Вступ

Програмування мовою Java

Я хочу робити все те саме, що й ти.
Ти — мій кумир.

Телесеріал «Альф»

Серед мов програмування мова Java — найпопулярніша й найбільш затребувана. Ця книга про те, як програмувати мовою Java.

Особливості мови Java

Погано на цій планеті з почуттям гумору.

Т/с «Альф»

Історія мови Java почалась в 1990-х роках, коли групі інженерів компанії Sun Microsystems у межах проекту під назвою «Green» було поставлено завдання з розробки універсальної, компактної і незалежної від конкретної платформи мови програмування Oak, призначеної для використання в побутових пристроях. У процесі реалізації проекту змінилися не тільки основні пріоритети, але й назва мови програмування. Як би там не було, в 1995 році світ познайомився з мовою програмування Java.

З моменту появи першої версії Java було декілька оновлень платформи. На момент написання книги актуальною є версія Java 10. Однак слід мати на увазі, що версії Java 9 та Java 10 підтримуються лише для 64-розрядних операційних систем. Тому актуальною залишається і версія Java 8, яка дозволяє працювати з 32-розрядними системами. Приклади для книги підбиралися відповідно до принципу оберненої сумісності, щоб програмні коди виконувались у всіх версіях, починаючи з Java 8. Методи та прийоми,

характерні для версій Java 9 або Java 10, виділені коментарями та супроводжуються поясненнями.

Мова Java, хоч і не без труднощів, але завоювала своє «місце під сонцем». Сьогодні Java міцно утримує позиції найбільш затребуваної мови програмування. Успіху мови сприяв бурхливий розвиток Internet-технологій. Справа в тім, що для Java-програм характерним є високий ступінь універсальності й незалежності від апаратного забезпечення. Це важливо у випадку створення програмного забезпечення, орієнтованого на роботу в мережі, оскільки кінцеві користувачі використовують різні операційні системи і різне апаратне забезпечення. До того ж не останню роль відіграла застосовність мови Java для програмування всіляких мобільних пристроїв. Тому немає нічого дивного в тому, що значна доля комерційних і вільно поширюваних програм написані на мові Java. Відповідно, попит на програмістів, які працюють з мовою Java, стабільно високий, а загальні тенденції такі, що він залишиться високим і найближчим часом.

Універсальність програм, написаних мовою Java, базується на використанні ***віртуальної машини***. Це такий специфічний «посередник», під керівництвом якого виконується байт-код, який створюється під час компіляції програми. Тут необхідні пояснення.

Після того, як програма написана, вона компілюється. Зазвичай в результаті компіляції програми створюється виконавчий файл з машинним кодом, який і виконується, коли необхідно виконати програму. Простіше кажучи, під час компіляції команди, зрозумілі для програміста, переводяться на «мову», зрозумілу для комп'ютера. Якщо мова йде про програму, написану мовою Java, то все відбувається схожим чином, але з деякими особливостями. Найважливіше те, що в результаті компіляції Java-програми створюється не машинний код, а проміжний ***байт-код***. Це дещо середнє між машинним кодом і кодом програми. Якщо машинний код, як правило, виконується під управлінням операційної системи, то байт-код виконується під управлінням спеціальної програми, яка називається віртуальною машиною (або віртуальною Java-машиною). Зрозуміло, що таку програму на комп'ютер треба заздалегідь установити.

Виникає запитання: а в чому ж виграш від використання віртуальної машини і як усе описане впливає на універсальність кодів? Виграш у тому, що під час написання коду можна абстрагуватися від особливостей

операційної системи і апаратного забезпечення, які використовуються кінцевим користувачем. Ці особливості враховуються — але враховуються на рівні віртуальної машини. Саме віртуальна машина під час виконання байт-коду «бере до уваги» особливості операційної системи й апаратного забезпечення комп'ютера, на якому виконується програма.

Припустимо, існує програма, написана мовою C++. У процесі її компіляції створюється машинний код, який для різних операційних систем буде різним. Якщо компілюється програма, написана мовою Java, то створений в результаті байт-код не залежатиме від операційної системи, котра встановлена на комп'ютері — він буде одним і тим самим для різних операційних систем. Але віртуальна машина для кожної операційної системи своя. Відмінність в операційних системах «враховується», коли на комп'ютер установлюється віртуальна машина.

Описаний вище механізм, загалом, забезпечує високий ступінь універсальності програм, написаних на мові Java. Особливо це помітно у випадку створення програм із графічним інтерфейсом.

Забігаючи наперед, відмітимо, що у плані створення прикладних програм із графічним інтерфейсом мова Java особливо ефектна.

Існує ще один важливий аспект, який стосується мови Java, на який одразу звертаємо увагу. Мова Java — повністю ***об'єктно-орієнтована*** мова. Сказане означає, що для написання навіть найменшої і найпростішої програми доведеться описати, щонайменше, один ***клас***. Це автоматично створює деякі труднощі в освоєнні премудростей мови Java. Особливо важко тим, хто не має досвіду програмування. Адже фактично відразу, з перших кроків, доводиться знайомитися з концепцією ***об'єктно-орієнтованого програмування*** (скорочено ООП), яка, слід визнати, не є тривіальною. Однак панікувати не варто — ми знайдемо спосіб донести необхідні відомості навіть до найбільш непідготовлених читачів. Головне, аби було бажання засвоїти мову Java.

Програмне забезпечення

— Кейт, мені потрібна твоя порада.
— Ось тобі моя порада: більше ніколи так не роби!

Т/с «Альф»

Якщо підійти до питання формально, то сама по собі мова Java — це набір правил, відповідно до яких формується програмний код. Однак програми пишуться задля того, щоб вони виконувались. А якщо так, то нам знадобиться спеціальне програмне забезпечення. Гарна новина в тому, що все необхідне програмне забезпечення можна отримати цілком вільно, просто і безкоштовно.

Зрозуміло, що існують і комерційні прикладні програми, призначені для написання програм на мові Java. Але для вирішення тих завдань, які ми ставимо перед собою, стандартного вільно поширюваного програмного забезпечення більш ніж достатньо.

Що ж нам знадобиться? У принципі, можемо обмежитись мінімальними засобами у вигляді пакета прикладних програм JDK (скорочення від ***Java Development Kit*** — засоби розробки Java). У склад пакета JDK, крім іншого, входить компілятор, усілякі бібліотеки, документація і виконавча система JRE (скорочення від ***Java Runtime Environment*** — середовище виконання Java) — фактично, віртуальна машина мови Java. Пакет прикладних програм JDK поширюється безкоштовно компанією Oracle (сайт компанії `www.oracle.com`).

Свого часу розробника мови Java, компанію Sun Microsystems, поглинула корпорація Oracle. Так що тепер підтримкою Java-технологій займається саме вона.

Ситуація така, що без пакета JDK нам не обійтися, але й обмежуватися тільки пакетом JDK не варто. Якщо обмежитися тільки пакетом JDK, то програмні коди доведеться набирати в текстовому редакторі, а компілювати програму доведеться «вручну» з командного рядка. Тому бажано використати ***середовище розробки*** (скорочено IDE від *Integrated Development Environment*).

Середовище розробки містить редактор кодів, наладник, який дозволяє в інтерактивному режимі відстежувати код на наявність синтаксичних помилок, набір інших утиліт, які дозволяють зробити процес написання, тестування і компіляції програм простим, зручним і десь навіть комфортним (наскільки це взагалі можливо). Простіше кажучи, середовище розробки має використовуватись — тим більше, якщо врахувати, що наявними є досить пристойні безкоштовно поширювані середовища розробки. Ми зупинимо свій вибір на середовищі розробки IntelliJ IDEA. Середовище поширюється безкоштовно, файли для його установки можна завантажити на сайті підтримки проекту `www.jetbrains.com`.

Далі коротко розглянемо, яке програмне забезпечення і звідки слід завантажити перед тим, як ми безпосередньо перейдемо до вивчення мови програмування Java.

Завдання наше просте:

- необхідно завантажити й установити пакет прикладних програм JDK;
- після встановлення JDK слід завантажити й установити середовище розробки IntelliJ IDEA.

Дії з завантаження й установки програмного забезпечення виконуються саме в тій послідовності, в якій вони перераховані вище.

Середовище розробки IntelliJ IDEA у процесі роботи з програмними кодами звертається до системи JDK. Якщо систему JDK встановити до установки IntelliJ IDEA, то всі налаштування середовища розробки, пов'язані з JDK, скоріш за все, будуть виконані автоматично. Якщо систему JDK встановлювати після установки середовища розробки IntelliJ IDEA, то налаштування середовища розробки доведеться виконувати самостійно (хоча це й не так складно).

Отже, в першу чергу встановлюємо пакет JDK, для чого попередньо з сайту компанії Oracle завантажуємо установочні файли. На рис. В.1 показано вікно браузера, відкрите на сторінці `www.oracle.com`.

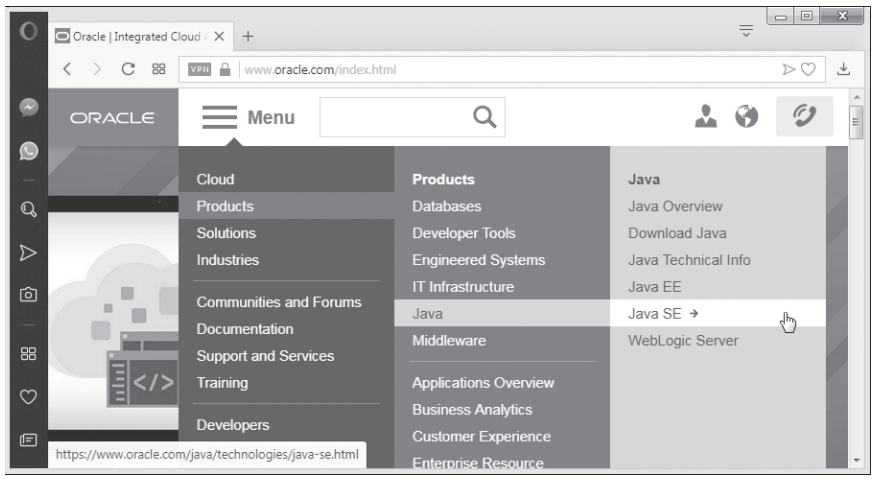

Рис. В.1. Вікно браузера, відкрите на сторінці `www.oracle.com` корпорації Oracle

На сайті потрібно знайти сторінку для завантаження програмного забезпечення для роботи з Java (як, наприклад, показано на рис. В.2).

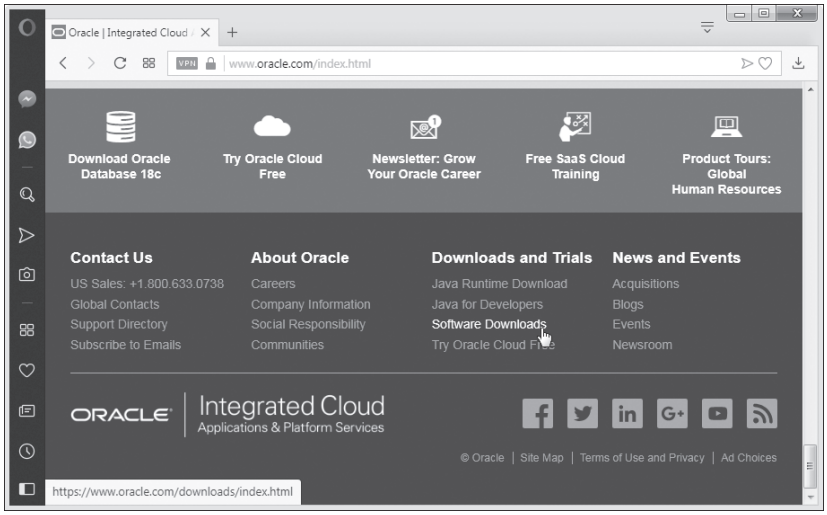

Рис. В.2. Гіперпосилання для завантаження програмного забезпечення для роботи з Java на сторінці `www.oracle.com`

Існує декілька редакцій для дистрибутивів Java. Наприклад, платформа Java для створення програмного забезпечення рівня великих корпорацій називається Java Enterprise Edition (скорочено Java EE). Стандартна редакція Java призначена для створення прикладних програм користувача і називається Java Standard Edition (скорочено Java SE). Також існує редакція Java Micro Edition (скорочено Java ME), яка використовується у випадку створення прикладних програм для усіляких мобільних пристроїв. Ми будемо використовувати стандартну редакцію Java Standard Edition (або Java SE).

Після кліку на гіперпосиланні для завантаження програмного забезпечення для роботи з Java переходимо до ще одної сторінки, з якої власне і виконується завантаження (рис. В.3).

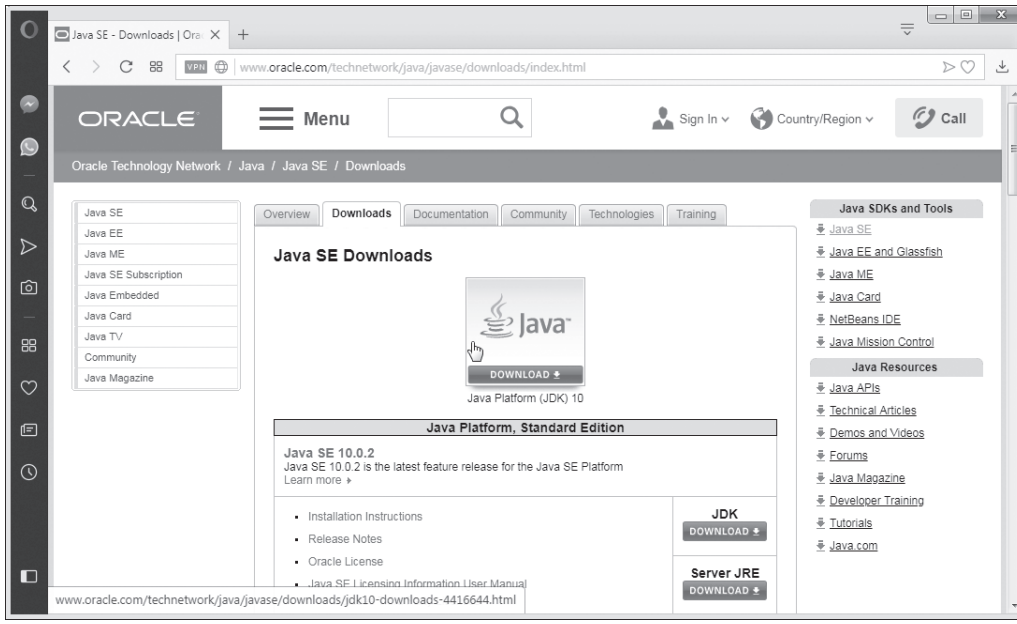

Рис. В.3. Вікно браузера, відкрите на сторінці завантаження установочного файлу пакета JDK

У процесі завантаження пропонується вибрати тип установочного файлу відповідно до того, яка операційна система використовується. Ситуація проілюстрована на рис. В.4.

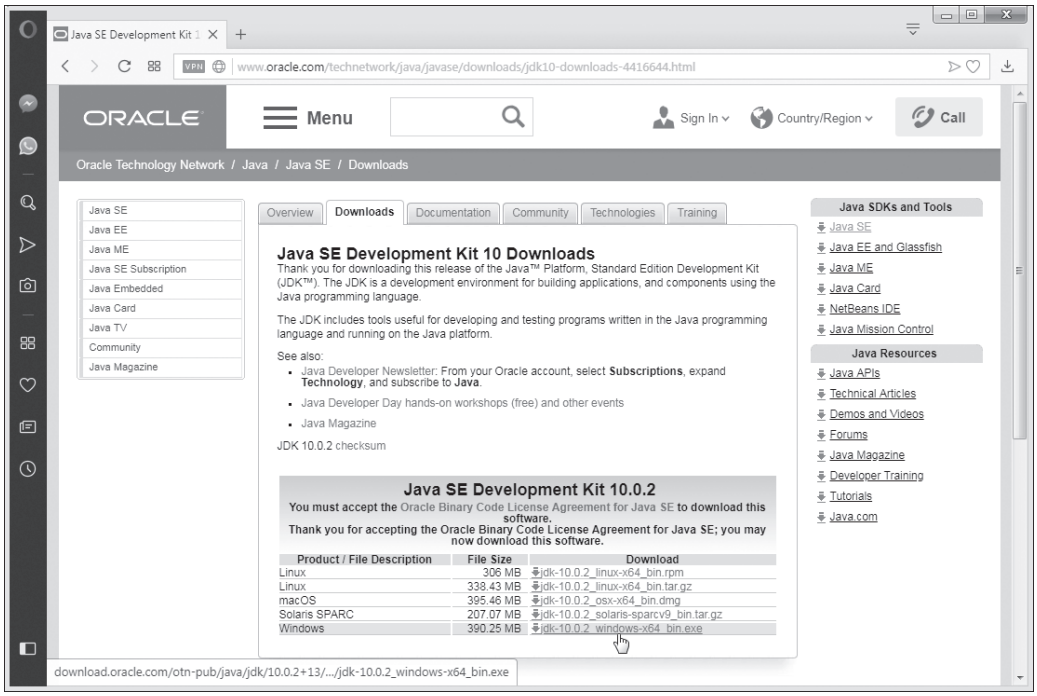

Рис. В.4. Вибір установочного файлу відповідно до того, яка операційна система використовується

Зовнішній вигляд сайтів, в тому числі й сайт корпорації Oracle, час від часу змінюється, тому не виключено, що для пошуку сторінки завантаження програмного забезпечення доведеться виявити деяку винахідливість.

Після завантаження й установки пакета JDK слід завантажити ще й установочний файл для середовища розробки IntelliJ IDEA. У цьому випадку переходимо на сторінку `www.jetbrains.com` проекту JetBrains, як показано на рис. В.5.

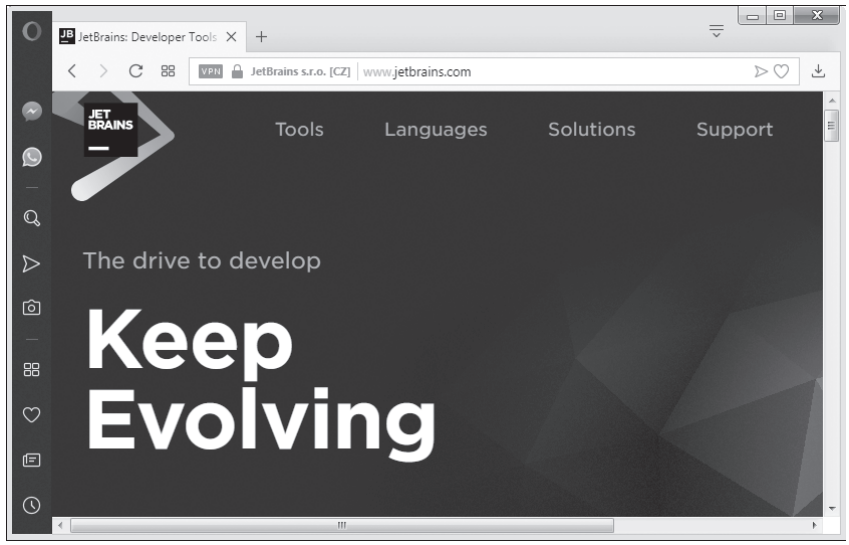

Рис. В.5. Сторінка `www.jetbrains.com` *проекту Jet Brains*

Потім переходимо на сторінку завантаження середовища IntelliJ IDEA, для чого вибираємо відповідний пункт у розділі **Tools**, як показано на рис. В.6.

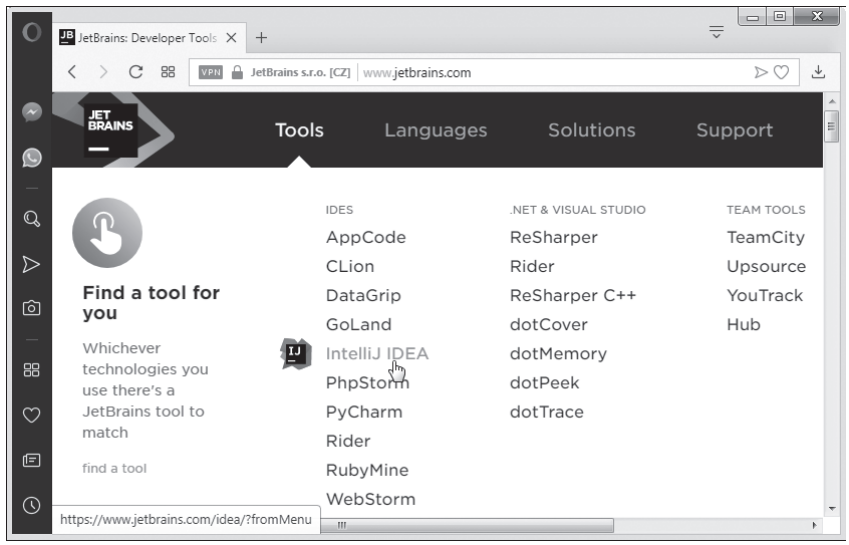

Рис. В.6. Перехід на сторінку для завантаження середовища розробки IntelliJ IDEA

Як виглядає сторінка, на якій можна завантажити установочний файл середовища розробки IntelliJ IDEA, показано на рис. В.7.

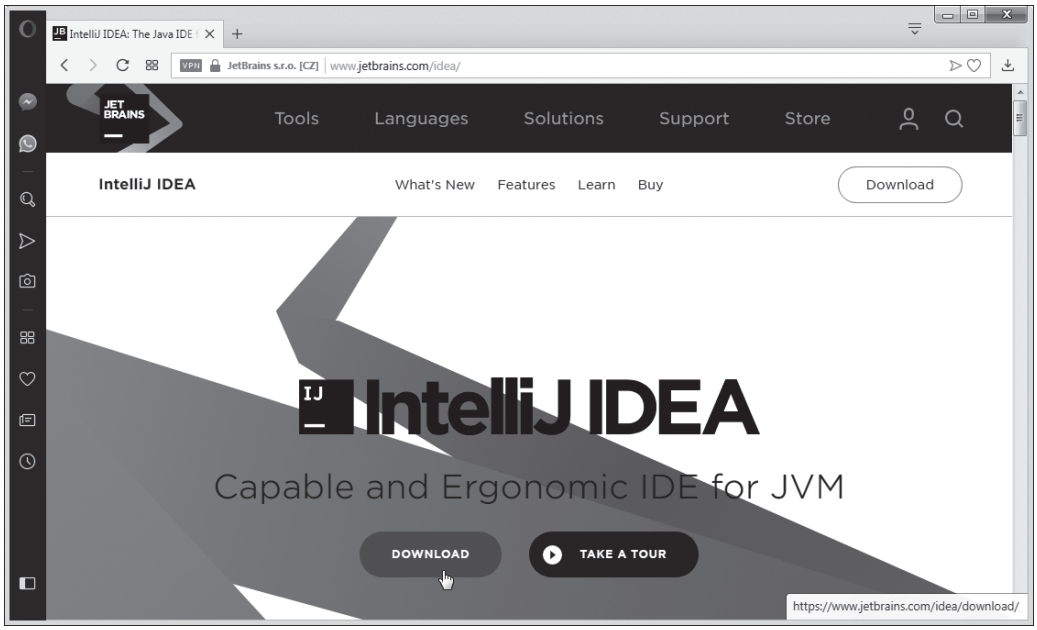

Рис. В.7. Сторінка `www.jetbrains.com/idea` для завантаження установочного файлу для середовища розробки IntelliJ IDEA

Завантажуємо установочний файл і встановлюємо середовище розробки IntelliJ IDEA. На цьому попередня підготовка до написання програм на Java завершується. Зазначимо лише, що ще одна корисна сторінка знаходиться за адресою `www.java.com`. На рис. В.8 показано вікно браузера, відкрите на цій сторінці.

Рис. В.8. Вікно браузера, відкрите на сторінці `www.java.com` *підтримки Java*

На сторінці можна завантажити оновлення платформи Java, які з'являються досить часто.

Середовище розробки IntelliJ IDEA

— Гарно?
— Так, так. Привабливо.

Т/с «Альф»

Середовище розробки IntelliJ IDEA компанії JetBrains є популярним і зручним засобом створення проектів мовою Java, тому доцільно приділити цьому середовищу хоча б трохи уваги. На рис. В.9 показане вікно, яке з'являється під час запуску середовища IntelliJ IDEA.

Рис. В.9. Стартове вікно середовища розробки IntelliJ IDEA

Якщо в середовищі вже створювались проекти, то в лівій частині вікна буде відображатися список таких проектів. Щоб відкрити проект, достатньо вибрати його в цьому списку.

Узагалі, особливості роботи з прикладною програмою IntelliJ IDEA можуть бути предметом окремої книги. В наші плани це не входить. Ми розглянемо лише деякі найбільш важливі операції, котрі читачеві доведеться виконувати у процесі написання програмних кодів. Щоб не відволікатися в основній частині книги на пояснення маніпуляцій з елементами інтерфейсу середовища IntelliJ IDEA, зробимо це заздалегідь. А саме, цікавість викликають такі дії:

- створення нового проекту (виконується у випадку написання нової програми);
- компіляція і запуск на виконання програми;
- закриття відкритого проекту (програми);
- відкриття вже існуючого проекту.

Далі ми розглянемо основні операції, які доведеться виконувати в середовищі IntelliJ IDEA, якщо читач вибере його для роботи. Насамперед, інтерес викликає процедура створення нового проекту. Щоб створити новий проект, слід натиснути піктограму **Create New Project** (див. рис. В.9). Відкриється ще одне вікно, яке називається **New Project**. Воно показане на рис. В.10.

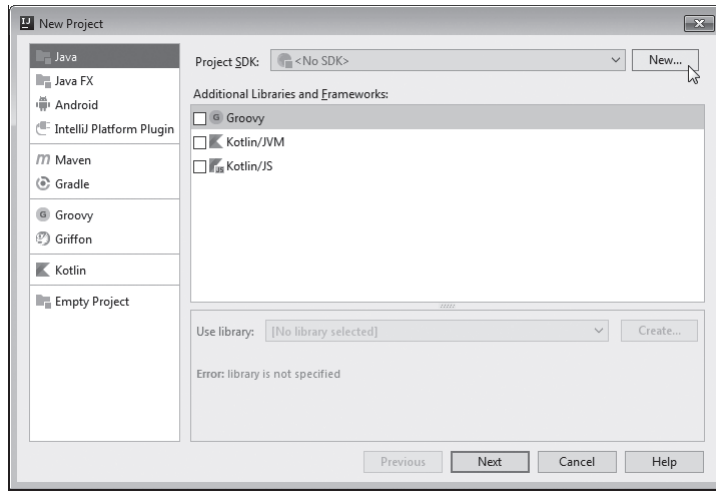

Рис. В.10. Вікно з налаштуваннями для створення нового проекту

У списку зліва вибираємо позицію **Java**. Але найголовніше — в динамічному списку **Project SDK** має бути вибрана система SDK (від Software Development Kit, але фактично це те саме, що JDK). Як правило, вона визначається автоматично, але іноді доводиться задавати її вручну. Для цього вибираємо систему SDK з динамічного списку (якщо вона там є), або натискаємо кнопку **New** справа від динамічного списку (див. рис. В.10). Далі потрібно знайти місце, куди встановлювалась Java, і вибрати відповідну папку, як показано на рис. В.11.

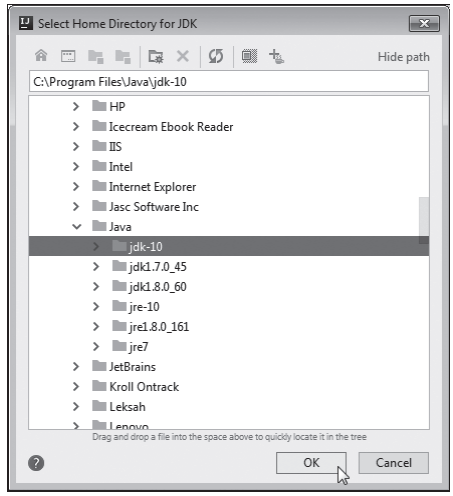

Рис. В.11. Вибір папки з установленою системою JDK

Після вибору системи SDK натискаємо кнопку **Next** (див. рис. В.10) і продовжуємо виконувати налаштування. Один із важливих етапів полягає у виборі назви проекту (поле **Project name**) і місця для зберігання файлів проекту (поле **Project location**), як показано на рис. В.12.

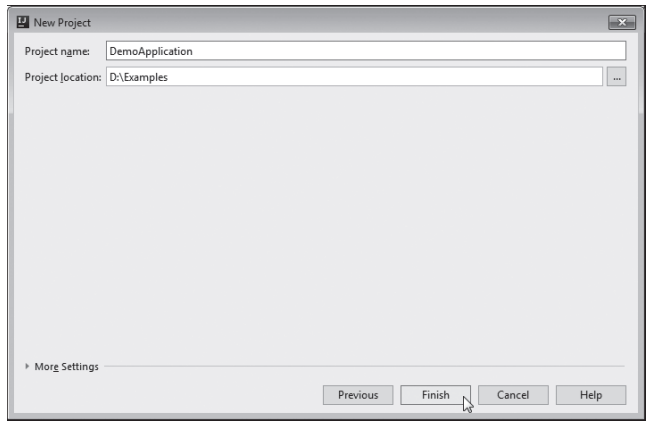

Рис. В.12. Вибір назви проекту і місця зберігання проекту

У результаті створюється новий пустий проект. Як буде в цьому випадку виглядати вікно середовища розробки IntelliJ IDEA, показано на рис. В.13.

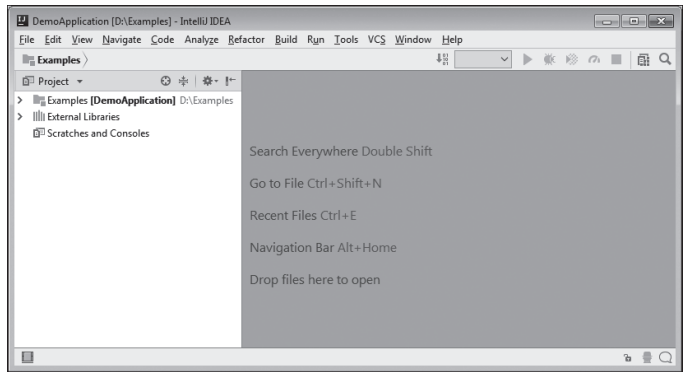

Рис. В.13. Вікно середовища розробки IntelliJ IDEA після створення нового проекту

Проект не містить нічого, крім автоматично створених допоміжних файлів. Нам необхідно добавити в проект файл із програмним кодом (код з описом головного класу програми).

Як зазначалося раніше, будь-яка програма мовою Java містить описання хоча б одного класу. Серед класів програми є хоча б один, який називається ***головним класом програми***. Клас містить код головного методу, і цей метод виконується у випадку виконання програми. Простіше кажучи, головний клас програми містить той код, який виконується у випадку виконання

програми. Взагалі, назва головного класу має співпадати з назвою файлу, який містить код головного методу.

Для додавання файлу в проект розкриваємо список із назвою проекту (область зліва з назвою **Project**), знаходимо в динамічному списку папку **src**, виділяємо її, натискаємо правою кнопкою миші і в динамічному контекстному меню, яке розкриється, в підменю **New** вибираємо команду **Java Class** (рис. В.14).

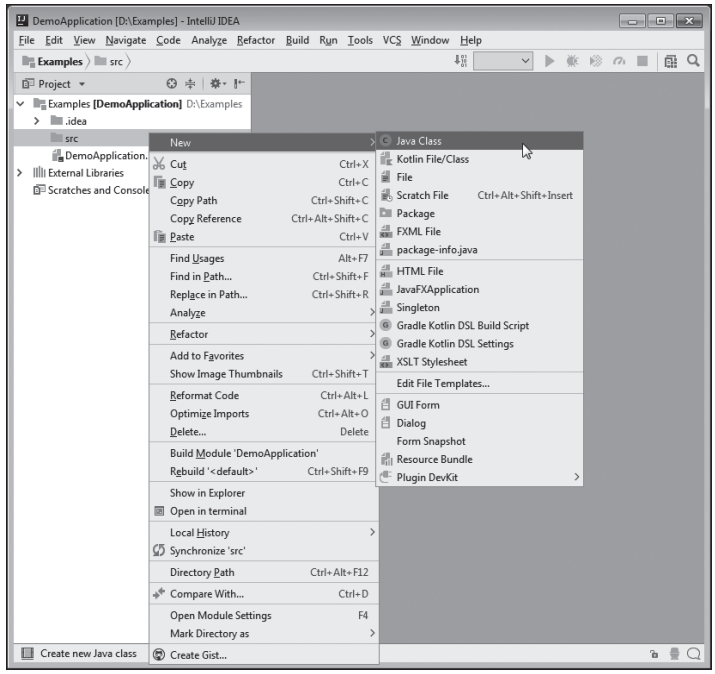

Рис. В.14. Додавання файлу з програмним кодом у проект

Замість використання контекстного меню можна скористатися головним меню **File**. Там є підменю **New**, в якому для додавання в проект файлу з програмним кодом слід вибрати команду **Java Class**. У результаті відкриється вікно **Create New Class**, представлене на рис. В.15.

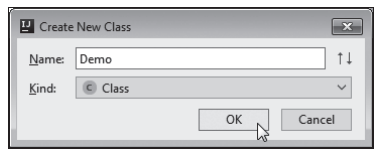

Рис. В.15. Визначення назви файлу з кодом, який додається до проекту

У полі **Name** вікна вказується назва класу (вона ж — назва файлу, в якому буде описаний цей клас). Ми для класу вказуємо назву `Demo`. Якщо все пройшло вдало, то в проект буде додано файл, в який потрібно внести програмний код. Який може мати вигляд вікно середовища розробки з доданим у проект файлом і занесеним у файл програмним кодом, показано на рис. В.16.

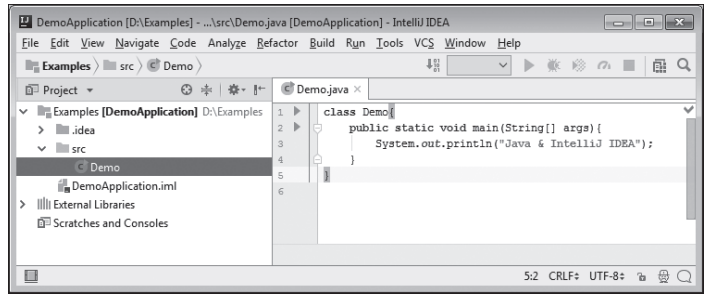

Рис. В.16. Вікно середовища розробки з файлом, який містить програмний код

Назва створеного файлу `Demo.java` відображається на корінці вкладки вікна з кодом головного класу. Код, який ми добавили у файл, представлений у лістингу В.1.

Лістинг В.1. Програмний код проекту DemoApplication для використання в середовищі IntelliJ IDEA

```
class Demo{
   public static void main(String[] args){
      System.out.println("Java & IntelliJ IDEA");
   }
}
```

Залишилось скомпілювати програму і запустити її на виконання.

На даному етапі сенс команд із наведеного вище програмного коду не такий вже й важливий. Аналізувати коди ми будемо в основній частині книги. Поки що лише відмітимо, що інструкцією `class Demo` починається описання класу. Саме описання класу розміщується у фігурних дужках `{` та `}`. У класі описується головний метод програми, який називається `main()`. В тілі методу виконується всього одна команда `System.out.println("Java & IntelliJ IDEA")`, якою у вікні виводу відображається повідомлення Java & IntelliJ IDEA.

Для компіляції і запуску програми на виконання в меню **Run** можна вибрати команду з такою самою назвою, як це показано на рис. В.17.

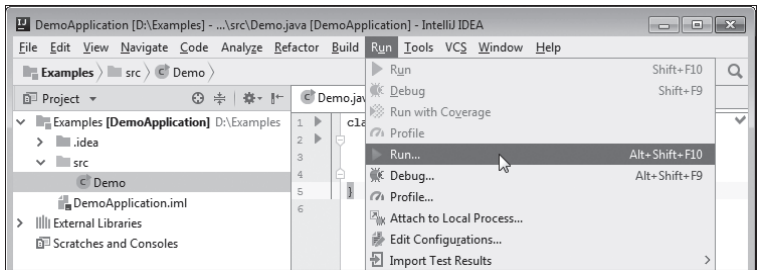

Рис. В.17. Запуск програми на виконання

Крім цього, можемо скористатися аналогічною командою з контекстного меню для файлу з кодом або спеціальною піктограмою із зеленою стрілкою зліва від опису програмного коду головного класу. Результат виконання програми наступний:

Результат виконання програми (з лістингу В.1)

```
Java & IntelliJ IDEA
```

За замовчуванням повідомлення програми відображаються в області виводу в нижній частині вікна середовища розробки. На рис. В.18 показано, як буде виглядати вікно середовища розробки після виконання програми.

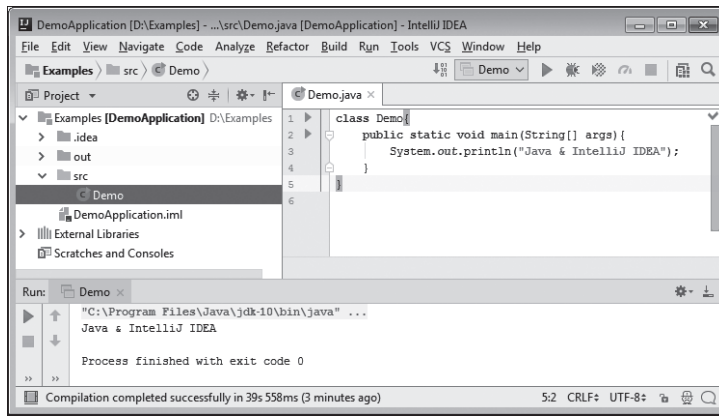

Рис. В.18. Вікно середовища розробки з результатом виконання програми

Файл з Java-програмою має розширення `.java`. В загальному випадку програма, написана мовою Java, може містити декілька класів. Під час компіляції для кожного класу створюється окремий файл. Назви таких файлів співпадають із назвами відповідних класів, а розширення у файлів `.class`.

Якщо нам в якийсь момент знадобиться закрити проект, можемо скористатися командою **Close Project** з меню **File** (рис. В.19).

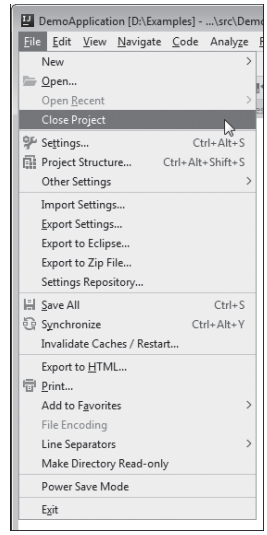

*Рис. В.19. Для закриття проекту вибираємо команду `Close Project` з меню **File***

Там же є команди для зберігання змін (**Save All**), для відкривання вже існуючого проекту (**Open**) і багато інших. Узагалі, середовище розробки IntelliJ IDEA відносно просте в роботі і досить гнучке в плані налаштувань. Хочеться вірити, що читач у випадку необхідності без труднощів опанує основні прийоми роботи з ним.

Про книгу

Мені аж не віриться. Що це, по-вашому, таке?

Т/с «Альф»

Декілька слів хочеться сказати безпосередньо про книгу. Як відмічалося зі самого початку, книга про те, як програмувати в мові Java. Ми почнемо з найпростіших речей і поступово розглянемо практично всі основні теми, які, так чи інакше, формують парадигму програмування в Java. Далі наводиться список деяких тем, які розглядаються в книзі:

- базові прийоми створення програм в Java;
- класи і об'єкти;
- перевантаження методів;
- лямбда-вирази;
- наслідування і переозначення методів;
- використання інтерфейсів;
- обробка помилкових ситуацій;
- багатопоточне програмування;
- узагальнені класи;
- створення прикладних програм із графічним інтерфейсом;
- робота з файлами.

Список не є вичерпним, так що окрім перерахованих вище, книга містить обговорення і багатьох інших проблем і механізмів.

Починаючи з версії Java 9, підтримка аплетів переведена у розряд застарілих (deprecated) технологій. І хоча формально технологія ще може використовуватися, але майбутнє у неї сумнівне. Головна причина в тому, що розробники основних браузерів відмовились (або планують відмовитись)

від використання аплетів (в основному з причин, пов'язаних із безпекою роботи браузерів). Тому в книзі аплети не розглядаються.

Для зручності засвоєння матеріалу книга розбита на відносно невеликі розділи. Кожен розділ містить приклади розв'язання різноманітних задач. У кінці кожного розділу є коротке резюме, яке покликане полегшити засвоєння матеріалу розділу. Віримо, що книга стане надійним помічником для всіх, хто вивчає мову програмування — в тому числі й для тих, хто опановує премудрості мови Java в режимі самонавчання.

Зворотний зв'язок з автором

Це вже починає трохи набридати.

Т/с «Альф»

Автор книги — ***Васильєв Олексій Миколайович***, професор кафедри теоретичної фізики Київського національного університету імені Тараса Шевченка. Автор книг із програмування та математичного моделювання. Більш детальну інформацію можна знайти на сайті `www.vasilev.kiev.ua`. Запитання і зауваження можна направляти електронною поштою `alex@vasilev.kiev.ua`.

Подяки

Цей вечір ми запишемо золотими буквами в книзі здобутків.

Т/с «Альф»

Автор висловлює щиру вдячність ***Ярославу Чолію*** і ***Володимиру Семенюку*** за допомогу в розв'язанні низки технічних питань, які виникли у процесі роботи над книгою.

Розділ 1

Починаємо програмувати

Непогано. Є ще порох у порохівницях.

Т/с «Альф»

Переходимо до вивчення мови програмування Java. Перше, що ми зробимо — напишемо невелику програму. І хоча ми ще практично нічого не дізнались про мову Java, написати першу програму нам це не завадить.

Перша програма

І все це — звичайна зубна паста.

Т/с «Альф»

Створення програми починається з визначення задачі, яка має бути розв'язана, або з окреслення мети, яка повинна бути досягнута у процесі виконання програми. Мета у нас проста: під час виконання програми повинно з'являтися діалогове вікно з текстовим повідомленням. Така задача в Java розв'язується виключно просто. Знадобиться всього декілька рядків коду.

Створення програми

Оскільки мова йде про першу програму, ми вчинимо так: спочатку створимо програму, перевіримо, як вона виконується, а уже потім проаналізуємо програмний код.

Використаємо програмний код, представлений у лістингу 1.1.

Лістинг 1.1. Програмний код проекту ShowMeAWindowApplication

```
import javax.swing.JOptionPane;
class ShowMeAWindowDemo{
   public static void main(String[] args){
      JOptionPane.showMessageDialog(null,
      "Перша програма на Java!");
   }
}
```

Код зовсім невеликий. Необхідно створити новий проект і ввести у вікно редактора програмний код з лістингу 1.1. Як в цьому випадку може виглядати вікно середовища розробки IntelliJ IDEA з кодом програми, показано на рис. 1.1.

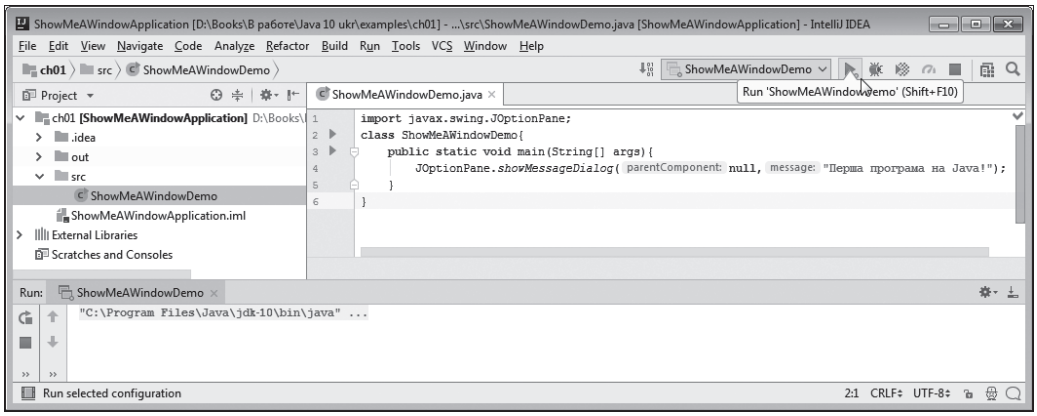

Рис. 1.1. Вікно середовища розробки IntelliJ IDEA з кодом програми

Нагадаємо алгоритм створення нової програми в середовищі IntelliJ IDEA. Для цього маємо скористатись командою ***Project*** з підменю ***New*** головного меню ***File*** (якщо у вікні середовища вже відкритий проект), або використовуємо команду ***Create New Project*** в стартовому вікні середовища. Як назву проекту вказуємо `ShowMeAWindowApplication`. У створений проект в папку ***src*** добавляємо файл з описом класу (команда ***Java Class*** з підменю ***New*** контекстного меню або головного меню ***File***). Вказуємо для файлу назву `ShowMeAWindowDemo`.

Для компіляції і запуску програми на виконання в середовищі IntelliJ IDEA вибираємо в меню **Run** команду **Run Project**. Запустити проект також можна за допомогою команди **Run** з контекстного меню файлу з головним класом. Така ж команда є в головному меню **Run**, а після першого запуску з'явиться кнопка з зеленою стрілкою на панелі інструментів середовища розробки (див. рис. 1.1). В результаті починає виконуватись програма, і на екрані з'являється діалогове вікно, як на рис. 1.2.

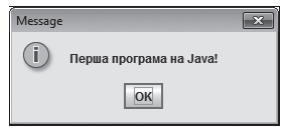

Рис. 1.2. Відображення діалогового вікна з повідомленням в результаті виконання програми

Вікно називається **Message** (назва вікна відображається зліва уверху в рядку заголовка), містить інформаційну піктограму (круг із буквою «і» всередині), текст `Перша программа на Java!`, під яким розміщена кнопка **ОК**. Після натискання кнопки **ОК** або системної піктограми (з хрестиком) у правому верхньому кутку вікна воно закриється, а програма припиняє виконання.

Аналіз програмного коду

Тепер проаналізуємо код програми. Найперша інструкція `import javax.swing.JOptionPane` необхідна для використання в програмі класу `JOptionPane` з бібліотеки Swing.

Бібліотека Swing містить набір класів для розробки прикладних програм із графічним інтерфейсом. Вона є невід'ємною частиною платформи Java. З бібліотекою Swing ми познайомимося детальніше дещо пізніше, коли будемо розглядати створення програм із графічним інтерфейсом.

Справа у тому, що діалогове вікно в нашій програмі відображається за допомогою методу `showMessageDialog()`. Це статичний метод класу `JOptionPane` — простіше кажучи, метод описаний у даному

класі, і говорити про метод `showMessageDialog()` поза контекстом класу `JOptionPane` просто немає сенсу.

У випадку виклику методу виконується певний блок програмного коду або набір команд. У даному випадку не важливо, навіть, які конкретно команди виконуються під час виклику методу `showMessageDialog()`, а важливий результат — створюється і відображається діалогове вікно. Тобто ми просто використовуємо вже існуючий метод, створений не нами.

Методи бувають статичними і звичайними, або не статичними. Якщо метод звичайний (не статичний), то він викликається з об'єкта. Такі методи нас поки що не цікавлять. Статичні методи викликаються з класу, в якому вони описані. Простіше кажучи, у випадку виклику статичного методу необхідно вказати клас, в якому описаний метод. Метод `showMessageDialog()` описаний у класі `JOptionPane`. Клас `JOptionPane`, у свою чергу, входить до складу бібліотеки Swing.

У загальному випадку інструкція `import` використовується для імпорту (тобто додавання) в програму всіляких утиліт — зазвичай класів. Причому після `import`-інструкції вказується не просто ім'я імпортованого класу, а повний «шлях» до нього, який відображає ієрархію пакетів, аж до того «місця», в якому зберігається клас.

В Java використовується система розподілу класів за пакетами. Ідея полягає в тому, що в межах пакета імена в усіх класів, що входять до пакета, є універсальними (вони різні), але якщо класи знаходяться в різних пакетах, то їхні імена можуть збігатися. Наслідки такі: кожен клас зберігається в якомусь пакеті. Причому пакет може сам входити в інший пакет, і так далі (пакет, який входить до іншого пакета, називається підпакетом). У результаті отримуємо ієрархію пакетів. Коли в `import`-інструкції вказується ім'я класу, що імпортується, то разом з іменем класу вказується і весь ієрархічний ланцюжок пакетів.

Зокрема, вираз `javax.swing.JOptionPane` означає, що клас `JOptionPane` входить до пакета `swing`, який є підпакетом пакета `javax`.

Увесь інший код програми — описання класу `ShowMeAWindowDemo`. Описання класу починається з ключового слова `class`, після якого вказується ім'я класу. Вираз `class ShowMeAWindowDemo`, наприклад, означає, що описується клас із назвою `ShowMeAWindowDemo`.

Назва класу збігається з іменем, яке ми вказали під час додавання файлу до проекту. Взагалі, в програмі може описуватись і використовуватись декілька класів, але серед них обов'язково (якщо мова не йде про аплети) є головний клас (у ньому описується головний метод програми). Ім'я головного класу збігається з іменем файлу, в якому він описаний. У даному випадку програма містить описання (явне) лише одного класу, він же є головним класом.

Власне описання класу міститься всередині фігурних дужок (між відкриваючою дужкою `{` і закриваючою дужкою `}`). У класі міститься тільки головний метод програми. Метод називається `main()`. Перед назвою методу вказані такі ключові слова:

- Інструкція `public` означає, що метод відкритий і доступ до нього існує й поза межами класу. Якщо врахувати, що виконання програми — це виконання методу `main()`, то доступність методу поза межами класу цілком логічна.
- Ключове слово `static` означає, що метод статичний і для його виклику немає необхідності створювати об'єкт (метод викликається з класу). Це теж цілком логічно, оскільки для створення об'єкта необхідно запустити програму на виконання, для чого слід викликати метод `main()`. Якби метод `main()` не був статичним, для його виклику довелося би створювати об'єкт, для чого необхідно запустити програму, і так далі — виходить замкнене коло.
- Ключове слово `void` означає, що метод `main()` не повертає результат. Тут теж усе розумно, оскільки результат просто немає куди повертати.

Методи можуть повертати результат. Якщо метод повертає деяке значення як результат, то інструкцію з викликом методу можна використовувати в якомусь виразі, ототожнюючи її з результатом, що повертається. Існують методи, які не повертають результат. Під час виклику таких методів просто виконується певна послідовність команд (яких саме — залежить від опису методу).

Після імені головного методу `main()` в круглих дужках указана інструкція `String[] args`. Даною інструкцією описується аргумент `args` методу `main()`, який являє собою текстовий масив. Аргумент методу `main()`

в програмному коді не використовуємо. А втім, за правилами мови Java, аргумент головного методу все одно має бути описаний.

У методу можуть бути аргументи — значення, які передаються методу під час виклику і від яких залежить результат виконання методу. Під час виклику методу аргументи через кому вказуються в круглих дужках після імені методу, в тому порядку, як вони оголошені в описі методу. Якщо у методу немає аргументів, то для виклику методу після його імені вказуються пусті круглі дужки.

Що стосується головного методу програми, то у нього є аргументи (точніше, один аргумент, але він є текстовим масивом — набором текстових значень). Справа в тому, що у випадку запуску програми на виконання їй можуть передаватися параметри. Саме такі параметри, котрі передаються в програму під час запуску, ототожнюються з текстовим масивом, який є аргументом методу `main()`. Аргументи головного методу використовуються не дуже часто. Ми також не плануємо їх використовувати. Однак, стандарт описання методу `main()` передбачає й описання аргументів методу.

Ключове слово `String` є назвою класу, через об'єкти якого в Java реалізуються текстові значення. Об'єкти ми обговоримо пізніше, а зараз ототожнюватимемо ідентифікатор `String` зі значенням текстового типу. Пусті квадратні дужки `[]` після ключового слова `String` свідчать про те, що мова йде про масив. Назва аргументу головного методу є довільною (але традиційно аргумент називають `args`).

Ще одна пара фігурних дужок (відкриваюча `{` і закриваюча `}`) визначає тіло головного методу програми. Там усього одна команда `JOptionPane.showMessageDialog(null,"Перша програма на Java!")`, якою власне і відображається діалогове вікно.

У кінці команди ставиться крапка з комою. Всі команди у мові Java закінчуються крапкою з комою.

Команда `JOptionPane.showMessageDialog(null,"Перша програма на Java!")` являє собою виклик методу `showMessageDialog()`. Метод викликається з класу `JOptionPane`. Першим аргументом методу передається ключове слово `null`, яке означає пусте посилання. В загальному випадку першим аргументом методу `showMessageDialog()` передається посилання на батьківське вікно (вікно верхнього рівня по відношенню

до вікна, яке відкривається). Але оскільки в даному випадку батьківського вікна просто немає, то використано стандартне ключове слово `null`, яке означає відсутність батьківського вікна.

Другим аргументом методу `showMessageDialog()` передається текст `"Перша програма на Java!"`. Саме він відображається в діалоговому вікні, яке з'являється на екрані під час виконання програми.

Текстове значення в Java береться в подвійні лапки. Вираз у подвійних лапках називається текстовим літералом.

Загальні зауваження

Таким чином, ми використали загальний підхід, який коротко зводиться до наступного:

- програма складається з опису класу;
- клас, у свою чергу, містить описання головного методу програми;
- при виконанні головного методу програми викликається статичний метод `showMessageDialog()` з класу `JOptionPane`;
- для імпорту в програму класу `JOptionPane` використовуємо `import`-інструкцію.

Розглянутий нами приклад дає певне формальне правило для написання програм в Java. Зокрема, базовим є наступний шаблон коду (ключові елементи шаблону виділені жирним шрифтом):

```
class ім'я_класу{
   public static void main(String[] args){
      // Програмний код
   }
}
```

Для створення програми необхідно, як мінімум, описати клас (головний клас програми, його назва така сама, як назва файлу з кодом, що додається в проект). У класі описується метод `main()`, перед його іменем указуються атрибути `public`, `static` і `void`, а в круглих дужках після

імені методу вказується вираз вигляду `String[] args`. Блоки програмного коду (тіло класу і тіло методу) виділяються фігурними дужками. Це, так би мовити, обов'язкова програма. Якщо в програмі необхідно використовувати бібліотечні класи, на кшталт класу `JOptionPane`, на початку програми використовуємо `import`-інструкцію.

У програмних кодах використовуються ***коментарі***. Коментарі ігноруються під час компіляції і призначені для пояснення програмного коду. Існує три типи коментарів.

Однорядкові коментарі починаються з подвійної нахиленої риски `//`. Усе, що знаходиться справа від подвійної нахиленої риски `//`, є коментарем. Однорядковий коментар розміщується в одному рядку.

Багаторядковий коментар починається з інструкції `/*` і закінчується інструкцією `*/`. Усе, що знаходиться між інструкціями `/*` і `*/`, є коментарем. Такий коментар може розміщуватись у декількох рядках.

Крім однорядкових і багаторядкових коментарів, є ще й коментарі документування. Вони починаються з інструкції `/**` і закінчуються інструкцією `*/`. Такі коментарі використовуються для автоматичного генерування вмісту в довідникових HTML-документах.