Тема 1. Огляд технологій JAVA і IDE

Лекція 1. Технологія Java

Java (вимовляється «Джава») – одна із найбільш затребуваних на сьогоднішній день мов програмування, і одночасно – це стек технологій, які покривають весь процес створення сучасних додатків будь-якого спрямування та рівня складності.

Мова Java була розроблена компанією Sun Microsystems (перша офіційна версія Java 1.0 була випущена 26 серпня 1996 року). З 2010 року Java розробляється і підтримується компанією Oracle. Автором мови вважається співробітник компанії Sun Microsystems Джеймс Гослінг (James Gosling), який розробив її основи, створюючи мову для програмування побутових електронних пристроїв. Ця мова спочатку називалась Oak («Дуб»), але згодом стала універсальною мовою програмування та була перейменована в Java. Вважається, що сьогоднішня назва мови і технології Java походить від назви сорту кави.

Межі технології Java досить розмиті і весь час розширюються. Спочатку Java-технологія призначалася для програмування побутової електроніки і телефонів. Потім Java стала виконуватися в інтернет-браузерах – з'явилися *аплети*. Потім виявилося, що на Java можна створювати повноцінні програми. Їх графічні елементи стали оформляти у вигляді компонентів *JavaBeans*, з якими Java увійшла в світ розподілених систем і проміжного програмного забезпечення.

Наступним кроком стало програмування серверів – з'явилися *сервлети* і розподілені компоненти *Enterprise Java Beans (EJB)*. Сервери повинні взаємодіяти з базами даних – тож з'явилась технологія *JDBC* (*Java DataBase Connectivity*). Перераховані технології виявилися вдалими і багато систем управління базами даних і навіть операційні системи включили Java в своє ядро, наприклад Oracle, Linux, MacOS X, AIX.

Таке швидке і широке поширення технології Java не в останню чергу пов'язано з тим, що вона використовує нову, спеціально створену мову програмування, яка так і називається – мова Java. Ця мова створена на базі мов Smalltalk, Pascal, C++ та ін., увібравши їх кращі, на думку творців, риси і відкинувши гірші. На цей рахунок є різні думки, але безперечно, що мова вийшла зручною для вивчення, написані на ній програми легко читати і відлагоджувати.

Чому саме Java?

Java працює на 3 мільярдах пристроїв по всьому світу – це ПК, мобільна електроніка, пристрої що носяться, бортові системи, мережеве обладнання, промислове програмоване обладнання.

На сьогоднішній день Java лідирує серед платформ і мов програмування (на даний момент щільно конкуруючи з PHP, Python і C/C++). Вона займає першу позицію як серед пропозицій на ринку праці IT-індустрії, так і в рейтингу вподобань розробників (рис. 1.1, 1.2).

Рейтинг мови програмування не дає відповіді на запитання, яка мова краща. Цей показник визначає популярність мови програмування. Взагалі існує кілька рейтингів, що оцінюють популярність мов програмування. Серед них – TIOBE Software, Red Monk, PYRL, IEEE Spectrum. Які висновки можна зробити, аналізуючи ці рейтинги? Незважаючи на деякі відмінності в розподілі місць мов, лідерами є Java, JavaScript, Python, C. Саме ці чотири мови займають близько 60% всього обсягу використання мов програмування в світі. Тому, зупинивши вибір на одному з них, ви будетев програмістом високого світового рівня. Взагалі класний програміст повинен досконало володіти двома мовами програмування. Щоб вибрати дві мови із приведеного квартету, необхідно детально познайомитись із особливостями кожної із чотирьох мов програмування з урахуванням специфіки ваших уподобань.

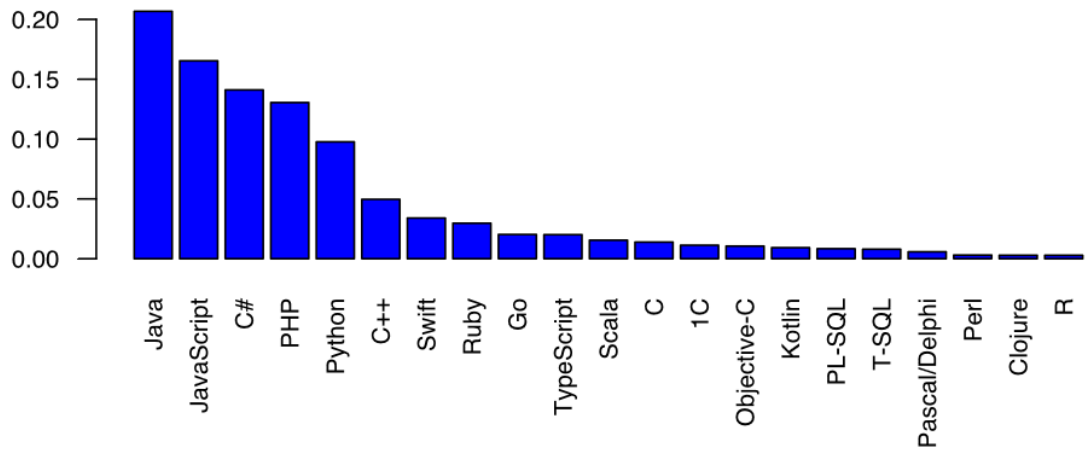

Рис. 1.1. Рейтинг мов програмування компанії dou.ua за частотою комерційного використання на початок 2018 р. В опитуванні взяло участь 7361 розробників, 90% з яких проживають в Україні. Джерело: https://dou.ua/lenta/articles/language-rating-jan-2018/

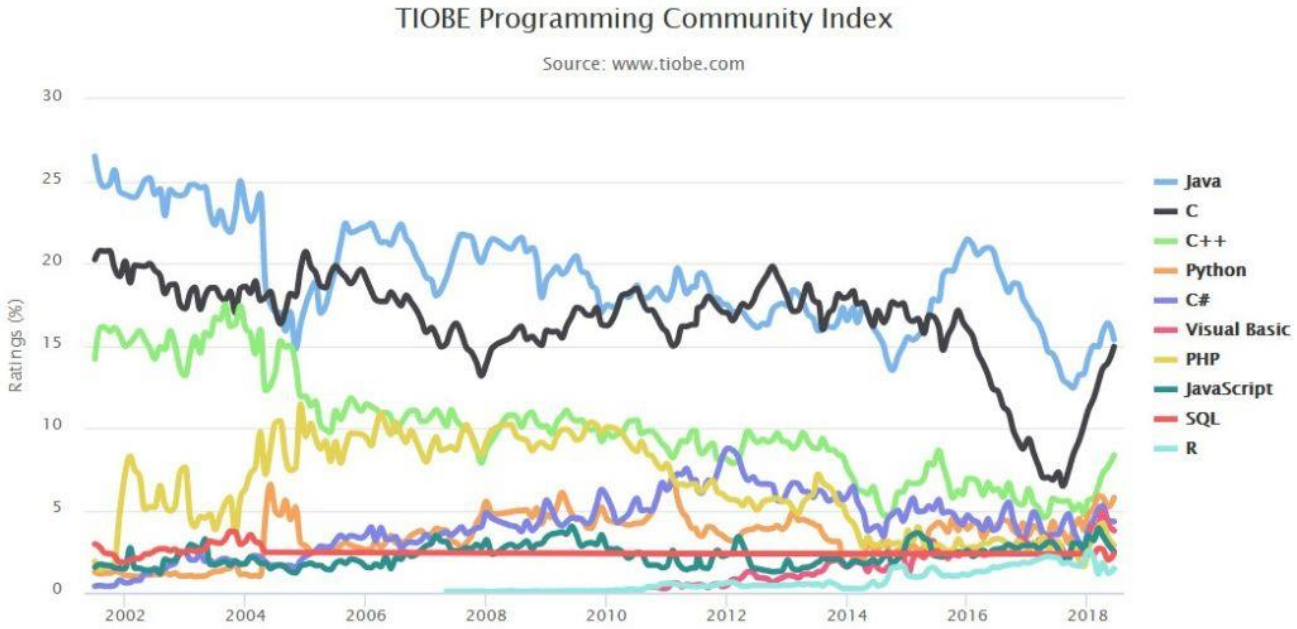

Рис. 1.2. Рейтинг популярності мов програмування в 2002-2018 рр., побудований компанією TIOBE Software шляхом аналізу трендів 25 основних пошукових систем, включаючи Google,Ebay,YouTube, Yahoo !, Wikipedia, Amazon, Bing та ін. Джерело: www.tiobe.com.

На основі Java технологій (точніше, в основному на основі J2EE-технологій) реалізовані такі проекти світового рівня, як Amazon, eBay, Twitter, Yandex, LinkedIn, Yahoo!, OpenOffice, RuneScape. Продукти компанії Google також багато в чому спираються на технології Java.

На Java-технологіях в основному спеціалізуються такі провідні компанії як Oracle, SAP, IBM. Зокрема, СУБД Oracle включає Java-машину як свою складову частину, що забезпечує можливість безпосереднього програмування СУБД на мові Java, в тому числі написання збережених процедур на Java.

Java підтримує всі новітні технології (в тому числі web- та android-розробки) і містить засоби, що значно спрощують і прискорюють процес розробки. Крім того, мова активно розвивається, а проблеми продуктивності зведені до мінімуму.

Особливості технології Java

Відомо, що програма, написана на будь-якій мові високого рівня, не може бути відразу ж виконана. Її спочатку треба *компілювати*, тобто перевести в послідовність машинних команд

процесора – *об'єктний модуль*. Але і він, як правило, не може бути відразу ж виконаний: об'єктний модуль треба ще скомпонувати з бібліотеками використаних в модулі функцій і розв'язати перехресні посилання між секціями об'єктного модуля, отримавши в результаті *завантажувальний модуль* – повністю готову до виконання програму (рис. 1.3).

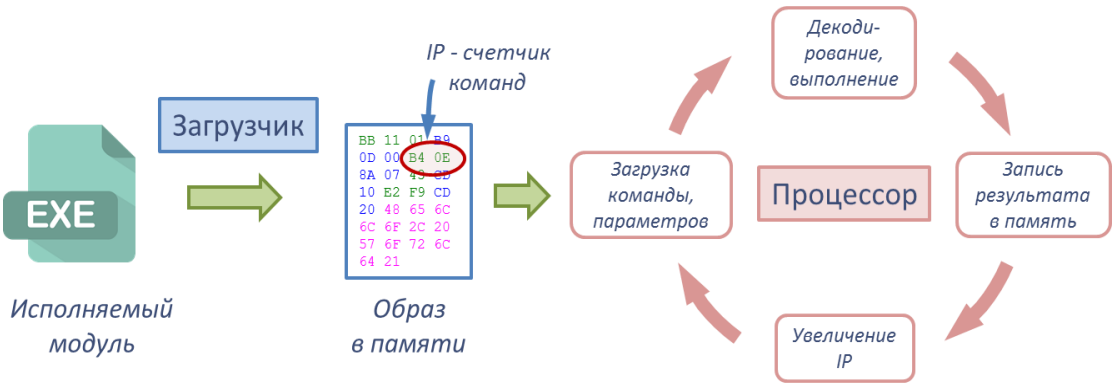

Рис. 1.3. Процес виконання звичайної програми.

Java-програма не може уникнути цих процедур, але тут проявляється головна особливість технології Java – програма компілюється не в команди якогось конкретного процесора, а в команди так званої ***віртуальної машини Java*** (JVM, Java Virtual Machine). Віртуальна машина Java – це сукупність команд разом з системою їх виконання. Віртуальна машина Java повністю стекова, використовує плоску адресацію пам'яті і невелику кількість регістрів. Тому команди JVM короткі, більшість з них має довжину 1 байт, а середня довжина команди складає 1,8 байти: ось чому команди JVM називають ***байт-кодами*** (bytecodes).

Повний опис команд і всієї архітектури JVM міститься в специфікації віртуальної машини Java (VMS, Virtual Machine Specification). Ця специфікація гарантує однакове виконання java-коду на будь-якій апаратній платформі. Так реалізується принцип Java "*Write once, run anywhere*"– "Написано один раз, виконується де завгодно".

Таким чином, програми на Java транслюються в ***байт-код***, що виконується ***віртуальною машиною Java*** (JVM) – програмою-інтерпретатором, яка обробляє команди байт-коду і перетворює їх в інструкції конкретного процесора або виклики функцій конкретної операційної системи (рис. 1.4). Такі віртуальні Java-машини розроблені для всіх популярних операційних систем, а також для багатьох маловідомих.

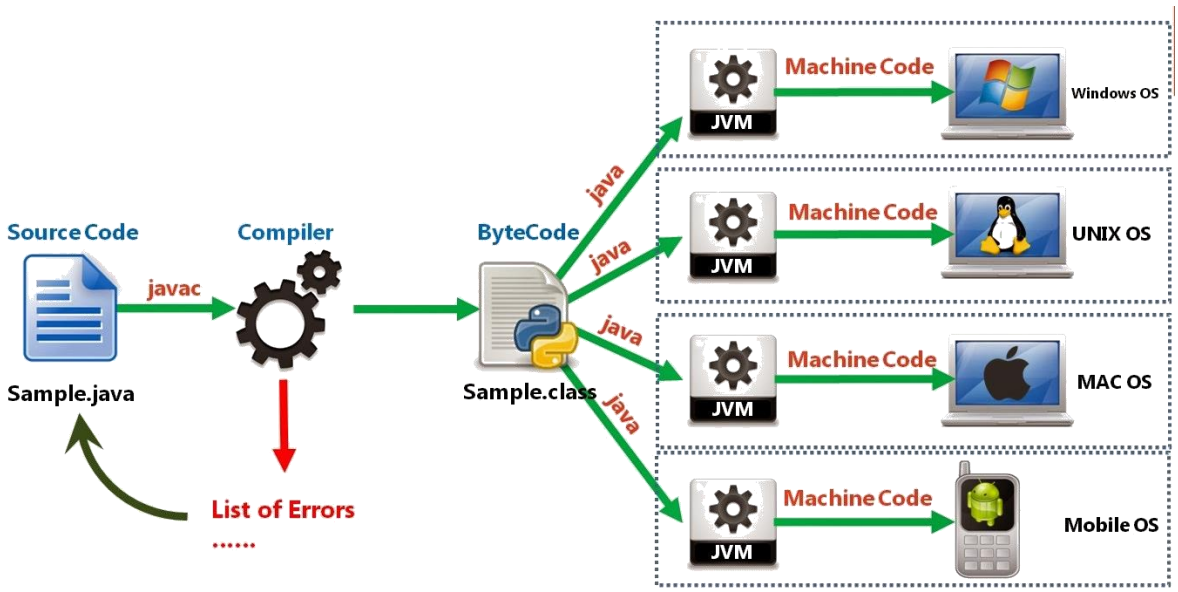

Рис. 1.4. Виконання Java-програми

Мова Java володіє всіма ***перевагами*** сучасної мови програмування:
Java-код володіє всіма перевагами сучасної мови програмування:

- *Простий* – завдяки продуманому, простому і лаконічному синтаксису: почати розробку на Java легко, навіть для програмістів-початківців (але комерційне використання Java вимагає багато знань і вміння працювати з великою кількістю технологій). Написання програм також спрощується за рахунок автоматичного управління пам'яттю і автоматичного збирання сміття.
- *Інтерпретований* – код виконується на віртуальній машині (JVM), що великою мірою визначає і наступні переваги.
- *Переносний* – повна незалежність байт-коду від операційної системи і устаткування, що дозволяє виконувати Java-додатки на будь-якому пристрої, для якого існує JVM
- *Надійний* – виключені помилки, пов'язані з управлінням пам'яттю; проста і гнучка обробка виняткових ситуацій.
- *Безпечний* – виконання програми повністю контролюється JVM. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером) викликають негайне переривання.
- *Об'єктно-орієнтований* – тобто застосовує всі переваги ООП та підтримує об’єктно-орієнтований цикл розробки.
- *Високопродуктивний* (питання швидкості роботи Java-програм детально розглядається далі).
- *Багатопотоковий, розподілений* – тобто ядро мови має вбудовані механізми паралельного виконання та синхронізації потоків.
- *Активно розвивається* – постійно додаються нові можливості, в тому числі на рівні стандарту мови. Наприклад, *анотації* були додані в стандарт синтаксису java 1.5, і з тих пір дуже активно використовуються і значно спрощують розробку. Також можливості мови Java в свій час розширювалися за рахунок додавання параметричної типізації, лямбда-виразів і багато чого іншого.

Недоліки концепції віртуальної машини:

За рахунок виконання байт-коду віртуальною машиною знижується продуктивність програм і алгоритмів. За даними сайту shootout.alioth.debian.org, для семи різних задач час виконання алгоритму на Java в середньому в півтора-два рази більший, ніж час виконання того ж алгоритму на C/C ++. В деяких випадках Java-код виконується швидше, а в окремих випадках – до 7 разів повільніше. З іншого боку, в більшості з тестів споживання пам'яті Java-машиною було в 10-30 разів більше, ніж програмою на C/C ++. Також примітним є дослідження, проведене компанією Google, згідно з яким відзначається істотно нижча продуктивність і більше споживання пам'яті в тестових прикладах на Java в порівнянні з аналогічними програмами на C++.

За останні роки проведена велика робота, щоб звести до мінімуму проблеми продуктивності Java. Зокрема, були зроблені такі удосконалення:

- Застосування *JIT-технології* – тобто трансляції байт-коду в машинний код безпосередньо під час роботи програми з можливістю збереження версій класу в машинному коді. Наприклад, в циклах JVM інтерпретує повторювані ділянки коду тільки один раз, а при повторному зверненні до них просто виконує готові машинні команди.
- У стандартних бібліотеках Java широко використовується платформо-залежний код (*native-код*). Це означає, що стандартні функції Java реалізовані найбільш оптимальним чином і максимально використовують можливості API кожної конкретної ОС.
- *Оптимізація* віртуальної машини Java, яка полягає у вдосконаленні роботи самого інтерпретатора JVM з метою зменшити використання пам’яті та підвищити швидкість перетворення байт-коду на машинні команди.

- У стандартних бібліотеках Java широко використовується переносний залежний код (*native-код*). Це означає, що стандартні функції Java реалізовані найбільш оптимальним чином і максимально використовують можливості API кожної конкретної ОС.
- Розробляються *апаратні засоби*, що забезпечують прискорену обробку байт-кодів. Наприклад, фірма SUN Microsystems випустила мікропроцесори PicoJava, що працюють на системі команд JVM і безпосередньо виконують байт-коди. Компанія Oracle пропонує потужні сервери, засновані на java-процесорах. Є і Java-процесори інших фірм. Сюди ж відноситься технологія Jazelle, яка прискорює виконання байт-кодів і підтримується деякими процесорами фірми ARM.

В результаті цих новацій вже станом на 2012 рік, код Java 1.7 в тестах показав продуктивність, порівнянну з продуктивністю коду, написаного на мові С. У середньому швидкість виконання відрізнялася лише в 1.8 рази. Бенчмаркінг поточної версії openjdk 11 2018-09-25 (2018 рік), проведений ресурсом debian.net, як і раніше показує різницю в швидкості в 1.5-3 рази в порівнянні з С.

(Джерело: https://benchmarksgame-team.pages.debian.net/benchmarksgame/faster/java-gpp.html).

Ідеї, закладені в концепцію віртуальної машини Java, надихнули безліч ентузіастів на розширення переліку мов, що виконуються на віртуальній машині. Ці ідеї знайшли також вираз в специфікації загальномовної інфраструктури CLI (*Common Language Infrastructure*), закладеної в основу платформи *.NET* компанією Microsoft.

Складові технології JAVA

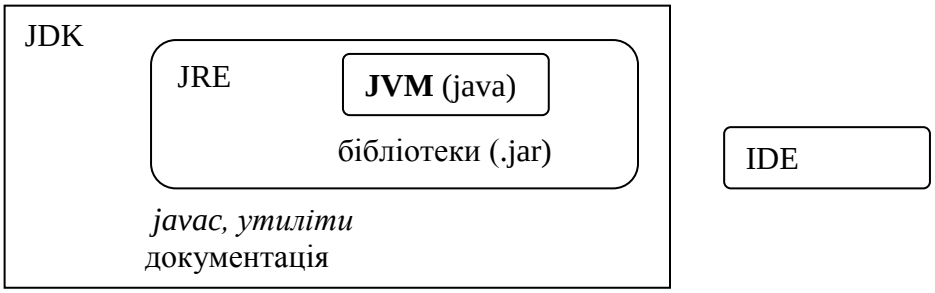

Рис. 1.5. Складові технології JAVA

Java Virtual Machine (JVM) – віртуальна машина Java є основою всієї технології (рис. 1.5). Вона безпосередньо відповідає за виконання java-програм, організовує для них роботу з системними ресурсами (пам'яттю, файловою системою) і системні виклики (тобто організує необхідні виклики функцій конкретної ОС).

Java Runtime Environment (JRE) – це мінімальна реалізація віртуальної машини, необхідна для виконання готового байт-коду. Складається з віртуальної машини – JVM і бібліотеки стандартних Java-класів, але не включає компілятор та інші засоби розробки.

Java Development Kit (JDK) – комплект розробника додатків на мові Java, що включає в себе все, що необхідне для розробки і зборки додатків, в тому числі:

- JRE, яка включає JVM (програма java.exe), стандартні бібліотеки у вигляді файлів .jar та ін.;
- Компілятор Java (програма javac.exe);
- Приклади, документацію;
- Програми-утиліти, що виконують різні допоміжні функції:
  - jar – програма архівації (стиснення) скомпільованих модулів. В результаті стиснення виходить Java-архів (з розширенням .jar), який можна розміщувати на сервері, або запускати на виконання на десктоп-комп'ютері;
  - javadoc – генерує документацію в HTML-форматі з вихідного коду класів, використовуючи для цього спеціальні коментарі в коді;
  - javah – генерує заголовкові та вихідні файли на мові С, за допомогою яких можна використовувати Java-класи в програмах на С;

- javakey – програма для додавання електронного підпису до скомпільованих модулів;
- javap – дизасемблер, що дозволяє відновити вихідний код класу за наявним скомпільованим файлом байт-коду (.class).

До складу JDK *не* входить середовище розробки (*IDE, Integrated Development Environment*), тому розробник, що використовує тільки JDK, змушений використовувати зовнішній текстовий редактор і компілювати свої програми з командного рядка за допомогою компілятора javac та інших утиліт.

Oracle і сторонні розробники також надають додаткові *SDK* (Software Development Kit – набір інструментів розробника) – набори додаткових бібліотек і утиліт, необхідних для створення певного виду додатків або компонентів. Наприклад, Java EE SDK дозволяє розробляти додатки Java Enterprise, про які йтиметься далі, Android SDK – відповідно містить все необхідне для розробки android-додатків.

Для компіляції і запуску звичайних desktop-додатків необхідним є тільки *JDK*, який можна завантажити на сайті компанії Oracle:
http://www.oracle.com/technetwork/java/javase/overview/index.html.

Існує велика кількість підручників, посібників і довідників з мови і технологій Java, наприклад: Герберт Шилдт «Java. Повне керівництво»; Кей С. Хорстманн, Гарі Корнелл «Java. Бібліотека професіонала ».

Повний та постійно оновлюваний довідник Java™ Platform, Standard Edition 10 API Specification опублікований на сайті компанії Oracle:
http://docs.oracle.com/javase/10/docs/api/index.html.

Крім того, компанія Oracle створює і підтримує керівництво для вивчення платформи java (https://docs.oracle.com/javase/tutorial/) – він містить найбільш якісний опис синтаксису мови, а також багатьох технологій, які супроводжують Java.

Приклад роботи з JDK (для Windows) і найпростіша програма на Java

Всі функції JDK орієнтовані на виклик з командного рядка. Незважаючи на це JDK повністю підтримує створення програм, що працюють в графічних середовищах, таких як MS Windows або X Window System.

Написати java-програму можна в будь-якому текстовому редакторі, наприклад, Notepad, WordPad в MS Windows, або в редакторах vi, emacs в Linux. Щоб створити java-додаток за допомогою JDK, потрібно виконати наступні кроки:

1. Підготовка Java SDK
   1.1. Завантажити останню версію SDK с сайту Oracle, або завантажити потрібну версію JDK. Після виконання установки в директорії SDK знаходяться наступні піддиректорії:
      - bin – програми, що виконують різні функції JDK (в т.ч. компілятор);
      - demo – приклади написаних на Java програм;
      - docs – документація стосовно мови та платформи Java;
      - include – заголовки native-методів;
      - jre – файли JRE;
      - lib – стандартні бібліотеки класів;
      - src.zip – вихідні коди бібліотек.

   1.2. Додати шлях до папки bin пакета SDK або JDK в системну змінну PATH. Це можна зробити в командному вікні (Win + R), ввівши системну команду із зазначенням шляху до папки bin, наприклад:
   ```
   path C:\Program Files\Java\jdk1.7.0\bin.
   ```
   У Windows 7/10 щоб змінити значення системної змінної, потрібно зайти в Панель управління – система – додаткові параметри системи – додатково – змінні середовища.
   ВАЖЛИВО: шлях до JDK повинен додаватися до наявного значенням PATH через «;», але не замінювати його.

2. Створення класу додатка.

Будь-яка java-програма складається з *класів*. Найпростіша програма має всього один клас – клас додатка. Для того щоб цей клас можна було запустити на виконання, він повинен мати спеціальний метод з ім'ям *main*(). Фактично в додатку може бути кілька класів з таким методом, тоді налаштуваннями проекту визначається, який саме з них буде запускатися при старті програми. При завершенні методу main() основний потік додатку завершується – якщо не було створено інших потоків, то це означає закриття програми.

Створимо файл вихідного коду нашого класу з розширенням .java (наприклад, HelloWorld.java) в будь-якій директорії за допомогою будь-якого текстового редактора. Назва файлу *обов'язково* має збігатися з назвою класу нашого додатку:

```
public class HelloWorld {                          // Основний клас програми
      public static void main (String [] args) {   // Головний метод
                                                    // (точка входу в програму)
                                                    // args - Параметри виклику
          System.out.println ("Hello World!");      // Виведення в консоль
      }
}
```

Метод main() має бути оголошений відкритим (*public*), щоб JVM могла його викликати. Також він повинен бути *статичним* (тобто таким, що відноситься до класу, а не до об'єкта), щоб для його виклику не потрібно було створювати об'єкт класу програми. Параметри цього методу завжди однакові і дозволяють отримати доступ до параметрів командного рядка, який використовувався для запуску нашої програми.

Всередині методу main() наразі знаходиться тільки один оператор, що використовує об'єкт System.out – стандартний потік виведення, пов'язаний з консоллю, – і викликає його метод println(), який відповідно виконує виведення на консоль.

3. Компіляція програми з командного рядка (рис. 1.6).

Викликаємо компілятор із зазначенням нашого .java-файлу.

```
C:\path\>javacHelloWorld.java
```

Результатом компіляції є файл HelloWorld.class – байт-код нашого класу (рис. 6).

4. Запуск програми. Викликаємо JVM із зазначенням імені класу-додатка (але не імені .class-файлу).

```
C:\path\>java HelloWorld
```

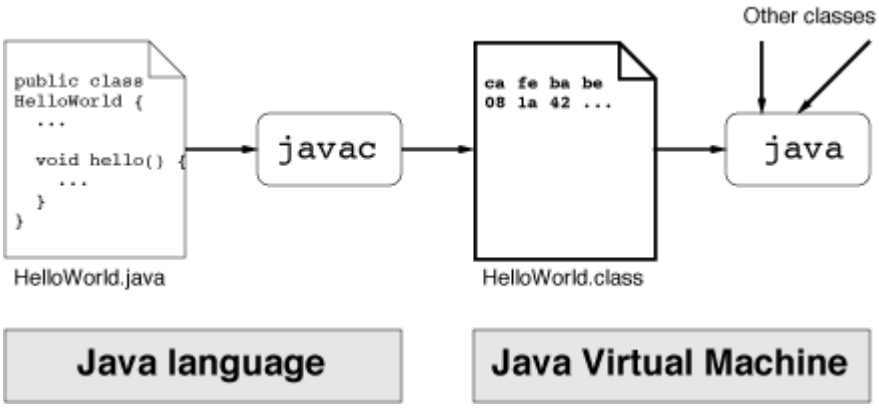

Рис. 1.6. Компіляція і виконання програми

Розглянемо тепер приклад програми, яка не просто виводить текст, але також вводить вихідні дані і проводить над ними деякі дії. Нехай необхідно порахувати суму всіх натуральних чисел від *a* до *b*. На Java це буде виглядати так:

```
import java.util.Scanner;                 // Підключаємо  клас із бібліотеки

public class Sum {                      // Основний клас програми

     public static void main(String[] args) {        // Головний метод

        Scanner scanner = new Scanner(System.in);  // Створюємо об’єкт для читання
        System.out.println("Enter a and b");       // Виводимо текст
        int a = scanner.nextInt();      // Читаємо два цілих числа
        int b = scanner.nextInt();
        int res = 0;                    // Змінна для збереження результату
        for (int i = a; i<b; i++) {     // Зчитуємо суму
            res += i;
        }
        System.out.println("The sum is "+res);     }
}
```

Тут першим рядком підключається стандартний клас *Scanner* із пакета *java.util*, який дозволяє дуже легко читати дані, введені користувачем.

Всередині методу main() створюється об'єкт цього класу – scanner (в іменах великі і маленькі букви розрізняються). Для читання цілого числа використовується метод scanner.nextInt(). Нарешті, підрахунок суми виконується циклом for, який в даному випадку абсолютно аналогічний циклу for мов C++ та C#.

Результат роботи компільованої програми буде таким:

```
Enter a and b
3   14
The sum is 88
```

Якщо в коді програми допущені помилки, то компілятор повідомить про це, із зазначенням опису помилки і рядка програми. Файл .class при цьому не створюється. Наприклад, якщо в попередній програмі написати:

```
int a = Scanner.nextInt();
```

то компілятор сповістить про помилку і опише її наступним чином:

```
Sum.java: 11: non-static method nextInt () can not be referenced from a
static context
```

Тобто в 11 рядку є помилка, яка полягає в тому, що замість імені об'єкта використано ім'я класу, а оскільки метод nextInt() не є статичним (в чому можна переконатися переглянувши документацію з цього методу), то він повинен обов'язково бути викликаний від об'єкта.

Лекція 2. Платформи JAVA і Java IDE

Платформи Java

У Java існує кілька основних сімейств технологій, які зібрані в різні «видання» (editions) і встановлюються окремо:

Java SE (Standard Edition) – основне видання Java, містить JDK для створення звичайних desktop-додатків. Саме з нього починається вивчення Java.

Java EE (Enterprise Edition) – набір технологій і специфікацій для створення програмного забезпечення рівня підприємства. Це видання призначене для створення серверних і розподілених

компонентів. Воно включає ряд серверних і web-технологій, таких як Servlet, JSP/JSTL, EJB, JAX-WS/JAX-RS, JTA (Transaction API), JPA (Persistance API). Додатки, що створюються за технологією Java EE, зазвичай також використовують сторонні фреймворки (які не є частиною Java EE), наприклад:

- Spring – набір різноманітних засобів, що спрощують розробку додатків корпоративного масштабу;
- Hibernate – фреймворк для доступу до даних із зовнішніх джерел, в першу чергу, із БД. Реалізує об'єктно-реляційне відображення (object-relational mapping, ORM), тобто відображення об'єктно-орієнтованої моделі даних в традиційні реляційні бази даних.

Java ME (Micro Edition) – створена для використання в пристроях, обмежених по обчислювальній потужності, наприклад в мобільних телефонах, КПК, вбудованих системах;

JavaFX – технологія створення додатків з розвиненим графічним інтерфейсом користувача, в тому числі RIA (Rich Internet Applications) – корпоративних і бізнес-web-додатків.

Java Card – технологія розробки додатків, що працюють на смарт-картах та інших пристроях з дуже обмеженим обсягом пам'яті і слабкими можливостями обробки.

Java и Android

Мова Java, починаючи з JDK 1.5, активно використовується для створення мобільних додатків для операційної систему Android. ОС Android поставляється із вбудованою віртуальною машиною java (JVM): ранні версії Android включали в себе JVM Dalvic, яка наразі (починаючи з Android 5.0 "Lollipop") замінена на більш сучасну ART (Android RunTime). При цьому java-програми компілюються в байт-код спеціального виду, який безпосередньо виконується такою вбудованою JVM.

Для компіляції таких додатків використовується додатковий інструмент – Android SDK, який розробляється і підтримується компанією Google. Розробку Android-додатків можна вести в різних IDE, в тому числі в IntelliJ IDEA і Eclipse (в Eclipse необхідно використовувати додатковий плагін – Android Development Tools, ADT). Також існує спеціальна збірка IDE на основі IntelliJ IDEA, яка спеціально призначена для Android-розробки – Android Studio.

IntelliJ IDEA

Із сучасних Java IDE найбільш популярні IntelliJ IDEA, Eclipse та NetBeans. Кожна з них має свої особливості і своїх шанувальників. Всі ці IDE опираються на сервіси (утиліти), що надаються JDK. У більшості випадків для компіляції програм середовище розробки використовує компілятор зі складу JDK (програму javac). Тому середовища розробки або включають в свій комплект поставки одну з версій JDK або вимагають для своєї роботи попередньої інсталяції JDK на машині розробника (в останньому випадку в налаштуваннях IDE потрібно вказати шлях до встановленого JDK).

В даному курсі в якості основної IDE розглядається IntelliJ IDEA. Це комерційна IDE для розробки на Java, яка за правом вважається найбільш зручним середовищем для Java-розробки. Весь її дизайн орієнтований на продуктивність роботи програмістів, дозволяючи їм сконцентруватися на розробці функціональності, в той час як середовище бере на себе виконання рутинних операцій. Так, IDEA має великі можливості генерації заготовок коду, конфігураційних файлів. Середовище надає потужні можливості автоматичного доповнення коду. Система підказок не тільки пропонує найбільш прийнятні об'єкти або методи, але також дозволяє автоматично виправляти багато помилок, наприклад, пропущене підключення класу або не вказану залежність модуля.

IntelliJ IDEA надає вбудований інструментарій для розробки графічного інтерфейсу – графічний редактор форм.

Серед інших можливостей, IntelliJ IDEA добре сумісна з багатьма популярними вільними інструментами розробників, такими як системи контролю версій (CVS, GIT, Subversion), системами зборки проектів (Ant, Maven), і фреймворками тестування (JUnit).

IntelliJ IDEA доступна в двох версіях: Community Edition і Ultimate Edition, порівняння яких наведене в табл. 1.1:

Таблиця 1.1. Порівняння версій IntelliJ IDEA.

Версія	Community Edition	Ultimate Edition
Ліцензія	Разповсюджується вільно під відкритою ліцензією Apache 2.0	Комерційна (платна)
Підтримує технології	Java SE, системи контролю версій, android- розробка, системи зборки Ant, Maven	Те ж + Java EE, + UML-діаграми, CASE-засоби, + інтегровані сервери додатків (Tomcat, GlassFish та ін.), + підтримка відлагодження веб-додатків, + підтримка фреймворків (Spring, Hibernate, …).

Таким чином, вільна версія Community Edition підходить для початку навчання Java і створення desktop-додатків, в той час як для комерційної розробки і створення веб- та корпоративних додатків необхідна версія Ultimate Edition. Всі практичні завдання даного курсу можна виконати в IntelliJ IDEA Community Edition. Далі ми розглянемо основні прийоми роботи в IDEA.

Створення проекту

У стартовому вікні «Welcome to IntelliJ IDEA» тиснемо «Create New Project». Або в меню вибираємо File | New project.

У вікні «New Project» (рис. 1.7):

1) Вказуємо ім'я проекту;
2) Додаємо Project SDK, тобто шлях до встановленої версії JDK:
   – тиснемо кнопку New;
   – із списку, що розкрився, вибираємо «JDK»;
   – вказуємо шлях до JDK, тобто стандартний шлях в Windows ОС, наприклад:
   ```
   C:/Program Files/Java/jdk1.9.0_51.jdk
   ```
3) Тиснемо Next внизу вікна, а потім на наступному екрані натискаємо кнопку Finish.

Рис. 1.7. Створення проекту

Створення класу

1) Створюємо новий клас в папці src на вкладці Project за допомогою правої кнопки миші (рис. 1.8). Керувати відображенням/приховуванням вкладок можна за допомогою меню View | Tool Windows або за допомогою гарячих клавіш.

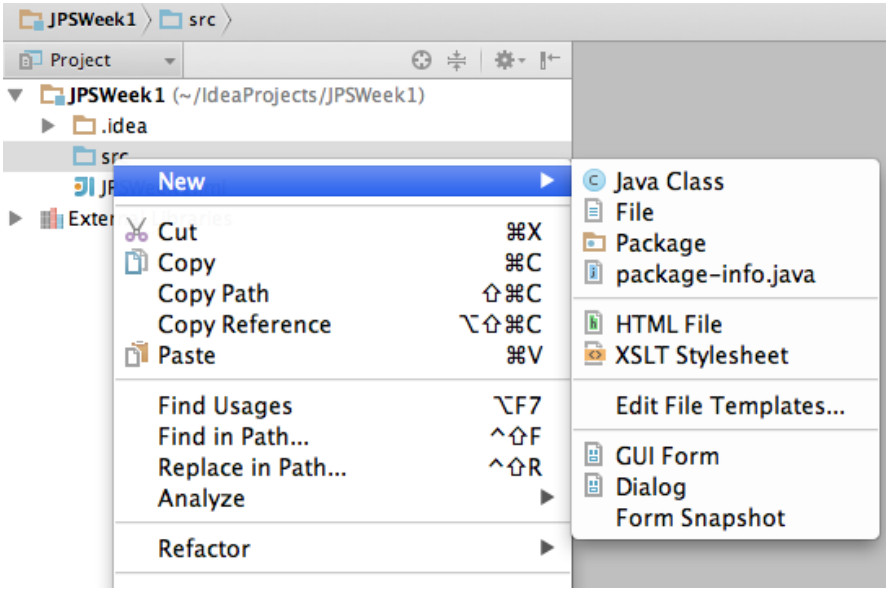

Рис. 1.8. Створення класу. Крок 1

2) Вказуємо ім'я класу HelloWorld, а в полі Kind залишаємо "Class" (рис. 1.9). Тиснемо ОК.

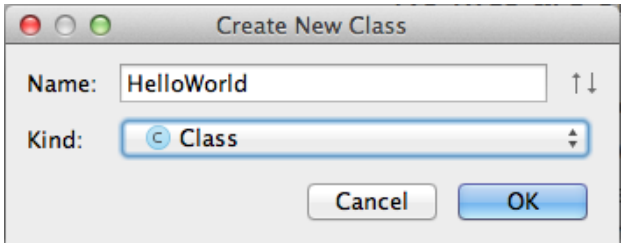

Рис. 1.9. Створення класу. Крок 2

Налаштування проекту

Налаштування проекту встановлюються у вікні Project Structure. Відобразити його можна кількома способами:

– вибравши пункт меню File | Project Structure;
– виділивши модуль в панелі Project і натиснувши F4;
– за допомогою гарячої клавіші Ctrl-Alt-Shift-S.

На вкладці Project (рис. 1.10) вказується ім'я проекту і задається версія JDK, наприклад "1.9", в полі Project SDK. Якщо JDK не задано, середовище не може впізнати стандартні класи і виділяє їх червоним кольором. Проект в цьому випадку не компілюється.

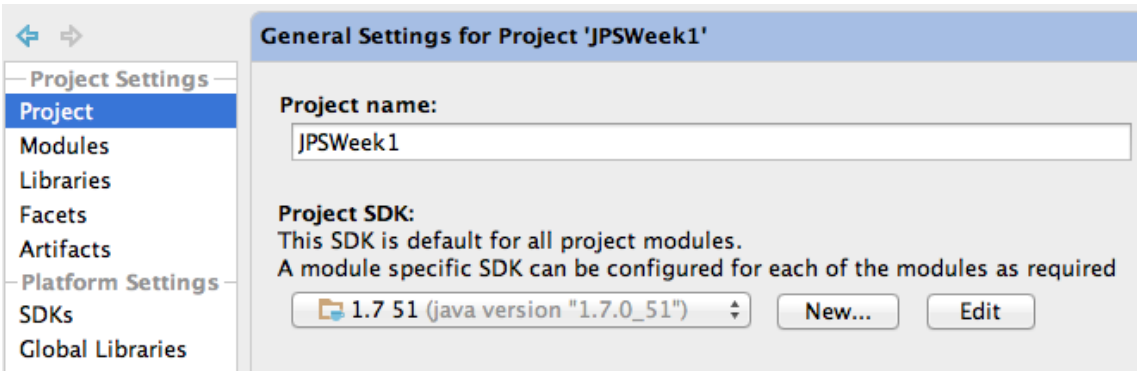

Рис. 1.10. Налаштування проекту: вкладка Project.

Якщо в списку Project SDK немає потрібної версії JDK, але вона встановлена на комп'ютері, її можна додати, натиснувши кнопку new і вказавши директорію установки JDK.

На вкладці Modules можна налаштувати JDK окремо для кожного модуля, що входить до складу проекту, а також задати інші налаштування модулів, наприклад, їх залежності.

Конфігурація запуску

Конфігурація запуску – це набір налаштувань модуля, який визначає, як він буде збиратися та запускатися в IDE. Для одного модуля можна створити кілька різних конфігурацій, наприклад, для запуску програми з різними параметрами.

Щоб налаштувати конфігурації запуску, потрібно в меню Run вибрати Edit Configurations. Конфігурації запуску звичайного desktop-додатка знаходяться в секції Application. Для кожної конфігурації задається ім'я (1), клас, що містить головний метод main() (2), домашня директорія для запущеного додатку (3), а також, якщо необхідно, опції запуску JVM, параметри командного рядка, змінні оточення (рис. 1.11).

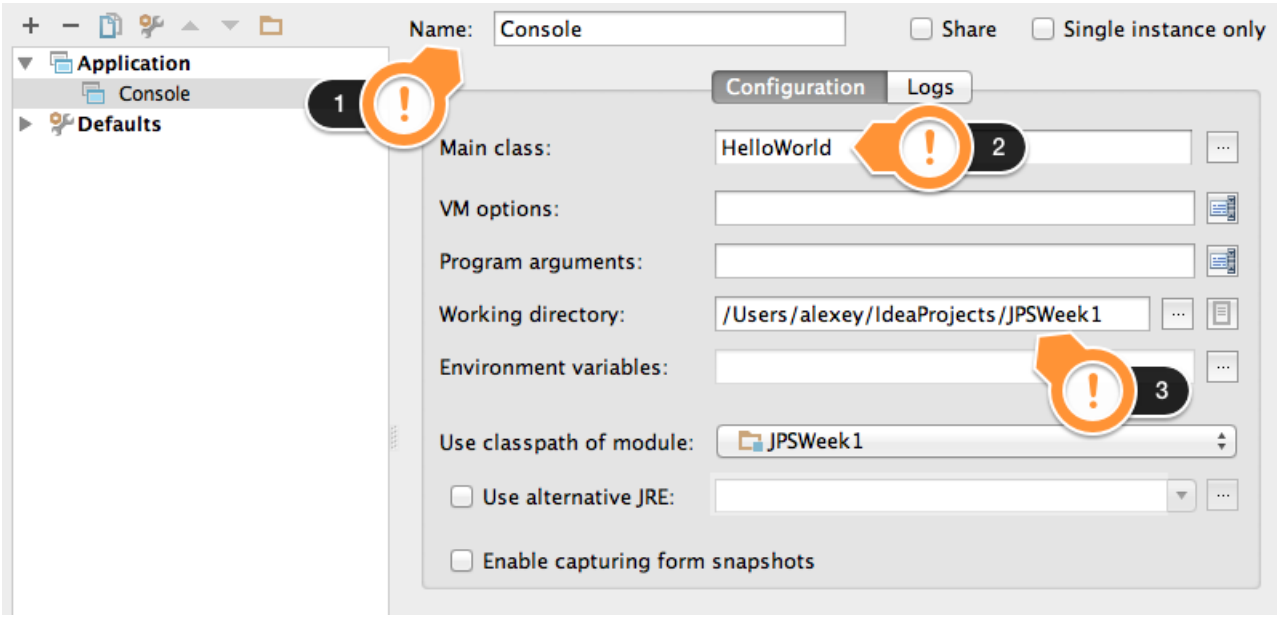

Рис. 1.11. Налаштування конфігурації запуску

Запуск та налагодження

Запустити модуль на виконання можна, вибравши команду Run (Shift + F10) або Debug (Shift + F9) з меню Run, або з контекстного меню класу додатка (який містить метод main()). Для консольних додатків IDEA не відкриває системну консоль, а направляє вихідний потік додатку в спеціальну панель Run.

У вікні налагоджувача (рис. 1.12) ви можете бачити стек викликів методів і список потоків, які наразі виконуються, разом з їх поточним станом і змінними. Вибравши контекст виклику методу, можна переглянути значення всіх змінних, присутніх в цьому контексті.

Іноді при покроковому налагодженні треба перейти всередину певного методу, але не першого який буде викликаний. В такому випадку ви можете натиснути Shift + F7 щоб вибрати із запропонованого списку метод, який потрібен. Це може заощадити масу часу.

Якщо потрібно «повернутися назад у часі» під час налагодження, ви можете зробити це, видаливши контекст виклику функції. Це сильно допоможе, якщо помилково зайдено занадто глибоко. Таким чином ви як мінімум повернетеся назад по стеку викликів функцій.

Іноді є необхідність відновити виконання програми і зупинитися на якомусь іншому рядку коду, не створюючи точку зупину. Це легко зробити – треба просто натиснути Alt + F9.

Якщо потрібно бачити якийсь елемент під час налагодження, то можна додати до нього кольорову мітку, натиснувши F11 або вибравши відповідний пункт в меню вкладки Variables і Watches. Коли цей елемент з'явиться в списку, тоді легко побачити його мітку. У режимі налагодження можна вирахувати будь-який вираз за допомогою дуже потужного інструменту натисканням Alt + F8.

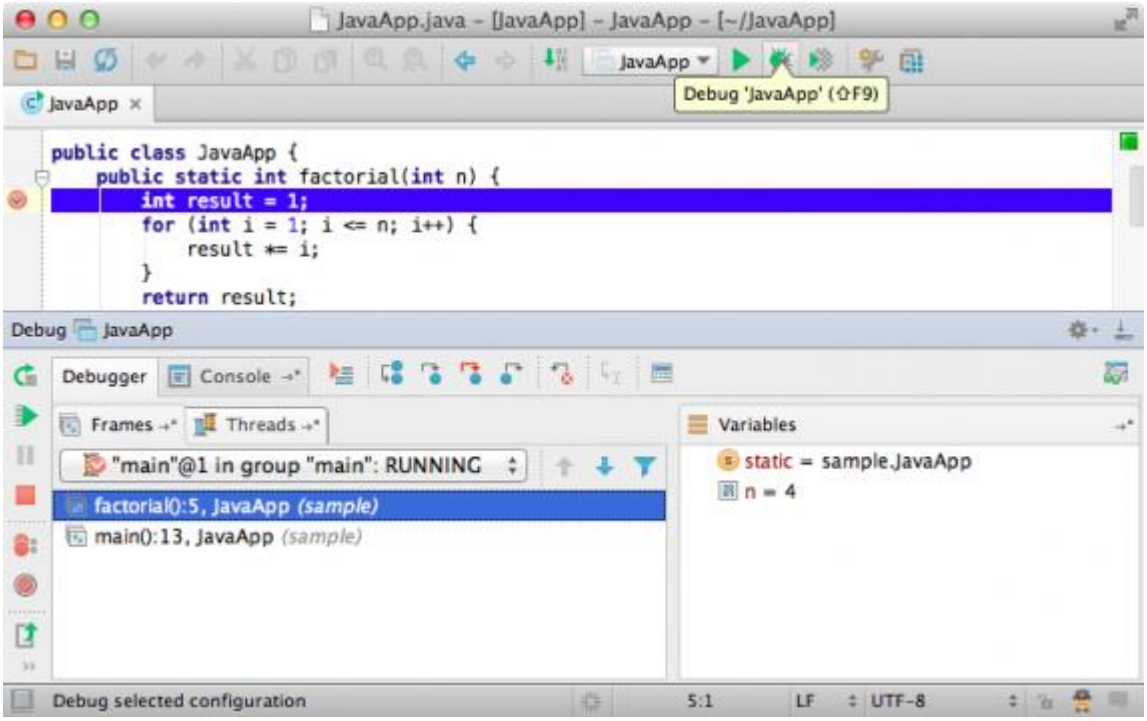

Рис. 1.12. Робота налагоджувача

Корисні клавіатурні скорочення налагоджувача

- Встановити/зняти точку зупинки – *Ctrl + F8;*
- Змінити налаштування точки зупинки (задати умову або зробити зупинку тільки для одного потоку) – *Shift + Ctrl + F8;*
- Відновити виконання програми – *F9;*
- Перейти до наступної інструкції – *F8*;
- Перейти всередину функції – *F7;*
- Виконання до позиції курсору – *Alt + F9;*
- Призупинити виконання – *Ctrl + F2;*
- У режимі налагодження ви можете обчислити будь-який вираз за допомогою дуже потужного інструменту, що викликається натисканням *Alt + F8.*

Точка зупинки змінної – це можливість налагоджувача зупинити виконання програми, коли проводиться читання або запис в задану змінну. Для того щоб створити таку точку зупинки, натисніть на панелі зліва від оголошеної змінної, затиснувши *Alt* (рис. 1.13).

Якщо ви хочете поміняти якісь настройки точки зупину, ви можете натиснути Shift + Ctrl + F. У спливаючому вікні ви можете ввести потрібні вам параметри.

Для того щоб отримати список всіх точок зупину у вашому проекті з розширеними налаштуваннями, треба знову натиснути Shift + Ctrl + F8.

На додаток до умовних точок зупину, ви можете також використовувати точки зупину змінної. Такі точки спрацьовують, коли проводиться читання або запис в якусь змінну. Для того щоб створити таку точку зупину, клацніть на панелі зліва від тексту, що редагується, навпроти потрібної вам змінної, затиснувши Alt.

Існує ще одна корисна можливість – обчислити певний вираз в потрібному рядку коду, не перериваючи виконання. Для цього потрібно клацнути на панелі зліва від редагованого коду навпроти потрібного рядка, затиснувши Shift.

З метою створення точки зупинки, яка спрацює тільки один раз, клацніть на панелі зліва від коду затиснувши Shift + Alt.

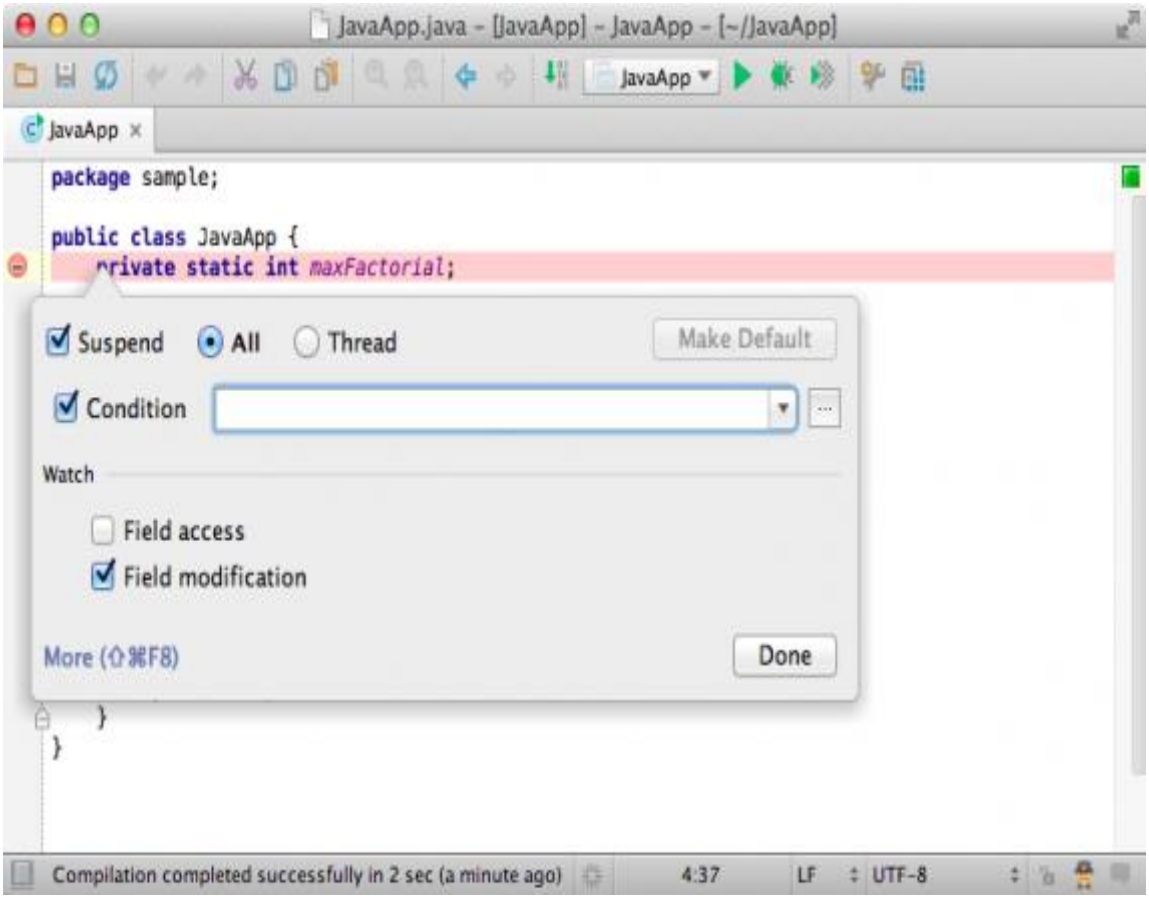

Рис. 1.13. Налаштування точки зупинки змінної

Доповнення коду

Перед тим, як розглянути варіанти доповнення коду, приведемо найбільш часто вживані так звані «гарячі клавіші» :

1. Ctrl + Space – список компонентів (класу, метада, змінної);
2. Ctrl + Q – швидкий пошук документації;
3. Shift + F1 – зовнішня документація;
4. Ctrl + наведення миші на фрагмент коду – коротка інформація;
5. Ctrl + F1 – показати опис помилки;
6. Ctrl+ / – однорядкове коментування/роз коментування;
7. Alt + Q – контекстна інформація;
8. Alt + Enter – показати пропоноване виправлення;
9. Ctrl + Alt + L – форматування коду;
10. Ctrl + X або Shift + Delete – вирізати фрагмент коду;
11. Ctrl + C або Ctrl + Insert – копіювати фрагмент коду;
12. Ctrl + D – дублювання рядка;
13. Ctrl + Y – видалення рядка;
14. Ctrl + P – відомості про параметри.

Основною метою IntelliJ IDEA є продуктивність розробника, і для цього вона надає потужні можливості автоматичного доповнення коду. Є кілька функцій доповнення коду, то ж важливо розібратися, як вони працюють і яку з них краще використовувати в кожній ситуації.

1) *Просте доповнення* пропонує найпримітивніші припущення імен змінних, типів, методів, виразів і т. д. (рис. 1.14). Воно автоматично працює по мірі введення тексту. Якщо список варіантів закінчення слова не відображається, його можна викликати примусово – *Ctrl + пробіл*. Зверніть увагу, якщо ви викликаєте простий додаток двічі, він покаже вам більше варіантів, включаючи приватні члени.

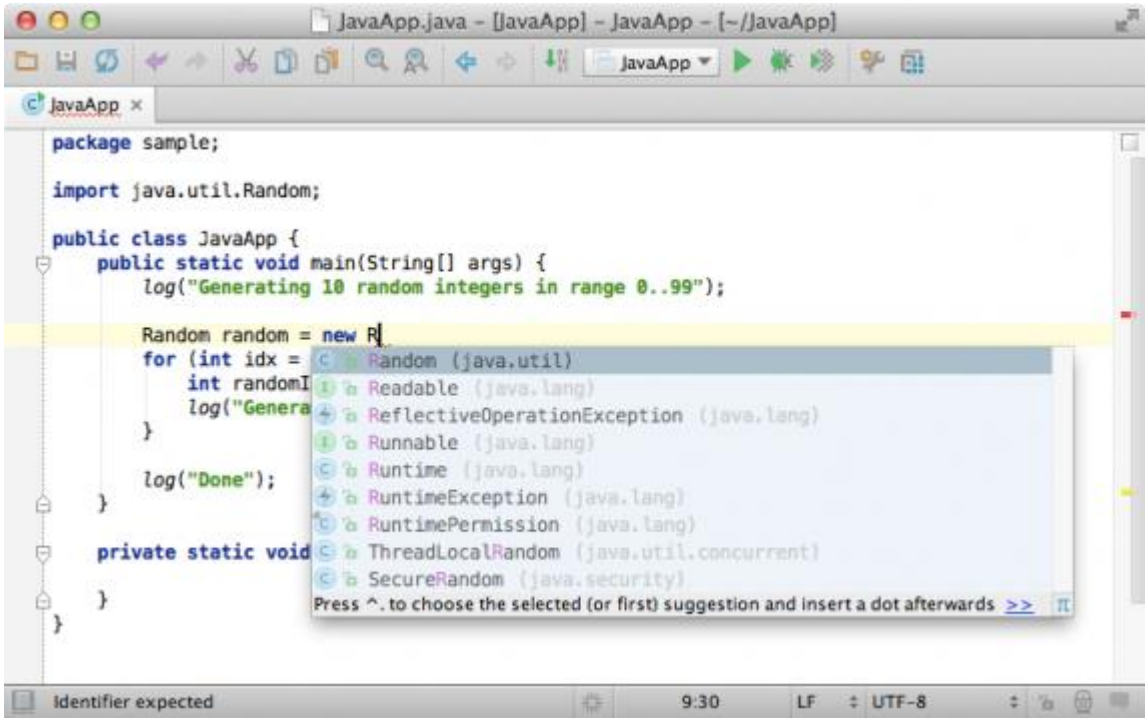

Рис. 1.14. Просте доповнення

2) *Розумне доповнення* для підбору варіантів аналізує контекст, в якому знаходиться слово що вводиться, тобто розбирається в тому, який тип виразу очікується в даномі місці коду, і як відбувається перетворення типів даних, та пропонує тільки ті варіанти, які підходять за контекстом (рис. 1.15). Для того щоб запустити *Розумне доповнення*, натисніть *Shift + Ctrl + пробіл*.

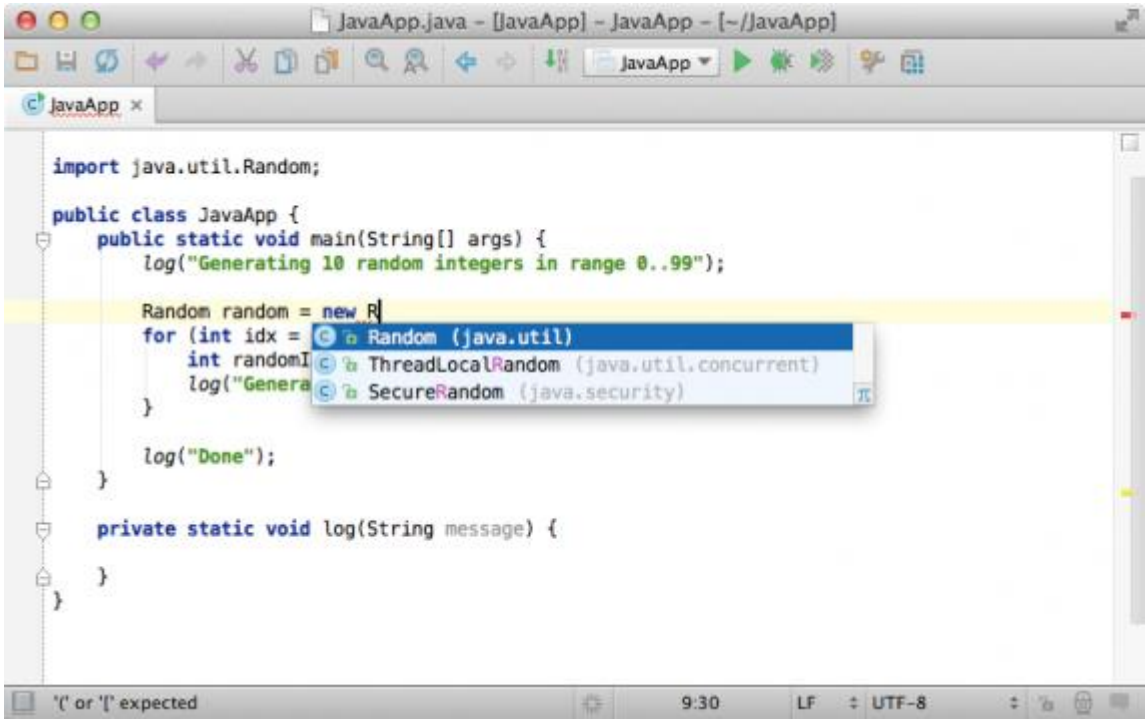

Рис. 1.15. Розумне доповнення підказує найбільш ймовірне продовження слова

Зверніть увагу, що якщо ви запустите *Розумне доповнення* двічі, воно покаже вам більше результатів, включаючи ланцюжки і не імпортовані статичні члени (рис. 1.16).

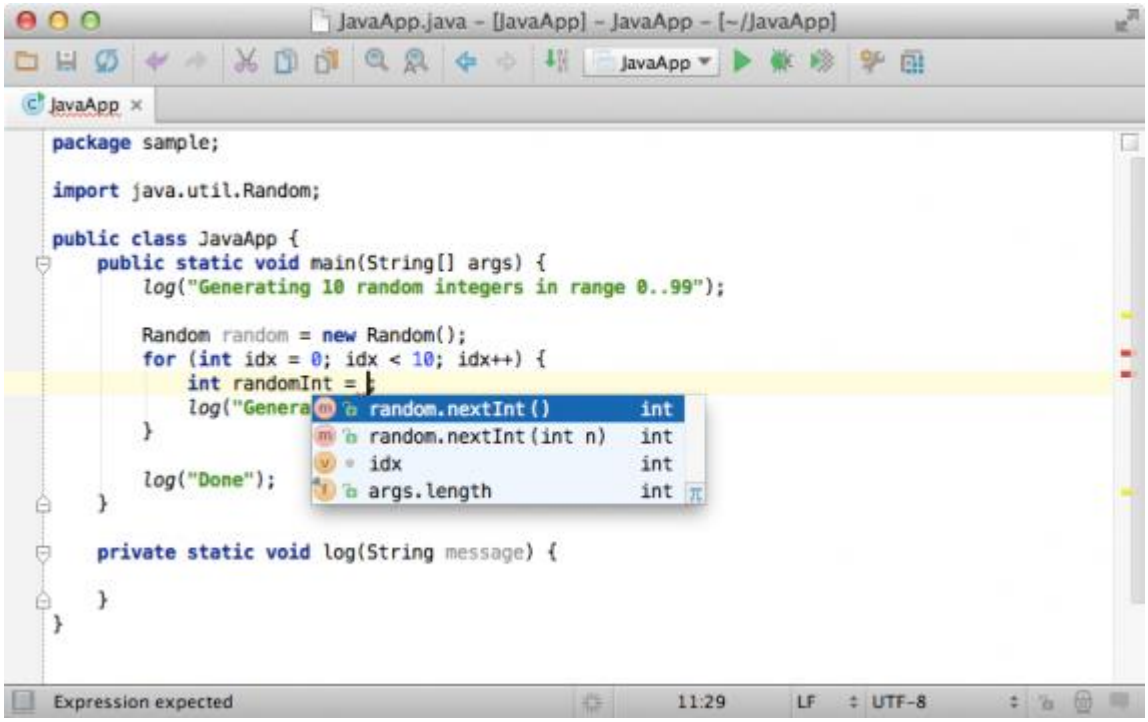

Рис. 1.16. Розумне доповнення підказує найбільш ймовірні ланцюжки за змістом

3) *Доповнення за клавішею Tab*. Якщо під час вибору елементу зі списку запропонованих варіантів натиснути *Tab*, то обраний варіант замінить собою ідентифікатор, на якому знаходиться курсор. Це буває корисно, якщо ви редагуєте частину ідентифікатора, наприклад, частину назви класу (рис. 1.17).

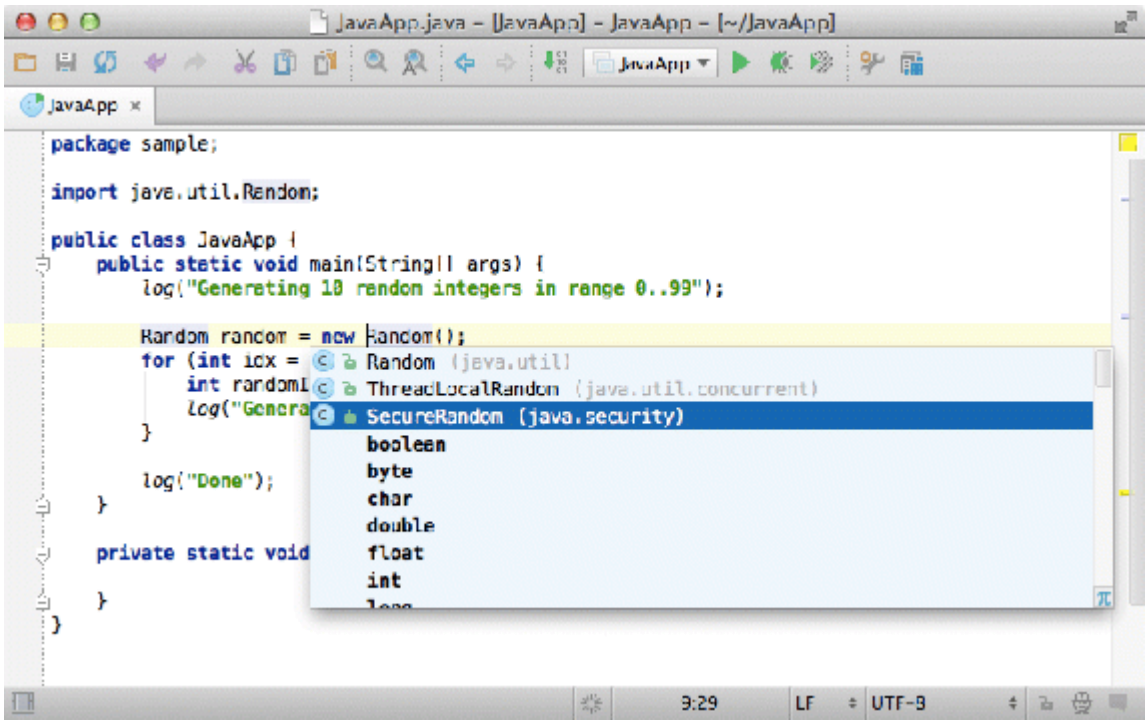

Рис. 1.17. Доповнення за клавішею *Tab*.

4) *Доповнення операторів* автоматично додає відсутні дужки, обробники до секцій *try* і необхідне форматування. Щоб доповнити оператор, просто натисніть *Shift + Ctrl + Enter*.

5) *Серединна відповідність* дозволяє не писати ідентифікатор з самого початку. Наприклад, якщо ви пам'ятаєте тільки середню частину імені класу або методу, напишіть її, і IntelliJ IDEA знайде правильний варіант (рис. 1.18).

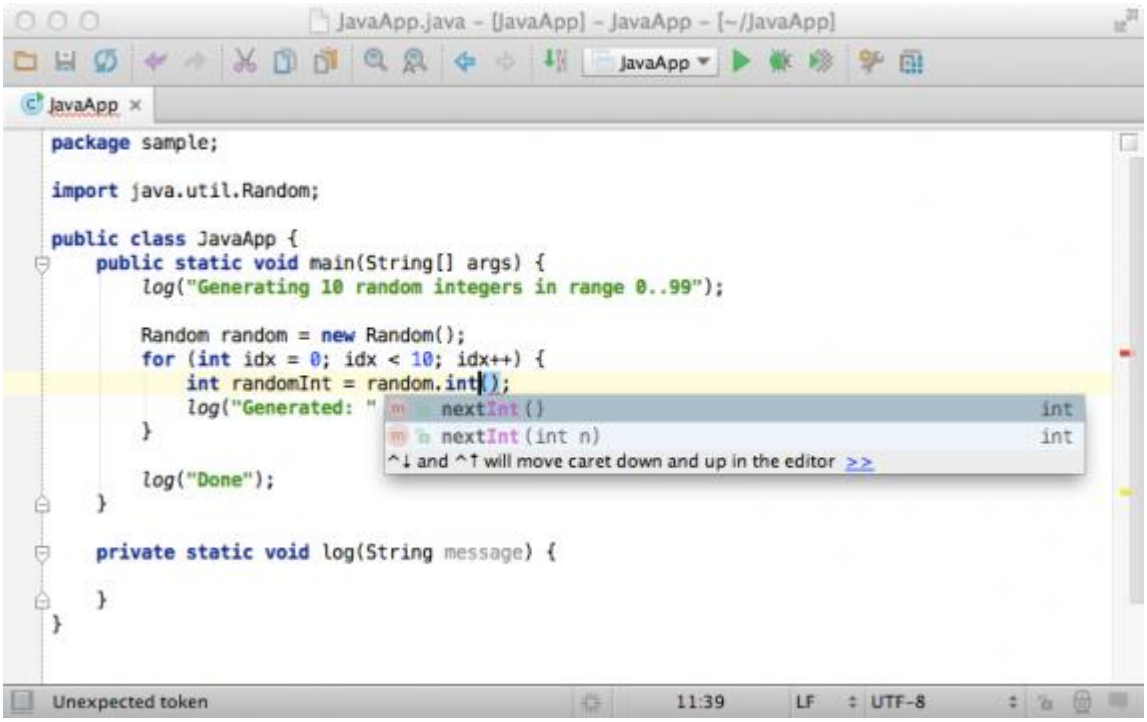

Рис. 1.18. Серединна відповідність

6) *Скорочений запис* часто використовуваних фрагментів коду. Наприклад, заголовок методу main() завжди виглядає однаково і програмісту не потрібно кожен раз писати його заново. Якщо в порожньому рядку надрукувати `psvm` і натиснути *Ctrl+Пробіл*, то IDEA згенерує стандартний заголовок main():

```
public static void main (String [] args) {}
```

Якщо всередині методу надрукувати `fori` і натиснути *Ctrl+Пробіл*, буде згенеровано заготовку коду для найбільш поширеного циклу:

```
for (int i = 0; i <; i ++) {}
```

Якщо надрукувати `sout` і натиснути *Ctrl+Пробіл*, буде згенеровано заготовку коду для виведення на консоль:

```
System.out.println();
```

Спливаючі вікна та підказки

Спливаючі підказки IDEA корисні для перевірки додаткової інформації, що відноситься до елементу, на якому встановлений курсор. Наведемо список підказок, які допоможуть вам підвищити продуктивність праці.

1) Якщо ви вже відкрили документацію і хочете залишити документацію відкритою, клацніть на кнопку Pin в верхньому правому куті. Буде показана (і не зникне) документація по поточному елементу.

2) Спливаючі вікна з підказками можуть бути викликані для будь-якого елемента коду в вашому редакторі. Крім того, ці підказки можуть бути отримані і для елементів в будь-якому іншому списку, якщо при цьому використовуються такі самі сполучення клавіш.

3) *Інформація про параметри*, необхідні для виклику методу доступна після натискання *Ctrl+P*. Спливаюче вікно показує відповідний перевантажений метод, виділяючи поточний параметр (рис. 1.19).

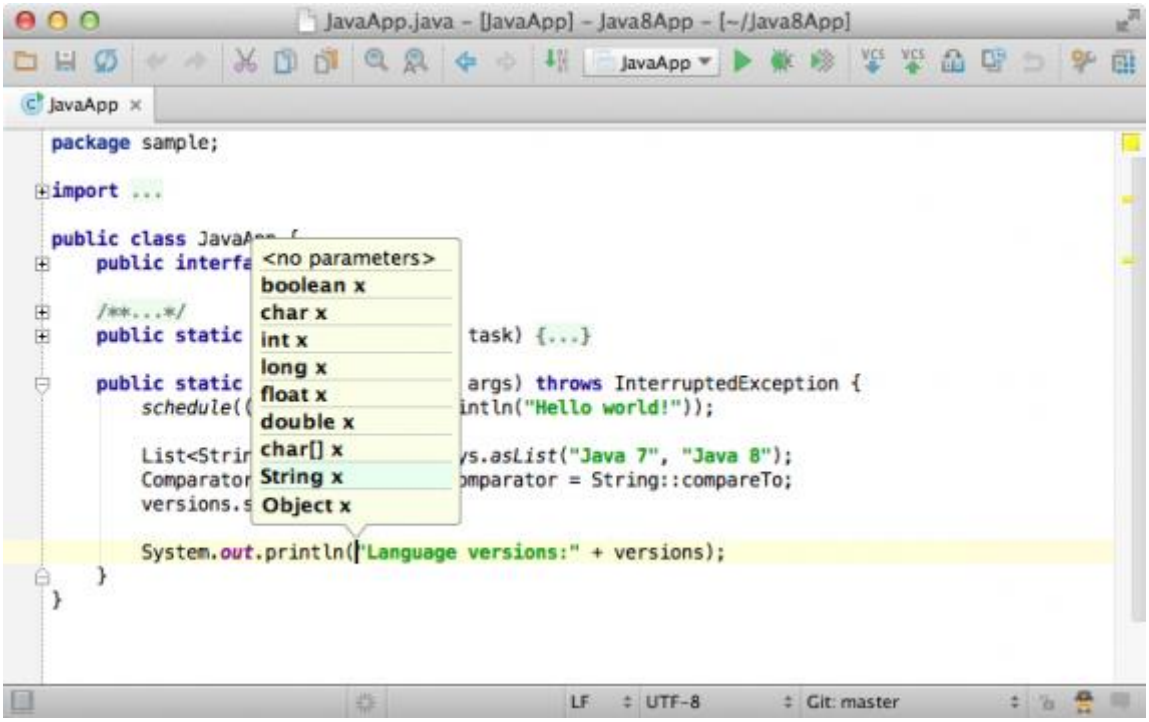

Рис. 1.19. Підказка: параметри методу

4) *Швидкий опис*. Для швидкого перегляду реалізації методу, на якому встановлений курсор, викличіть швидкий опис натиснувши *Shift + Ctrl + I* (рис. 1.20).

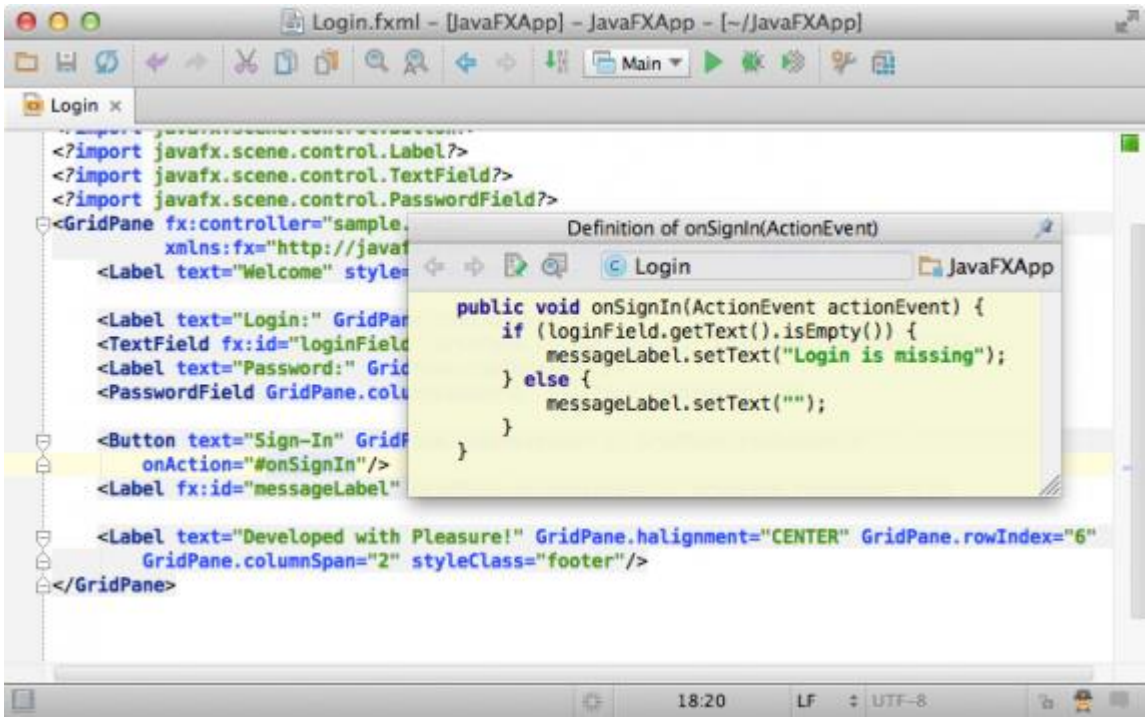

Рис. 1.20. Вікно «definition» показує, як оголошений в коді елемент,
на якому знаходиться курсор

5) *Документація*. Спливаюче вікно з документацією є одним з найбільш часто використовуваних. Воно відображає коротку довідку за класом або методом (рис. 1.21, 1.22). Щоб його відкрити, помістіть курсор на імені метода або класу та натисніть *Ctrl + Q*.

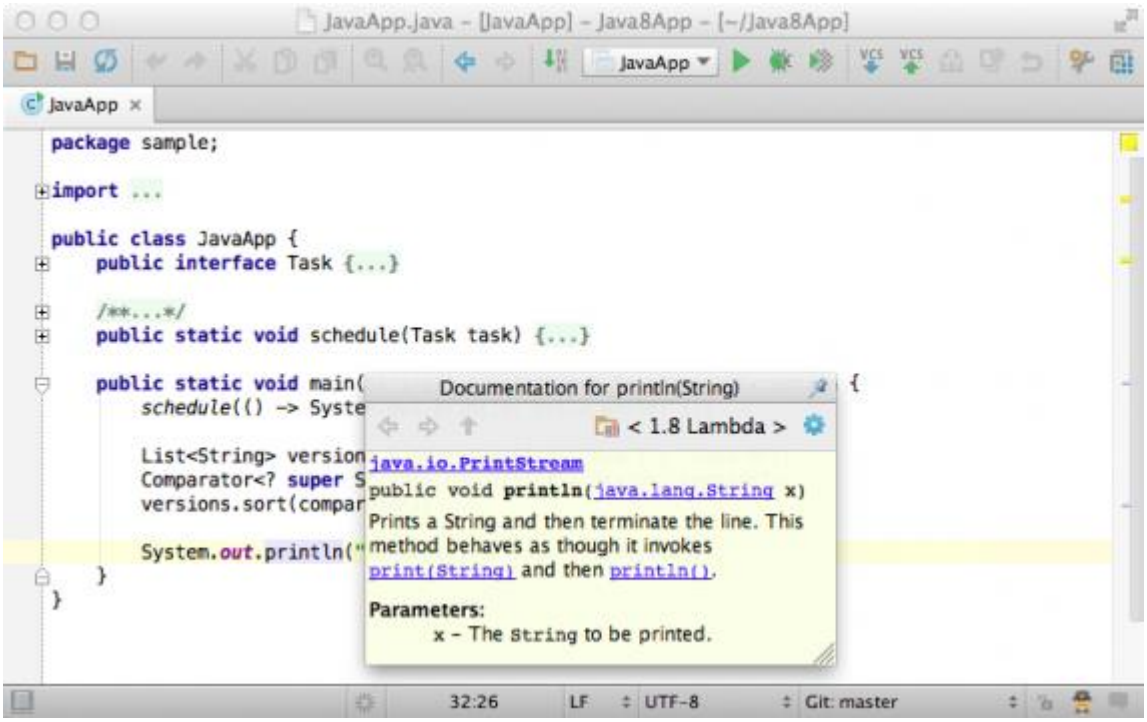

Рис. 1.21. Документація за стандартним методом println()

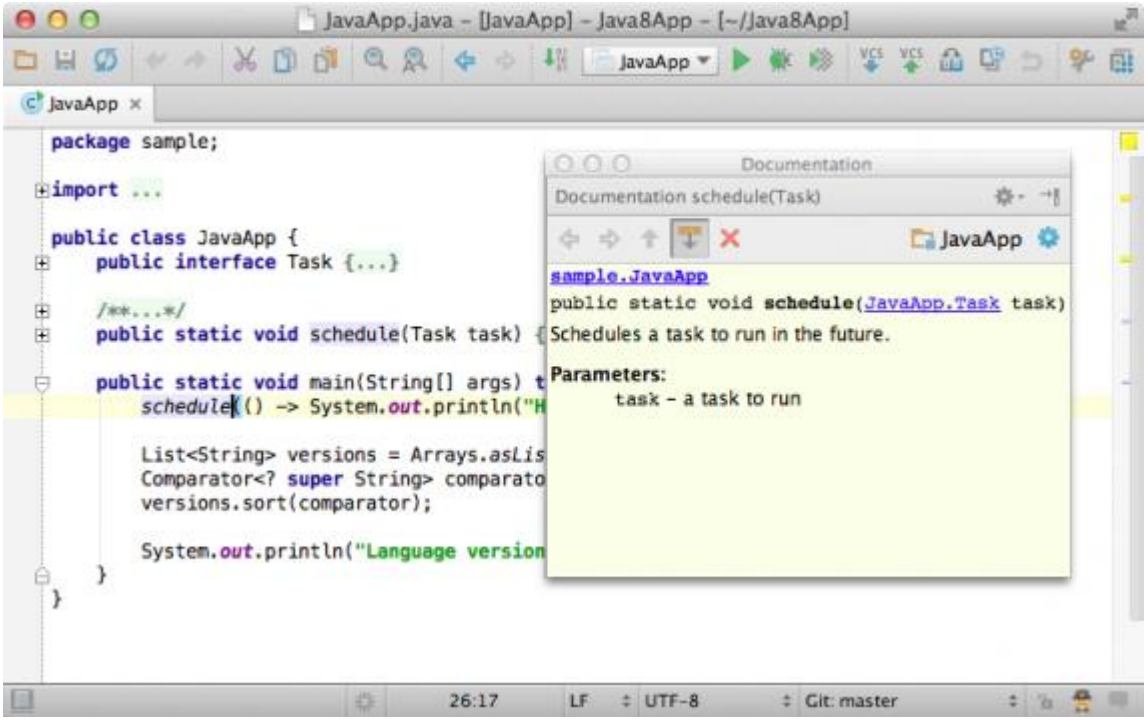

Рис. 1.22. Документація за методом shedule() з поточного проекту

Підказки без спливаючих вікон

Якщо вас не влаштовують "спливаючі" вікна, ви можете приєднати їх до краю головного вікна середовища, натиснувши на кнопку в правому верхньому кутку спливаючого вікна (рис. 1.23).

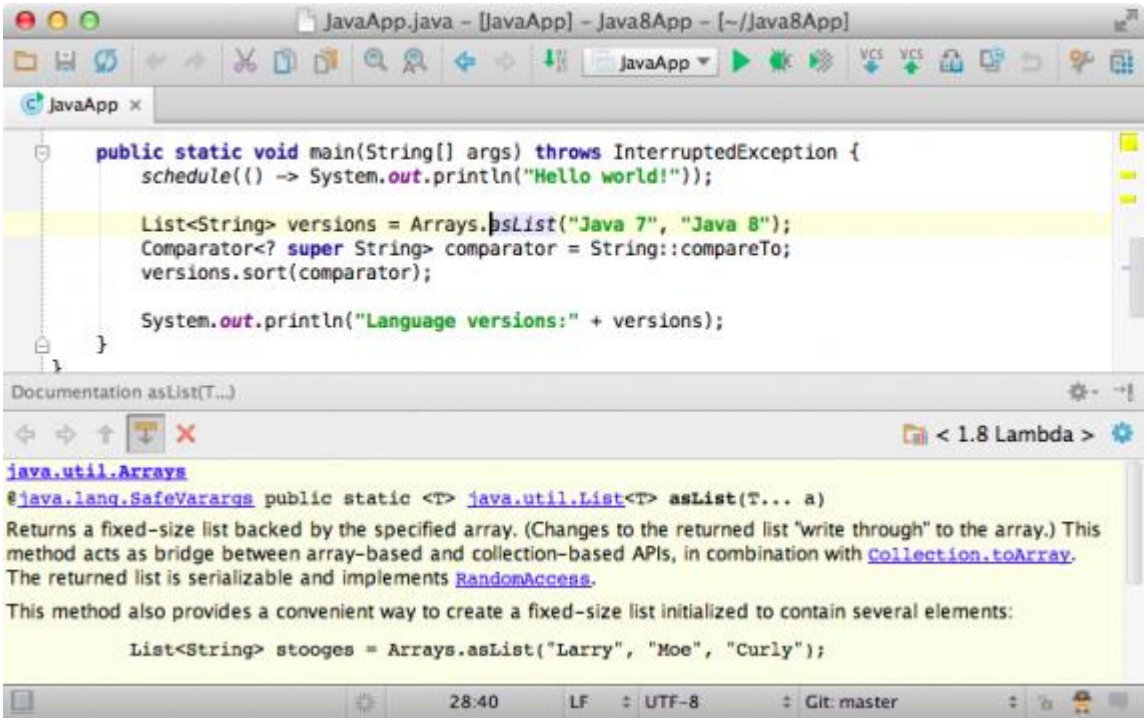

Рис. 1.23. Відображення підказок в панелі головного вікна

Спливаючі вікна в інших компонентах середовища розробки

Добре відомо, що спливаючі вікна з підказками можуть бути викликані для будь-якого елемента коду в вашому редакторі. Однак, також вони доступні і для елементів в будь-якому іншому списку або панелі при використанні таких же сполучень клавіш (рис. 1.24).

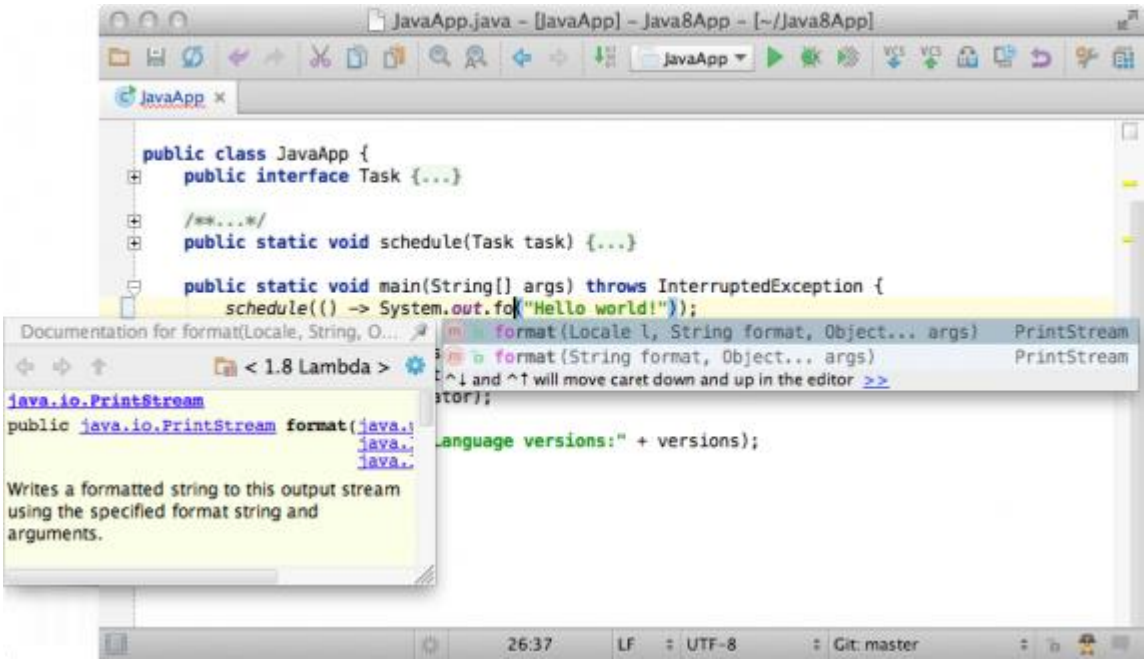

Рис. 1.24. Документація стосовно одного з елементів випадного списку в коді

Ви можете викликати спливаючі підказки навіть коли перебираєте елементи на панелі навігації після натискання *Alt + Home* (рис. 1.25).

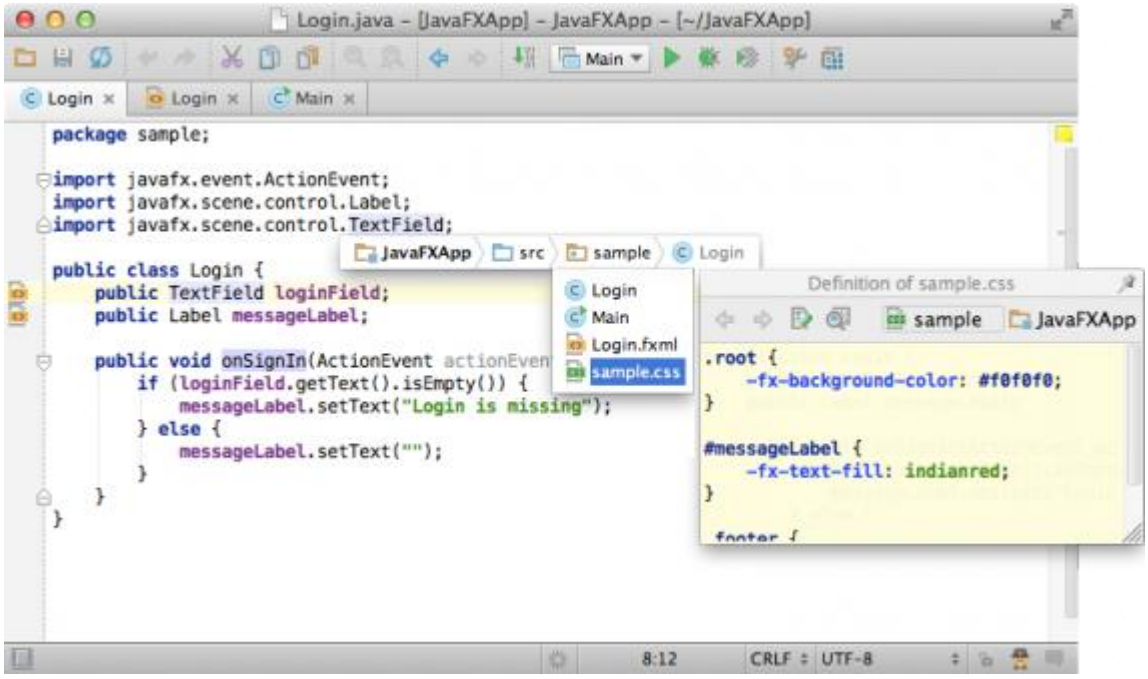

Рис. 1.25. Підказка «визначення» стосовно одного з елементів проекту

"Показати місця використання"

"Показати місця використання" – це дуже проста, але потужна функція. Помістіть курсор на ім'я змінної, методу або класу, натисніть *Alt + Ctrl + F7*, і ви побачите список тих місць в вашому коді, де використовується елемент (змінна, метод або класс), що знаходиться під курсором (рис. 1.26).

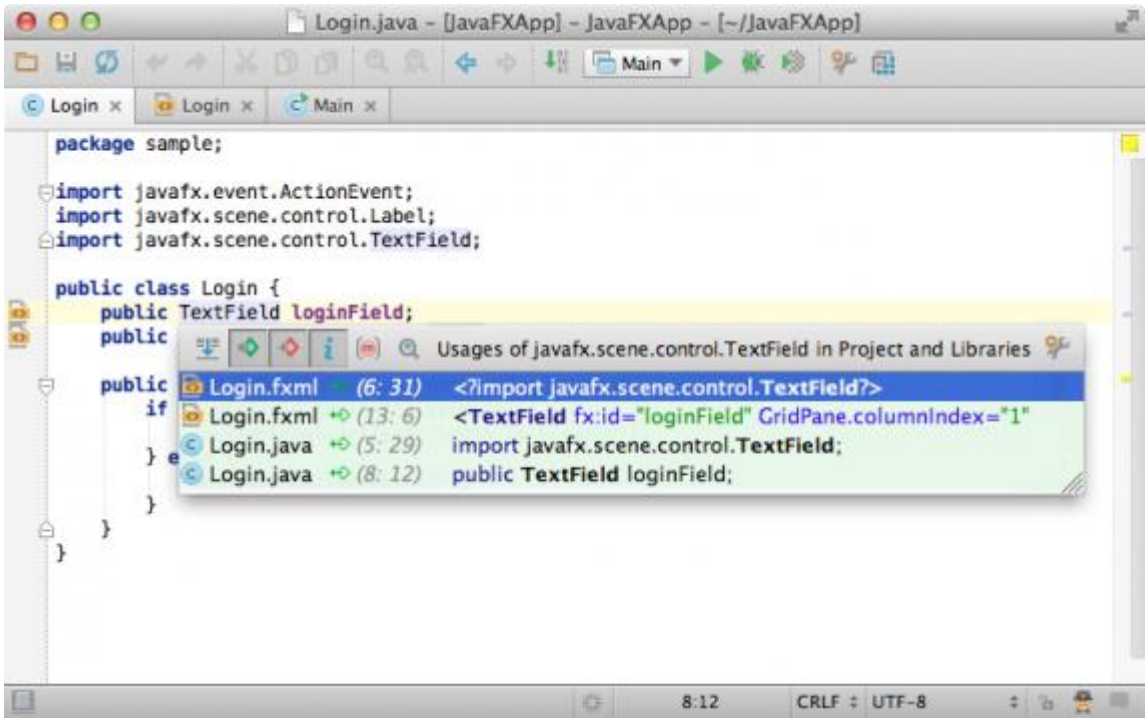

Рис. 1.26. Підказка «використання»

Основні гарячі клавіші IDEA в зручному вигляді перераховані тут: http://eax.me/intellij-idea-hotkeys/.

Крім основної IDE, коротко розглянемо також два її конкуренти – Eclipse і NetBeans – щоб розібратися, в яких випадках вони можуть бути корисні.

Eclipse

Eclipse – це вільне інтегроване середовище розробки модульних крос-платформових додатків, яке розвивається і підтримується компанією Eclipse Foundation.

Eclipse відрізняється модульною розширюваною архітектурою, маючи безліч розширень (плагінів) для всіляких завдань розробки. Eclipse також слугує платформою для розробки таких плагінів, чим він і завоював свою популярність: будь-який розробник може розширити Eclipse власними модулями-плагінами. Існують комерційні і некомерційні розширення, найбільш популярними з них є:

- розширення середовища Eclipse для роботи з базами даних, серверами додатків і ін.;
- модулі, що спрощують створення графічних інтерфейсів (наприклад, Window Builder);
- Eclipse JDT (Java Development Tools), який призначений для групової розробки та підтримує системи управління версіями – CVS, GIT;
- розширення для роботи з мовами програмування C/C++, Perl, PHP, JavaScript, Python, Ruby.

Eclipse написана на Java, тому є платформо-незалежним продуктом, тобто може працювати під усіма популярними ОС.

Основними перевагами Eclipse в порівнянні з IntelliJ IDEA є безкоштовність і велика гнучкість за рахунок плагінів. Однак в IDEA набагато краще розвинені засоби інтелектуального аналізу коду. Ці засоби дозволяють, зокрема, при підказці варіантів продовження враховувати весь контекст коду і пропонувати тільки ті варіанти, які підходять і за типом, і за семантикою.

Також IDEA має набагато більше можливостей авторефакторинга і генерації коду. Тому, якщо є можливість платити за ліцензію, зазвичай вибирають IntelliJ IDEA, а Eclipse залишається лідером серед вільних IDE.

NetBeans IDE

Це вільна IDE для мов Java, Python, PHP, JavaScript, C, C++. Проект NetBeans IDE підтримується і спонсорується компанією Oracle, однак розробка NetBeans ведеться незалежною спільнотою розробників-ентузіастів (NetBeans Community) і компанією NetBeans Org.

За якістю і можливостями останні версії NetBeans IDE наближаються до кращих комерційних (платних) IDE, таких як IntelliJ IDEA, підтримуючи рефакторинг, профілювання, виділення синтаксичних конструкцій кольором, автодоповнення набраних конструкцій на льоту, безліч вбудованих шаблонів коду. Перевагою цієї IDE також є те, що вже базова версія має повну підтримку технологій Java EE (тобто для розробки веб-додатків не потрібно нічого встановлювати додатково).

Однак в плані зручності розробки NetBeans IDE поступається IntelliJ Idea, оскільки не має настільки розвинених інтелектуальних засобів аналізу коду, а у багатьох задачах (особливо пов'язаних з J2EE-розробкою) вимагає набагато більше ручної роботи зі створення інтерфейсів, дескрипторів та ін.