після else System.out.println(“***Сидимо вдома та вивчаємо програмування***”);

Тепер додаємо ще одну умову: для подорожі нам потрібен паспорт, при чому нам не потрібно зберігати у пам'яті комп'ютера кількість, паспорт або є, або немає, це всі можливі значення змінної. Такі змінні, які мають лише два значення «так» або «ні» мають тип boolean і можуть мати тільки значення true або false. Тоді наш код буде мати такий вигляд:



Для того, щоб додати ще одну умову в оператор if, у дужках між умовами треба поставити оператор амперсанд & (цей знак означає and).

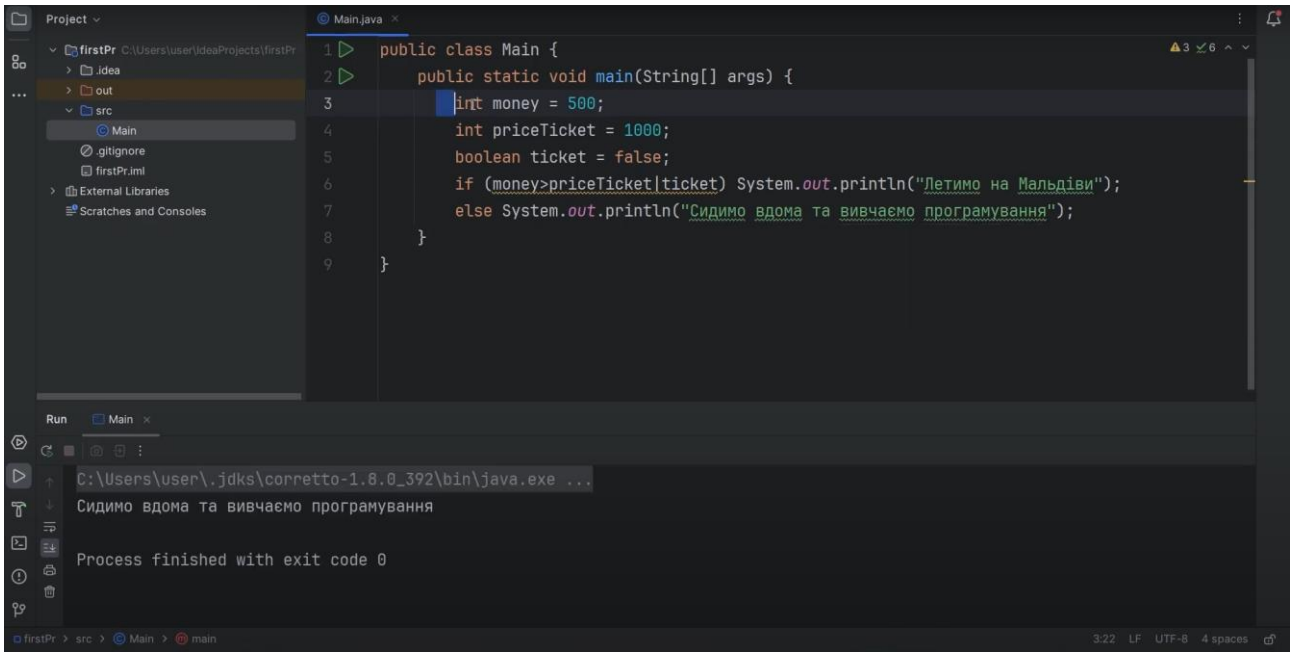
Ви зустрічали цей знак не один раз у житті. До речі, назва цих цукерок не «Ем ем денс», правильно читати «Ем енд емс», уявляєте, ви все життя неправильно називали цукерки, тепер живіть з цим.

По центру і є той самий and, і назва цих цукерок складається з імені m однієї цукерки та багатьох інших m`s

Тобто тепер шостий рядок ми можемо прочитати так: якщо грошей більше, ніж коштує квиток **ТА** є паспорт, то тільки тоді ми летимо на Мальдіви

Існує ще один оператор | Це оператор "or" (або), він знаходиться на англійській розкладці клавіатури біля клавіші Enter. Для його використання треба натиснути клавішу shift.

Поміняємо наш код. Ми можем полетіти на Мальдіви, якщо в нас грошей більше, ніж коштує квиток, або в нас вже є квиток, тоді грошей може бути і менше, але ми все одно зможемо полетіти. Цю умову можна написати так:



У шостому рядку дві умови розділені оператором or |

Тоді System.out.println(“***екран Летимо на Мальдіви***”); спрацює, якщо грошей більше, ніж коштує квиток, або якщо є квиток. І лише коли грошей менше, ніж коштує квиток, і немає квитка, ми отримаємо else System.out.println(“***Сидимо вдома та вивчаємо програмування***”);

Оператори & та | частіше використовують парними, тобто && та ||. Такий запис дозволяє прискорити виконання умовного оператора.

Наприклад, якщо в умові грошей більше, ніж коштує квиток, тоді після перевірки першої умови вже зрозуміло, що ми можемо полетіти, перевіряти наявність квитка немає сенсу. Але якщо ми використовуємо один оператор |, то перевірка все одно відбудеться. А от якщо ми використаємо подвійний оператор ||, то перевірка на наявність квитка не буде відбуватися, тим самим ми вдвічі прискорили наш код. Тому краще завжди використовувати подвійні оператори && та ||.

Домашнє завдання: Написати код, який буде виводити на екран текст «Йдемо у кіно», якщо у нас грошей більше, ніж коштує квиток та є компанія друзів, в іншому випадку виводимо текст «Йдемо додому»

Тестове опитування для самоконтролю №1

1. Який синтаксис оператора if в Java?

   a) if (умова) { блок коду }

   b) if { умова : блок коду }

   c) if (блок коду : умова)

2. Що буде результатом виразу if (x > 5) { System.out.println("x більше за 5"); }, якщо x дорівнює 3?

   a) Повідомлення "x більше за 5" виведеться

   b) Нічого не виведеться

   c) Виникне помилка компіляції

3. Який оператор використовується для перевірки кількох умов в операторі if?

   a) else

   b) else if

   c) or

4. Що виведеться на екран після виконання наступного коду:

```
int number = 10;
if (number < 5) {
    System.out.println("Число менше 5");
} else if (number < 15) {
    System.out.println("Число менше 15");
} else {
    System.out.println("Число більше або рівне 15");
}
```

   a) Число менше 5

   b) Число менше 15

   c) Число більше або рівне 15

5. Яка буде дія, якщо умова в операторі if є хибною?

   a) Виконається блок коду після if

b) Виконається блок коду після else

c) Ніякий з блоків коду не буде виконаний

6. Що буде значення змінної result після виконання наступного коду:

```
int age = 25;
String result;

if (age >= 18) {
    result = "Дорослий";
} else {
    result = "Дитина";
}
```

a) "Дорослий"

b) "Дитина"

c) Нічого не виведется

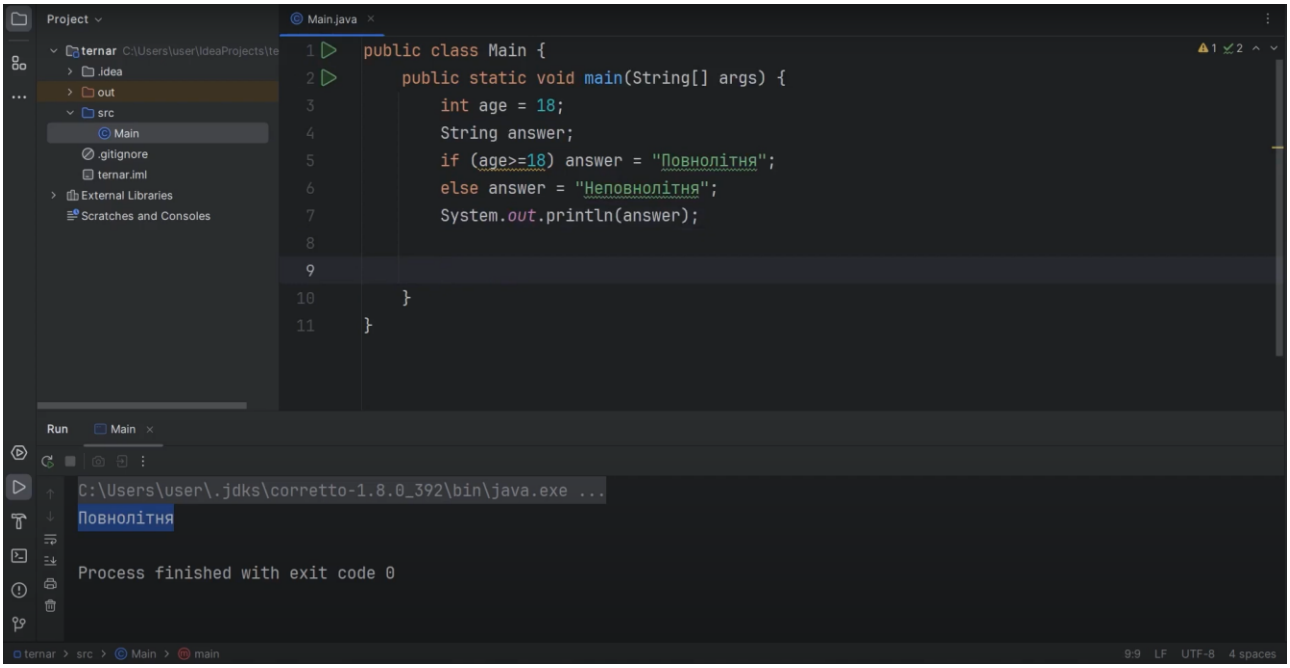
Тема 4: Тернарний оператор if

Тернарний оператор в Java – це спеціальний оператор, який дозволяє виразити умову в одному рядку коду, уникнувши довгих конструкцій if-else. Він має наступний синтаксис:

вираз_умови ? вираз_якщо_умова_істинна : вираз_якщо_умова_хибна

Цей оператор перевіряє вираз_умови. Якщо вираз_умови істинний (true), виконується вираз_якщо_умова_істинна; якщо вираз_умови хибний (false), виконується вираз_якщо_умова_хибна.

Розглянемо задачу: треба вивести на екран, залежно від віку людини, інформацію: повнолітня чи неповнолітня людина.



У третьому рядку ми оголошуємо змінну age (вік) та присвоюємо їй значення 18.

У четвертому рядку створюємо змінну answer типу String.

String – це тип, в якому можна зберігати будь-який текст, далі у п'ятому рядку ми присвоюємо змінній answer значення «Повнолітня, якщо людина має 18 або більше років», в іншому випадку (у рядку 6) присвоюємо значення «Неповнолітня», далі в сьомому рядку виводимо на екран значення змінної answer.

У консолі ми бачимо «Повнолітня», тому що умова виконана.

А тепер подивимось як цей код буде виглядати при використанні тернарного оператора:

```
9           answer = age>18?"Повнолітня":"Неповнолітня";
10          System.out.println(answer);
```

Рядок 9 повністю ідентичний п'ятому та шостому, тобто він виконує ті самі дії. Знак питання відокремлює умову від двох варіантів подій, двокрапка розділяє ці події. Перша подія відбудеться, якщо умова виконується, якщо умова не виконується, то відбудеться друга подія.

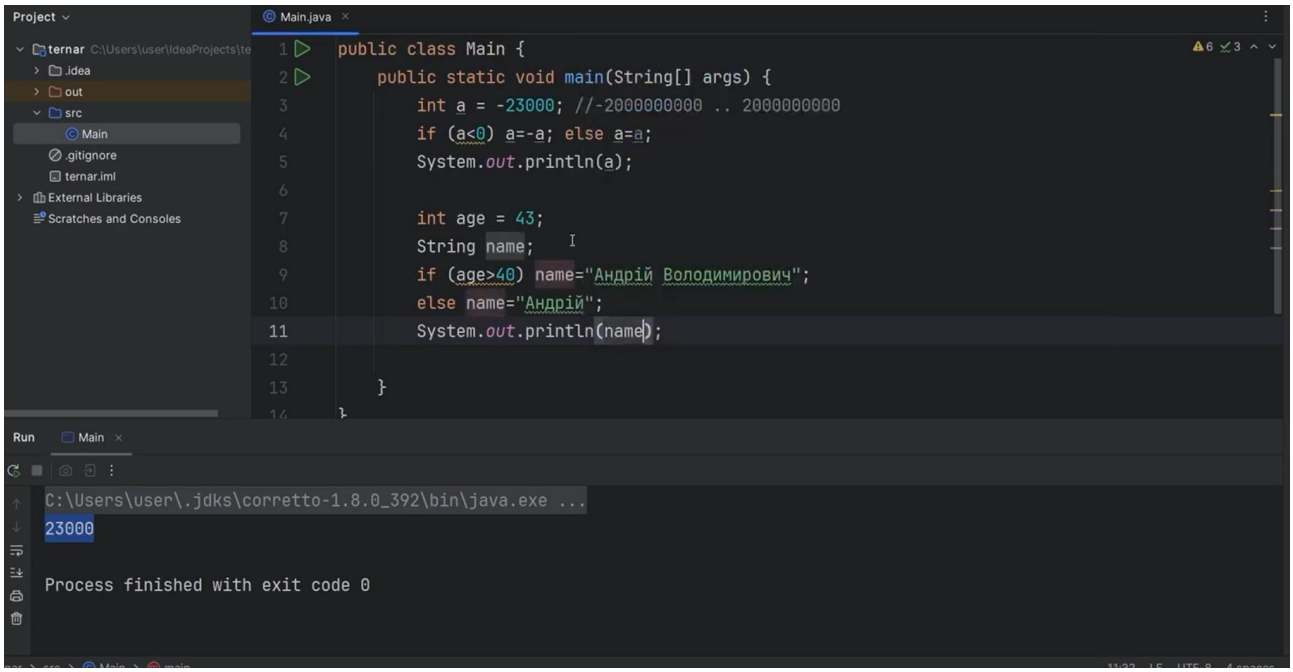
Розглянемо ще один приклад використання тернарного оператора:

```
32          int number = 10;
33          String result = (number > 0) ? "Додатне" : "Від'ємне або нуль";
34          System.out.println(result);
```

У цьому прикладі змінна number порівнюється з нулем. Якщо number більше за 0, то змінній result присвоюється рядок "Додатне". Якщо number менше або дорівнює 0, то змінній result присвоюється рядок "Від'ємне або нуль".

Тернарний оператор часто використовується для присвоєння значень залежно від умови в одному рядку, що робить код більш компактним та більш зрозумілим. Однак слід бути обережним при використанні тернарного оператора, оскільки в деяких випадках він може зробити код менш зрозумілим, особливо якщо в ньому використовуються складні вирази.

Домашнє завдання:



Замініть код if-else на тернарний оператор.

Практичні завдання:

1. Напишіть програму, яка визначає, чи є число додатним чи від'ємним, використовуючи тернарний оператор. Виведіть відповідне повідомлення на екран.
2. Напишіть програму, створіть три змінних типу int і запишіть туди числа, треба вивести найбільше з них за допомогою тернарного оператора.
3. Напишіть програму, яка аналізує оцінку студента (від 0 до 100) і виводить відповідний рядок (наприклад, "відмінно", "добре", "задовільно", "незадовільно") використовуючи тернарний оператор.

Тема 5: Типи даних byte, shot.

На другому занятті ми з вами ознайомилися з різними типами даних, але частіше за все використовуються: тип int – для цілих чисел, тип double – для дійсних чисел. Хоча окрім int у Java існують ще типи shot, byte та long, але користуються ними дуже рідко. З'ясуймо, чому це так.

Пропоную набрати такий код:

```
8       byte e=10;
9       byte f=15;
10      byte sum=e+f;
```

Нагадаю, що у тип даних byte можна зберігати числа від -128 до 127.

Ми створили дві змінні

byte e=10;

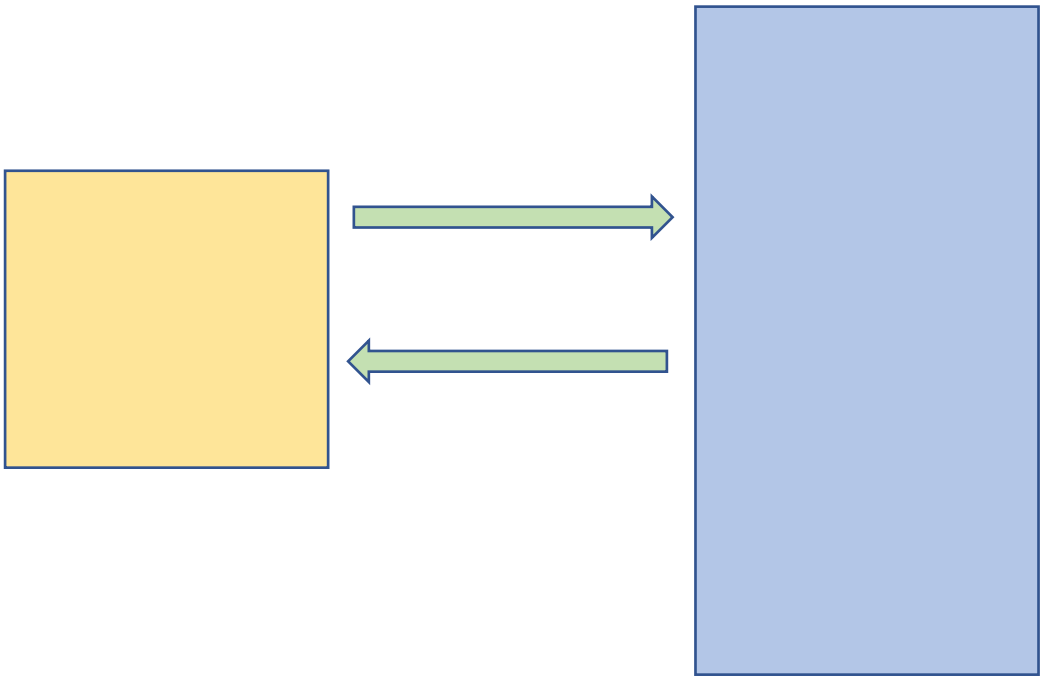
byte f=15;

Створюємо змінну byte sum=e+f; яка, начебто задовільняє вимогам типу byte, але у нас виникає помилка. З'ясуймо, звідки вона виникає.

Але спочатку розберемо схему Фон Неймана, суть якої полягає в тому, що для роботи комп'ютера потрібен процесор, який виконує обчислення, та пам'ять, у якій зберігаються дані для обчислень. Процесор та пам'ять за один раз можуть передати один одному 32 значення 0 або 1.

32 біта – це 4 байти, саме стільки виділяється для типу int. Тобто всі дані, які обробив процесор будуть займати 4 байти.

Тепер стає зрозумілим, чому byte sum=e+f; видає помилку: тому що "e+f" – це додавання, яке виконує процесор, і після додавання тип даних стане int. І ми намагаємося тип даних, який займає 4 байти, помістити у byte sum, який займає один байт. Це неможливо, як і не можливо у стакан налити відро води.



Варіанти вирішення проблеми:

1 варіант: зробити обчислення, а після цього зробити кастинг до byte

```
byte sum=(byte)(e+f)
```

2 варіант: зробити змінну sum більшим типом int

```
int sum=e+f;
```

Отже, через цю проблему типи byte та shot не використовуються для обчислень, тому навіть якщо у вашому проєкті невеликі числа, все одно краще використовувати тип int.

Тема 6: Цикли у Java.

Кожен наш день протягом тижня, у переважній більшості, схожий на попередній, тож алгоритм тижня можна прописати таким чином:

повторити 5 разів:

{

1. **прокинутися**
2. **вмитися**
3. **поїсти**
4. **попрацювати**
5. **ще декілька разів поїсти**
6. **відпочити**

}

Так писати зручніше, ніж 5 разів повторювати ці самі рядки.

Загадка: програміст не може вже тиждень вийти з ванної кімнати, бо не може помити волосся. Що пішло не так?

Відповідь: на шампуні криво написана інструкція:

“Нанести шампунь на волосся

змити

повторити”

Команда повторити повертає у початок алгоритму. Умова виходу з циклу в цій інструкції загубилася. В результаті програміст не може закінчити алгоритм, він застряг у циклі.

Про умову виходу з циклу треба завжди пам’ятати, інакше ваша програма “зависне”.

Розглянемо, як користуватися циклами у Java.

Перший оператор циклу **while, (поки)**.

Напишемо код, який буде виводити у консоль 5 разів рядок “Hello, World!”

У першу чергу створимо лічильник, щоб не було такої ситуації, як з голодним програмістом у ванній кімнаті; напевно, за тиждень він зголоднів.

Main.java

```
public class Main {
    public static void main(String[] args) {
        int count = 0;
        while (count<5){
            System.out.println("Hello world!");
            count = count +1;
        }


    }
}
```

У третьому рядку ми оголошуємо лічильник count

У четвертому рядку пишемо оператор циклу, який можна перекласти *поки count<5 виконуй рядки 5 та 6*

Саме рядки 5 та 6 знаходяться між фігурними дужками після оператору while, і все, що знаходиться між цими дужками і буде виконано у циклі.

Рядок 5 просто виведе на екран “Hello, World!”

У шостому рядку збільшуємо лічильник на одиницю. Цикл буде повторюватися доти, доки лічильник не стане дорівнювати 5, тоді цикл завершиться.

Другий варіант циклів у Java, це оператор for

```
9       for (int i = 0; i <= 5; i++) {
10          System.out.println("Hello world! for"+i);
11      }
```

У циклі while() треба окремо створити лічильник, прописати у дужках умову роботи циклу, і ще в тілі циклу, між фігурних дужок збільшити лічильник на одиницю, у циклі for() всі ці перелічені дії знаходяться в одному місці у параметрах циклу (у дужках).

Тепер з’ясуємо, що означає i++. Якщо коротко, то це те саме, що i=i+1; і має назву ***інкремент,*** Переваги використання інкременту – це можливість менше писати та виконання коду в два раза швидше. Можна ще використовувати ***декремент i--*** це те саме, що i=i-1;

Тестове опитування для самоконтролю №2

1. Який синтаксис циклу for в Java?

   a) for (ініціалізація; умова; крок) { блок коду }

   b) for { умова : блок коду }

   c) for (блок коду : умова)

2. Яка буде дія циклу while, якщо умова виконання циклу завжди істинна?

   a) Цикл буде виконуватися безкінечно

   b) Цикл не буде виконуватися жодного разу

   c) Виникне помилка компіляції

3.. Що виведеться на екран після виконання наступного коду:

```
for (int i = 0; i < 5; i++) {
    System.out.print(i + " ");
}
```

   a) 0 1 2 3 4

   b) 1 2 3 4 5

   c) 0 1 2 3

4. Який результат буде на екрані після виконання такого коду:

   byte a = 7;

   byte b = 8;

   byte sum = a*b;

   System.out.println(sum);

   a) 48

   b) 56

   c) буде помилка

5. Який результат буде на екрані після виконання такого коду:

   short a = 10;

   int b = 5;

   int result = a+b*2;

   a) 20

   b) 30

   c) буде помилка

6. Який тип повинна мати змінна res, щоб код запустився без помилок

```
byte a = 23;
short b = 12;
_____ res = a+b;
```

a) byte

b) short

c) int

Тема 7: Вкладені цикли у Java.

Давайте проаналізуємо наш робочий тиждень. В середньому людина приймає їжу тричі на день, написати алгоритм можна так:

повтори 3 рази: їсти

Ця дія повторюється протягом тижня, тоді запишемо так:

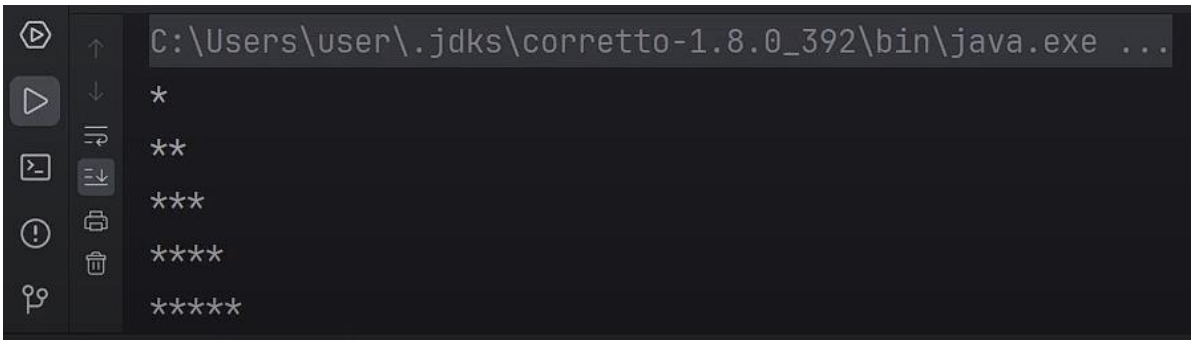
повтори 5 раз:{

    повтори 3 рази: їсти

}

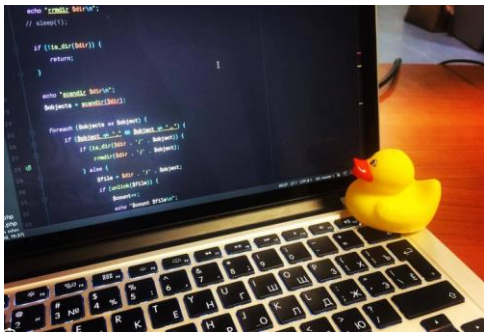
Це називається вкладений цикл, тобто в середині одного циклу існує інший цикл.

Розглянемо дуже поширену задачу: за допомою вкладених циклів вивести на екран зірочки у вигляді трикутника.



Для виконання цього завдання нам потрібно 5 циклів, для кожного рядка один цикл, тому що зірочки ми будемо друкувати по одній, в першому циклі - один раз, в другому 2, і так далі.

```
8       //*
9       //**
10      //***
11      //****
12      //*****
13
14      for (int i = 1; i < 6; i++) {
15          for (int j = 0; j < i; j++) {
16              System.out.print("*");
17          }
18          System.out.println();
19      }
```



Для кращого розуміння цього коду скористаємося методом “Каченя”: якщо код не зовсім зрозумілий, то треба придбати маленьке жовте каченя, потім посадити це каченя біля монітора так, щоб йому все було гарно видно, наприклад, отак:

Тепер треба нашому каченяті пояснити, як працює код, для кращого розуміння треба якомога чіткіше все пояснювати, кожен крок.

Отже, почнемо.

У 14 рядку ми створюємо цикл for, перший параметр створює змінну i=1, далі перевіряємо 1<6, поки що менше, тому заходимо в тіло циклу.

У 15 рядку ми знову створюємо цикл для друку зірочок, створюємо змінну j=0. На цей момент у нас i=1, тепер перевіряємо, чи j<i? Так, 0<1, якщо відповідь “так”, то заходимо в тіло циклу.

У 16 рядку ми виводимо на екран одну зірочку, причому зверніть увагу на команду ***System.out.print(“*”);*** Раніше ми додавали в кінці ***ln,*** що означало ***line new (новий рядок)*** Зараз нам не треба кожен раз переходити на новий рядок, навпаки, нам треба залишатися на одному рядку, тому пишемо без ***ln.***

У 17 рядку ми дійшли до кінця внутрішнього циклу, який почався у 15 рядку, в кінці циклу до лічильника додається одиниця, отже, тепер j=1. Повертаємося у рядок 15.

У 15 рядку перевіряємо j<i? Ні, 1 не менше 1, якщо відповідь “ні”, то завершуємо цикл і потравляємо до рядку 18.

У 18 рядку ми друкуємо порожній рядок, але завдяки ***System.out.println();*** ми переходимо на новий рядок, де нам треба надрукувати вже дві зірочки, перший рядок вже надрукований.

У 19 рядку ми додаємо до лічильника i=1 одиницю, тепер i=2. Повертаємось у перший зовнішній цикл до рядку 14.

14 рядок перевіряємо i<6? Так 2<6? якщо відповідь “так”, то заходимо в тіло циклу. Переходимо до 15 рядка.

У 15 рядку ми знову створюємо цикл для друку зірочок, створюємо змінну j=0. На цей момент у нас i=2, тепер перевіряємо, чи j<i? Так, 0<2, якщо відповідь “так”, то заходимо в тіло циклу.

У 16 рядку ми виводимо на екран одну зірочку.

У 17 рядку ми дійшли до кінця внутрішнього циклу, який почався у 15 рядку, в кінці циклу додається до лічильника одиниця, отже, тепер j=1. Повертаємося у рядок 15.

У 15 рядку перевіряємо j<i? Так, 1<2, якщо відповідь “так”, то заходимо в тіло циклу.

У 16 рядку ми виводимо на екран другу зірочку.

У 17 рядку ми дійшли до кінця внутрішнього циклу, який почався у 15 рядку, в кінці циклу додається до лічильника одиниця, отже, тепер j=2. Повертаємося у рядок 15.

У 15 рядку перевіряємо j<i? Ні, 2 не менше 2, якщо відповідь “ні”, то завершуємо цикл і потрапляємо до рядка 18.

У 18 рядку ми друкуємо порожній рядок, але завдяки ***System.out.println();*** ми переходимо на новий рядок, де нам треба надрукувати вже три зірочки, два рядки вже надруковані.

і так далі.

Сподіваюся, що каченя зрозуміло, а ви?

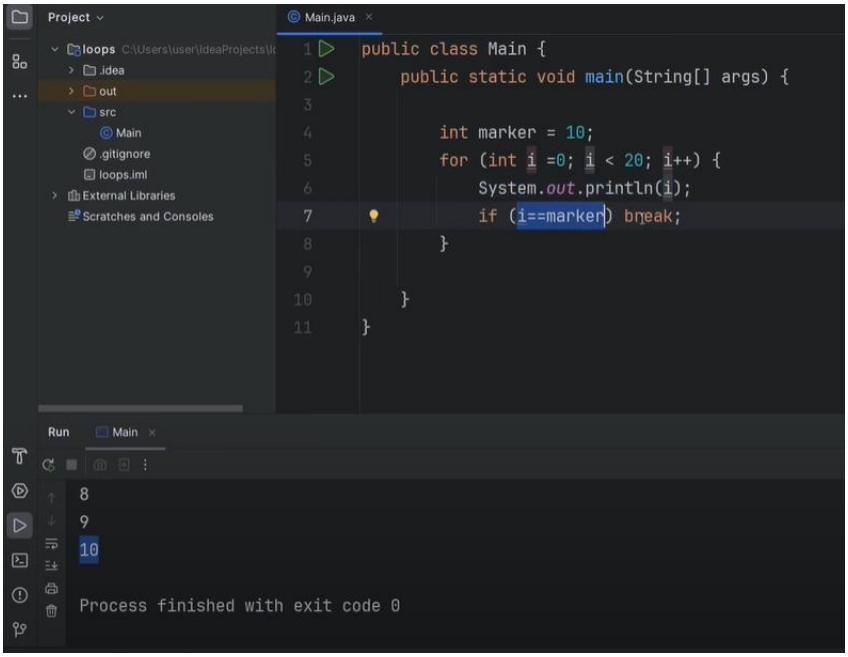
Тема 8: Оператори break, continue.

Оператори break та continue в першу чергу призначені для більш тонких налаштувань у циклах. Завдяки їм у нас є можливість або аварійно завершити цикл **(break)**, або пропустити команди нижче, але продовжити виконання циклу **(continue)**.

Розглянемо код:

```
4       int marker = 10;
5       for (int i =0; i < 20; i++) {
6           System.out.println(i);
7           if (i==marker) break;
8       }
9
10  }
11  }
```

у параметрах циклу стоїть умова: поки i<20. І на перший погляд складається враження, що програма 20 разів виведе на екран змінну i, але надруковано буде і до 10. Тому, що у рядку 7 в нас є перевірка, якщо i==marker, тоді ми аварійно завершуємо виконання циклу.



Тепер уважніше подивімось на сьомий рядок, де бачимо два знаки дорівнює.

З’ясуймо, чому ж їх два.

Насправді, знак дорівнює використовується у двох випадках в математиці.

І значення: коли ми копіюємо значення, наприклад m= 4, це означає, що ми четвірку скопіювали у змінну m і вона тепер має значення 4.

ІІ значення: якщо x=0, при використанні слова “якщо” в математиці ми не копіюємо значення, ми його порівнюємо.

Процесор не вміє визначати за змістом, як ми в математиці, тому для нього придумали у програмуванні два варіанти використання знака рівності:

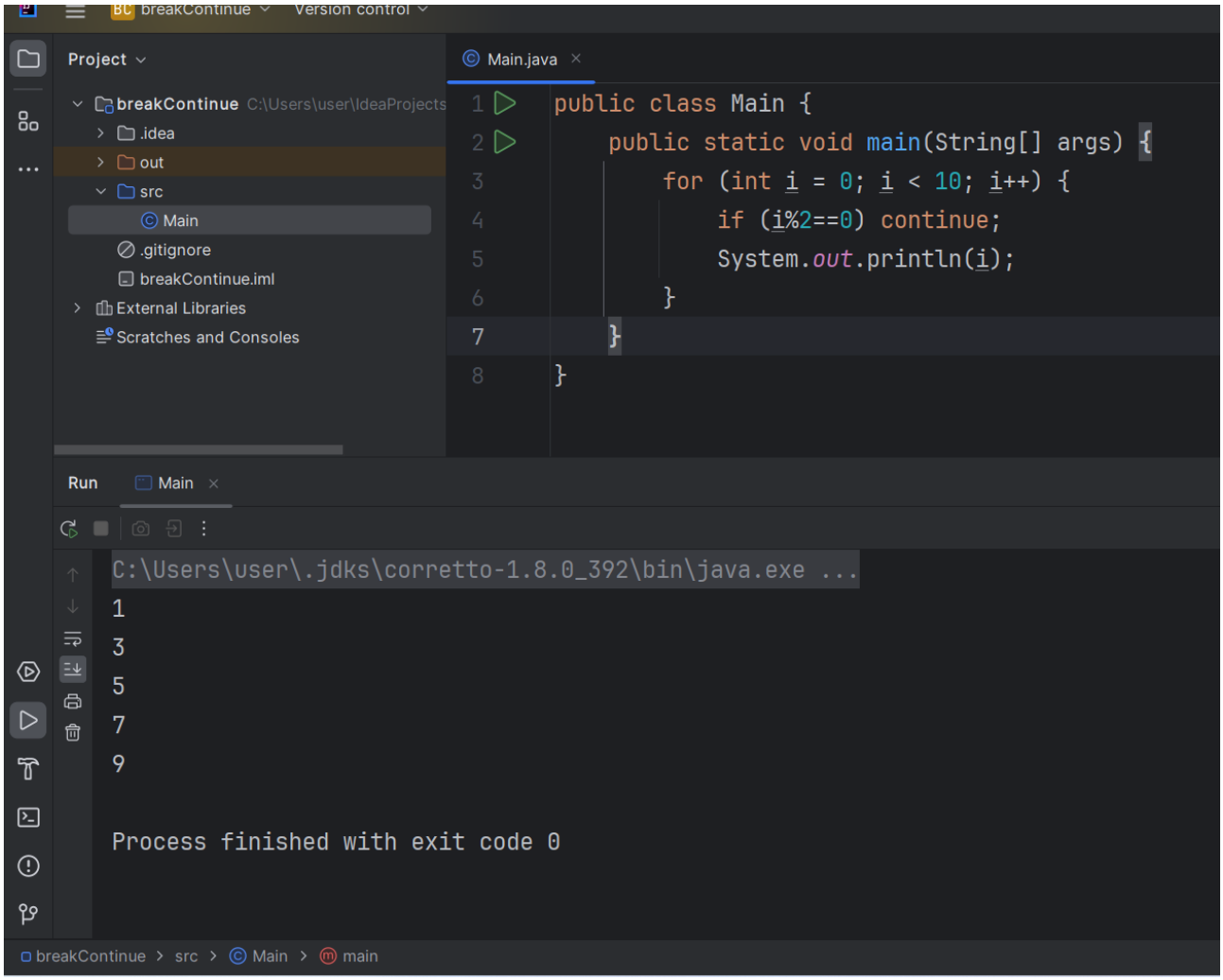
один знак = це копіювання з правої частини в ліву, наприклад x=10;

два знаки == це порівняння, наприклад if(x==0).

Отже, оператор break аварійно завершує роботу циклу.

Тепер познайомимось з оператором continue;

Розглянемо наступний код:



У третьому рядку ми створюємо цикл для змінної i, яка змінюється від 0 до 10, кожен раз збільшується на одиницю.

У тілі цикла виводимо на екран змінну i у п’ятому рядку, але перед цим є рядок 4, в якому ми перевіряємо число на парність. І якщо число парне, тобто ділиться на 2 без залишку, то виконується оператор continue, який пропускає всі команди нижче. Далі лічильник збільшується на одиницю і продовжується виконання циклу. В результаті всі кратні числа будут пропущені при виводі на екран, що ми і бачимо знизу екрану в консолі: 1 3 5 7 9.

Тепер докладніше поговоримо про рядок 4, а саме вираз `if (i%2==0)`

% означає залишок від ділення.

Уявіть собі, до вас прийшов ваш друг, у вас є 5 смачних чебуреків. Для того, щоб ніхто не образився, треба кожному в тарілку покласти по 2 чебуреки, а один залишиться чекати своєї долі. Мабуть, саме ви вирішите його долю після того, як друг піде додому. Тож ми можемо записати такий вираз:

5%2=1

якщо ж чебуреків буде 6, тоді залишиться 0:

6%2=0

Саме так працює залишок від ділення у Java.

Отже, оператор continue пропускає наступні рядки коду, але продовжує виконання циклу на відміну від оператору break.

Тема 9: Масиви.

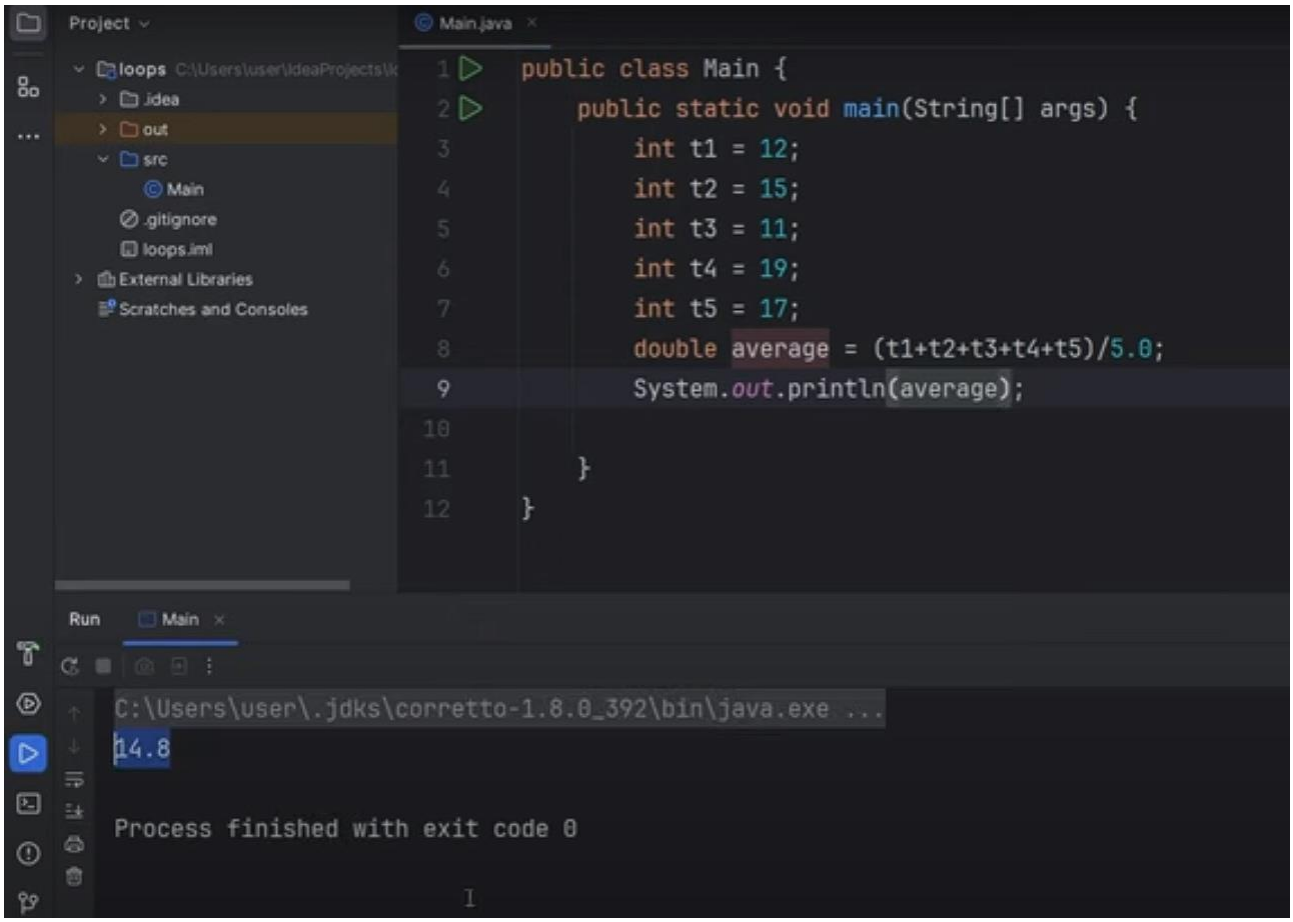
Більшість людей, коли чують слово “масив” уявляють щось велике і потужне, але це не зовсім так. У житті ми часто чуємо словосполучення “житловий масив”, під ним розуміють багато будинків, які були побудовані приблизно в один час, при цьому вони дуже схожі між собою.

У програмуванні постійно доведеться працювати з великими наборами даних. І всі вони розташовані у масивах, тобто робота за масивами - це головне завдання програміста.

Тепер розглянемо, чому масиви такі популярні і в чому їх перевага перед простими змінними.

У початкових класах учні іноді ведуть щоденник температур, де кожен день записують, яка температура була цього дня. Давайте зробимо програму, яка буде обчислювати середню температуру за тиждень, для цього треба скласти всі значення температур за кожен день і поділити на кількість днів.

Без використання масивів програма буде виглядати так:



3-7 рядок ми оголошуємо змінні t1-t5 та заносимо туди значення температур.

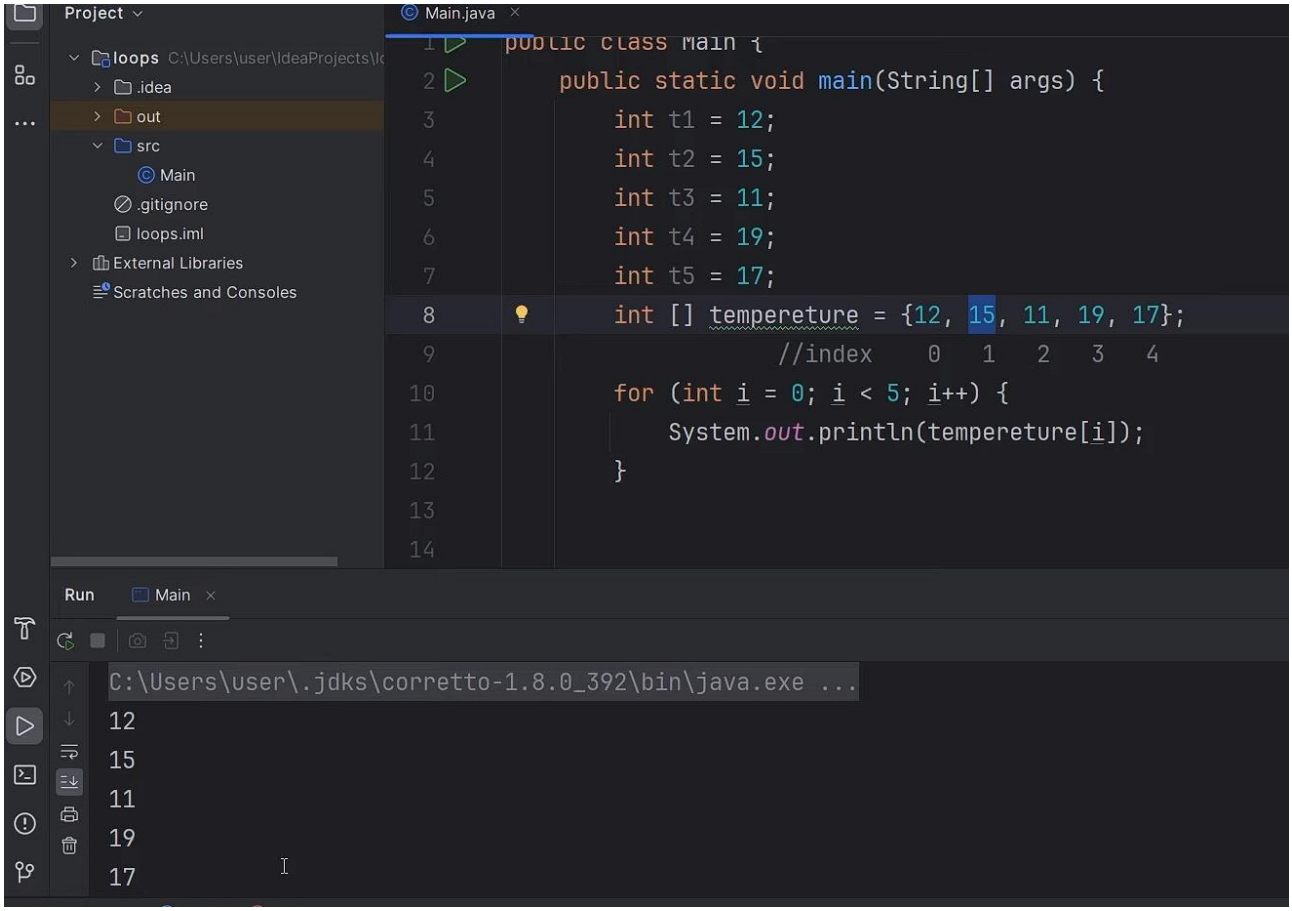
У 8 рядку оголошуємо змінну average, що перекладається як середнє значення. Ви можете вигадати власну назву змінній, але бажано, щоб назва змінної пояснювала, за що ця змінна відповідає. В цю змінну ми покладемо обчислення середнього значення. Зверніть увагу на те, що ділимо ми на 5.0. Крапка та нуль увімкнуть арифметико-логічний пристрій, який відповідає за більш точні обчислення, тому що нам потрібна більш висока точність, ніж просто цілі числа int.

У 9 рядку ми виводимо на екран середнє значення 14.8.

Ця задача підходить тільки для робочого тижня, якщо нам треба обчислити середню температуру за місяць або взагалі за рік, то треба переробляти весь код, причому, цікаво, в який момент часу вам набридне додавати ці змінні, якщо треба обчислити середню температуру за рік.

Отже, нам треба якось автоматизувати процес додавання.

Тепер поглянемо на код з використанням масиву.



У восьмому рядку ми оголошуємо масив, перша відмінність від оголошення змінної полягає у квадратних дужках, які ми ставимо перед назвою масиву, друга відмінність полягає у фігурних дужках, у яких ми перелічуємо всі значення елементів масиву.

Рядок 8 замінює собою рядки з 3 по 7, ми в одному рядку створили п’ять змінних, які об’єднані у масиві temperature

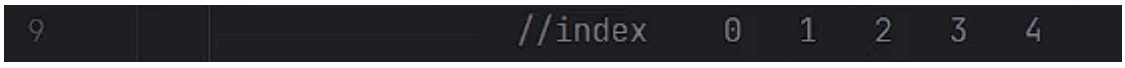
Для того, щоб виконувати операції зі змінними у цьому масиві, до них можна звертатись за назвою масиву та індексом (індекс - це порядковий номер у масиві).

Тобто для того, щоб вивести на екран перший елемент масиву, в нас це число 12, треба написати

System.out.println(temperature[0]);

Зверніть увагу, що нумерація у програмуванні починається з нуля.

У дев’ятому рядку ми зустрічаємо такий запис:



Два слеша (слеш - знак ділення) означають коментар, все що написано після // буде проігноровано при запуску програми, так програмісти пишуть собі коментарі, наприклад що треба зробити пізніше. Це наче чернетка.

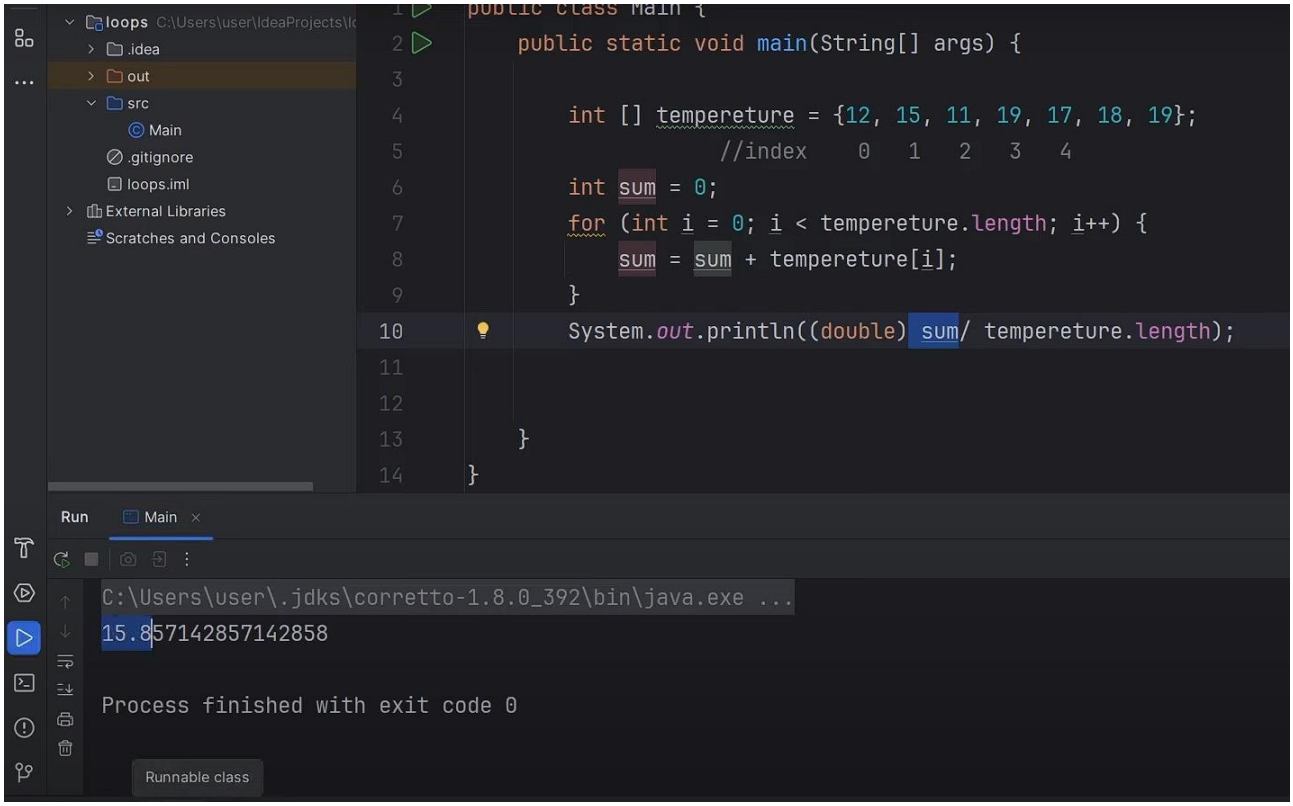
Я зробив цей коментар, щоб легше зрозуміти, який індекс має кожний елемент масиву.

У 10 рядку починається цикл, де ми по черзі будемо виводити всі елементи масиву, за рахунок того, що лічильник і буде змінюватися від 0 до 4, ми виведемо всі елементи масиву.

До речі змінна лічильника у циклі for, має назву i, саме через те, що частіше вона використовується саме для позначення індексу.

При виконанні цього коду ми виведемо на екран всі елементи масиву.

А тепер напишемо код, який буде обчислювати середнє значення за допомогою масиву:



Видалимо попередні рядки, де ми оголошуємо змінні, вони нам тепер не потрібні.

У шостому рядку оголошуємо змінну sum, вона нам потрібна для додавання всіх елементів масиву.

У сьомому рядку, зверніть увагу на умову виконання масиву в дужках:

```
i < tempereture.length;
```

Слово length перекладається як “довжина”, перед ним написано ім’я масиву, отже, можна перекласти, як довжина масиву. Ця команда обчислить кількість елементів у масиві та підставить в умову, такий варіант має перевагу, на відміну від конкретного числа, тому що в процесі роботи може виявитися, що дані змінилися, наприклад, масив став більшим, тоді цей варіант

```
i < tempereture.length;
```

завжди буде актуальним, тому що він завжди буде підставляти правильне значення кількості елементів у масиві.

У восьмому рядку ми у змінну sum по черзі додаємо всі елементи масиву

І у 10 рядку ми виводимо середнє значення температур.

Зверніть увагу на кастінг до double у десятому рядку

без кастінга буде задіяний арфіметико-логічний пристрій призначений для цілих чисел, і результат буде не точний.

Вітаю! Ви, маю надію, зрозуміли основні принципи структурного програмування, вони однакові в усіх мовах програмування.

Наступний крок - це об’єктно орієнтоване програмування, воно вже відрізняється у різних мовах програмування, але принципи все одно дуже схожі. В наступній частині ми з вами познайомимось з ними та зрозуміємо навіщо придумали ООП і які у нього переваги.

Чекаю вас на наступному кроці. Сил вам та наснаги!

Тестове опитування для самоконтролю №3

1. **Що таке вкладений цикл?**

a) Цикл, який використовується для повторення блоку коду задану кількість разів.

b) Цикл, який знаходиться всередині іншого циклу.

c) Цикл, який використовується для перебору елементів масиву.

d) Цикл, який використовується для обчислення суми чисел.

2. **Наведіть приклад завдання, яке можна вирішити за допомогою вкладених циклів.**

a) Вивести на екран трикутник з зірочок.

b) Обчислити суму всіх чисел від 1 до 100.

c) Знайти максимальне значення в масиві.

d) Відсортувати масив чисел.

3. **Чому в прикладі з трикутником використовується System.out.print(“*”); замість System.out.println(“”);?**

a) Щоб зірочки друкувалися в одному рядку.

b) Щоб зірочки друкувалися на новому рядку.

c) Щоб зірочки друкувалися через пробіл.

d) Щоб зірочки друкувалися в центрі екрана.

4. **Який оператор використовується для аварійного завершення циклу?**

a) Continue

b) Break

c) For

d) While

5. Який оператор використовується для пропуску наступних команд в циклі?

a) Continue

b) Break

c) For

d) While

6 Що таке масив?

a) Змінна, яка може зберігати одне значення.

b) Змінна, яка може зберігати кілька значень одного типу.

c) Змінна, яка може зберігати значення різних типів.

d) Змінна, яка використовується для зберігання тексту.

Підсумкове тестове опитування

1. Що таке алгоритм?

(A) Набір інструкцій, що описують послідовність дій для виконання певного завдання.

(B) Програма, написана мовою програмування.

(C) Тип даних, який використовується для зберігання чисел.

(D) Оператор, який використовується для порівняння двох значень.

2. Які типи даних ви знаєте?

(A) Цілі числа

(B) Дійсні числа

(C) Символи

(D) Всі вищезазначені

3. Що таке змінна?

(A) Іменована область пам'яті, яка використовується для зберігання значення.

(B) Ключове слово, яке використовується для оголошення змінної.

(C) Тип даних, який використовується для зберігання даних.

(D) Оператор, який використовується для присвоєння значення змінній.

4. Що таке оператор?

(A) Символ або слово, яке використовується для виконання певної дії.

(B) Ключове слово, яке використовується для оголошення циклу.

(C) Тип даних, який використовується для зберігання логічних значень.

(D) Оператор, який використовується для порівняння двох значень.

5. Що таке масив?

(A) Впорядкована колекція даних одного типу.

(B) Ключове слово, яке використовується для оголошення масиву.

(C) Тип даних, який використовується для зберігання символів.

(D) Оператор, який використовується для доступу до елемента масиву.

6 Який з наступних кроків НЕ є необхідним для встановлення IntelliJ IDEA?

(a) Завантажити дистрибутив з офіційного сайту.

(b) Запустити інсталятор та прийняти умови ліцензії.

(c) Вибрати тип інсталяції (типова, мінімальна, custom).

(d) Створити новий проект Java.

7. Який код потрібно написати, щоб вивести на екран рядок "Hello, World!"?

(a) System.out.println("Hello, World!");

(b) System.out.print("Hello, World!");

(c) System.out.write("Hello, World!");

(d) System.out.writeLine("Hello, World!");

8. Який тип даних використовується для зберігання цілих чисел в Java?

(a) byte.

(b) short.

(c) int.

(d) long.

9. Який результат виразу 25 / 15?

(a) 1.6667.

(b) 1.

(c) 2.

(d) 3.

10. Як правильно виконати ділення двох цілих чисел з плаваючою комою?

(a) Використовувати оператор /.

(b) Перетворити один з операндів в тип double.

(c) Використовувати метод Math.round().

(d) Використовувати метод Math.floor().

11. Напишіть код для обчислення суми двох чисел, введених користувачем.

(a) int sum = a + b;

(b) System.out.println("Введіть два числа:");

(c) Scanner scanner = new Scanner(System.in);

(d) int a = scanner.nextInt();

12. Який синтаксис тернарного оператора if в Java?

(a) умова ? результат_якщо_true : результат_якщо_false

(b) if (умова) { результат_якщо_true } else { результат_якщо_false }

(c) if (умова) результат_якщо_true

(d) if (умова) : результат_якщо_true

Шкала оцінювання

Максимальна кількість балів за один тест - 12 балів

Результат навчання визначається формулою:

(тест1+тест2+тест3+підсумкове опитування)/5

Максимальна оцінка з курсу - 12 балів

Тест	Ціна питання	Кількість балів
Тестове опитування для самоконтролю №1	Одна правильна відповідь = 2 бали	
Тестове опитування для самоконтролю №2	Одна правильна відповідь = 2 бали	
Тестове опитування для самоконтролю №3	Одна правильна відповідь = 2 бали	
Підсумкове опитування	Одна правильна відповідь = 2 бали	

Для закріплення знань пропонується пройти інтерактивну гру на платформі Quiziz за посиланням:

https://quizizz.com/join?gc=90563487