

Об'єктно-орієнтоване програмування

Додаткові матеріали

ЗМІСТ

1	Додаток А. Налаштування середовища розробки.....	3
1.1	Інсталяція та налаштування JDK.....	3
1.2	Перевірка працездатності платформи Java.....	4
1.3	Інсталяція та налаштування Eclipse.....	5
2	Додаток В. Основи роботи в середовищі Eclipse.....	8
2.1	Робочий простір Eclipse.....	8
2.2	Створення нового проекту на Java.....	9
2.3	Архітектура платформи Eclipse.....	14
3	Додаток С. Стил ь кодування.....	17
3.1	Рекомендації з використання імен.....	17
3.2	Документування програмного коду.....	17
3.2.1	Коментарі документації.....	18
3.2.2	Загальна форма коментарів.....	18
3.2.3	Дескриптори javadoc.....	21
3.3	Структура проекту.....	22

1 Додаток А. Налаштування середовища розробки

Для самостійної роботи студентів, а так само рішення задач практичних і лабораторних робіт рекомендується використовувати засновану на Java розширювану платформу розробки з відкритим початковим кодом Eclipse. Попередньо необхідно встановити Java Runtime Environment (JRE) або краще Java Development Kit (JDK).

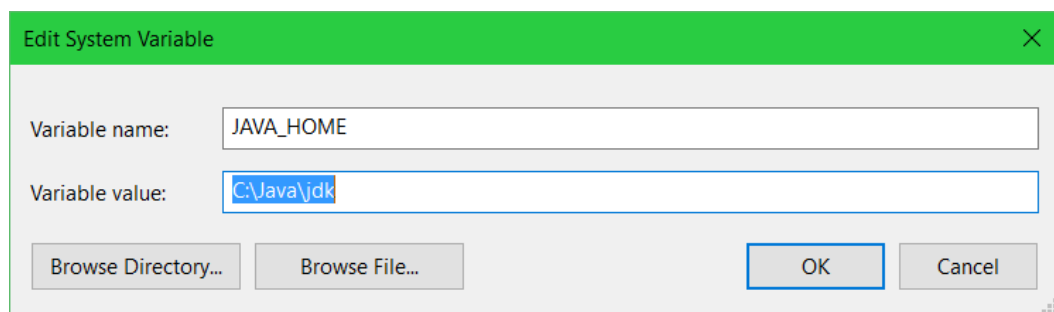
1.1 Інсталяція та налаштування JDK

1) Виконати інсталяцію Java Development Kit (JDK) з сайту Oracle, розділ "Java Platform, Standard Edition".

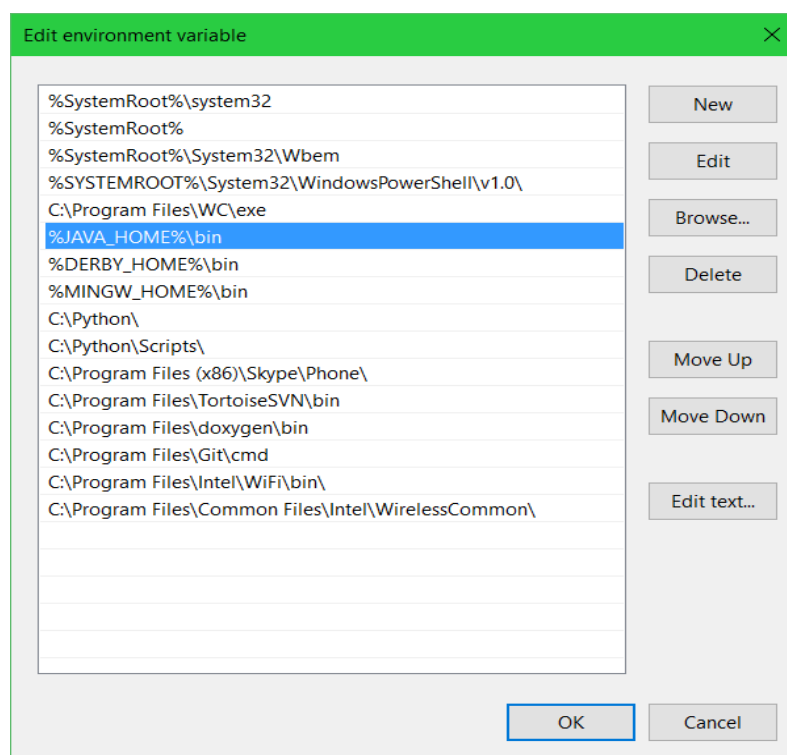
2) Під час інсталяції:

- відмовитися від Public JRE.
- вказати простий шлях для інсталяції, наприклад, "C:\Java\jdk\".

3) Встановити змінну середовища оточення "JAVA_HOME"



4) Додати в Path шлях "%JAVA_HOME%\bin"



1.2 Перевірка працездатності платформи Java

1) Переконайтеся в доступності компілятора Java, що відповідає раніше встановленій версії JDK, з командного рядка консолі (комбінація клавіш "Win+R", команда "cmd.exe") увести команду:

```
javac -version
```

2) У будь-якому каталозі створити текстовий файл One.java з вмістом:

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

3) У тому ж каталозі в командному рядку увести команди:

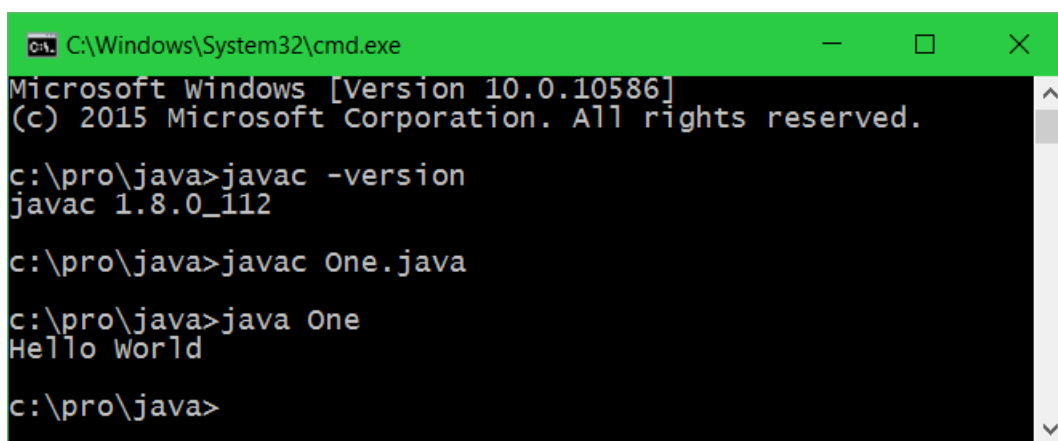
- для компіляції програми

```
javac One.java
```

- для виконання та перегляду результатів роботи

```
java One
```

Очікуваний результат (при відсутності помилок і коректній установці JDK):

A screenshot of a Windows Command Prompt window. The title bar is green and shows 'C:\Windows\System32\cmd.exe'. The window has standard minimize, maximize, and close buttons. The command prompt shows the following text: 'Microsoft Windows [Version 10.0.10586] (c) 2015 Microsoft Corporation. All rights reserved.' followed by a series of commands and their outputs: 'c:\pro\java>javac -version' outputs 'javac 1.8.0_112'; 'c:\pro\java>javac One.java' outputs nothing; 'c:\pro\java>java One' outputs 'Hello World'; and 'c:\pro\java>' shows the prompt again.

```
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.10586]  
(c) 2015 Microsoft Corporation. All rights reserved.  
  
c:\pro\java>javac -version  
javac 1.8.0_112  
  
c:\pro\java>javac One.java  
  
c:\pro\java>java One  
Hello World  
  
c:\pro\java>
```

У випадку помилки при виконанні команди "javac" з повідомленням "javac' is not recognized as an internal or external command, operable program or batch file" слід перевірити наявність шляху в змінній середовища оточення "PATH" до утиліт каталогу "BIN" встановленого JDK.

1.3 Інсталяція та налаштування Eclipse

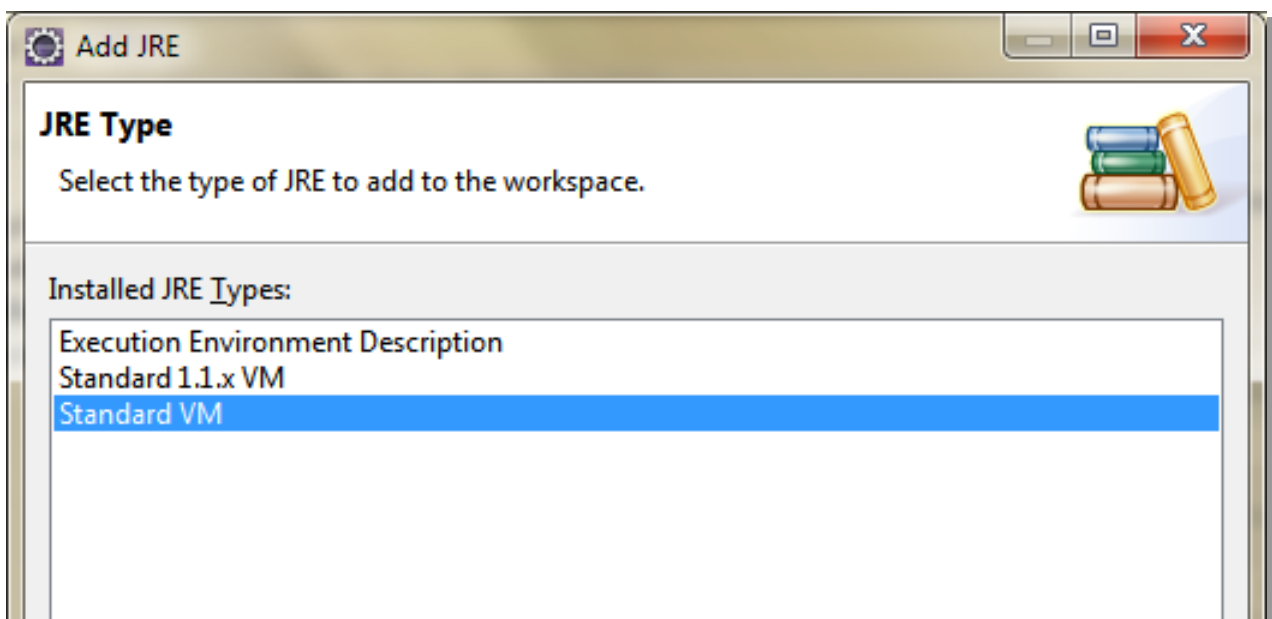
Рекомендується встановити пакет (версії не нижче Eclipse Kepler 4.3.2 SR2)

"Eclipse IDE for Java Developers" або "Eclipse IDE for Java EE Developers".

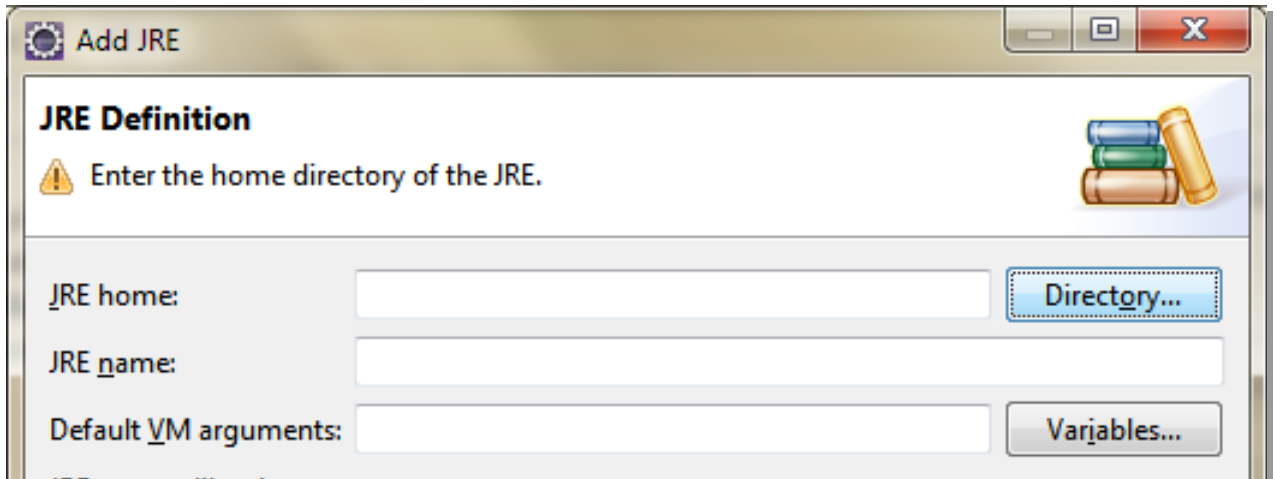
Необхідно витягти вміст архіву (наприклад, "eclipse-java-mars-R-win32-x86_64.zip") у кореневий каталог обраного накопичувача й виконати запуск файлу "eclipse.exe".

Для полегшення процесу розробки програм (доступу до розширених довідкових даних засобами Javadoc) рекомендується включити до списку робочих шляхів нових проектів JDK (замість обраного за замовчуванням JRE). Для цього необхідно через меню "Window / Preferences" відкрити вікно налаштування "Java / Installed JREs":

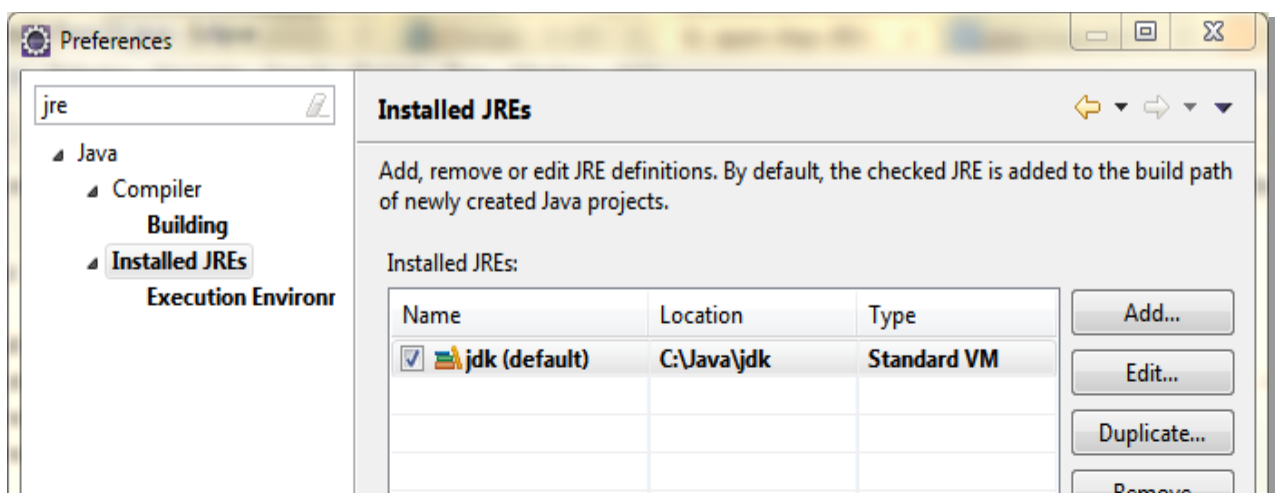
- обрати (відмітити) встановлений раніше JDK;
- якщо JDK відсутня в списку "Installed JREs", виконати команду "Add":
 - для "JRE Type" обрати "Standart VM"



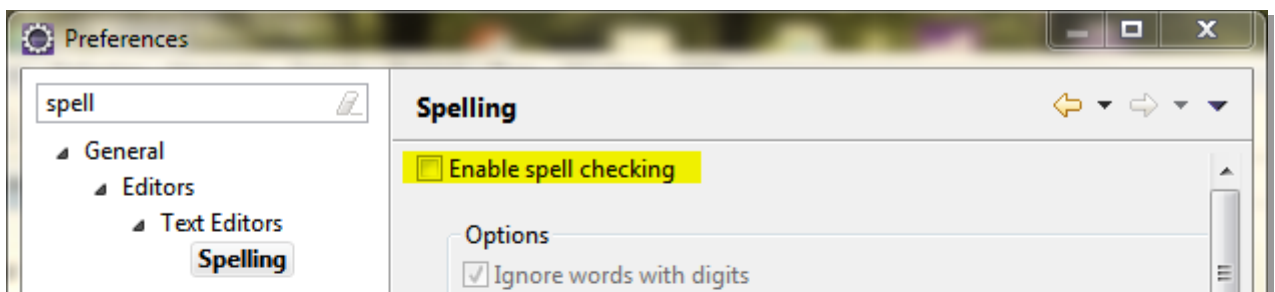
- для "JRE home / Directory" вказати кореневий каталог встановленого JDK



- в результаті автоматичного пошуку повинно бути встановлено:

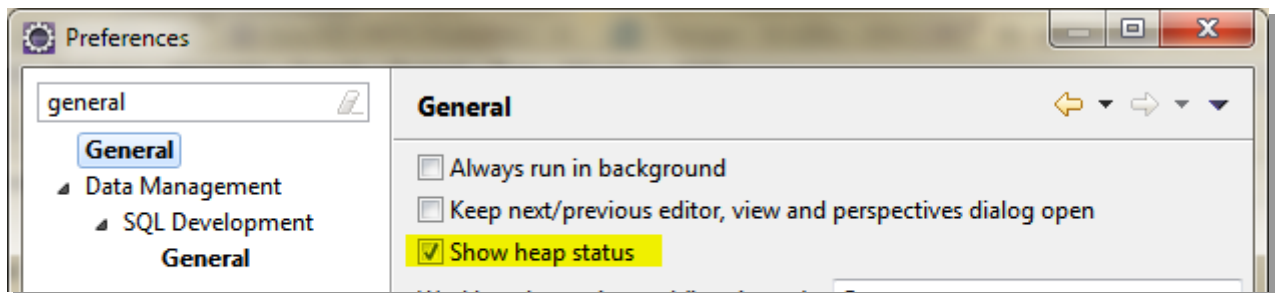


При використанні коментарів мовою, що відрізняється від англійської, рекомендується в опціях редактора відключити автоматичну перевірку орфографії: меню "Window / Preferences", розділ "General / Editors / Text Editors / Spelling", зняти позначку пункту "Enable spell checking"

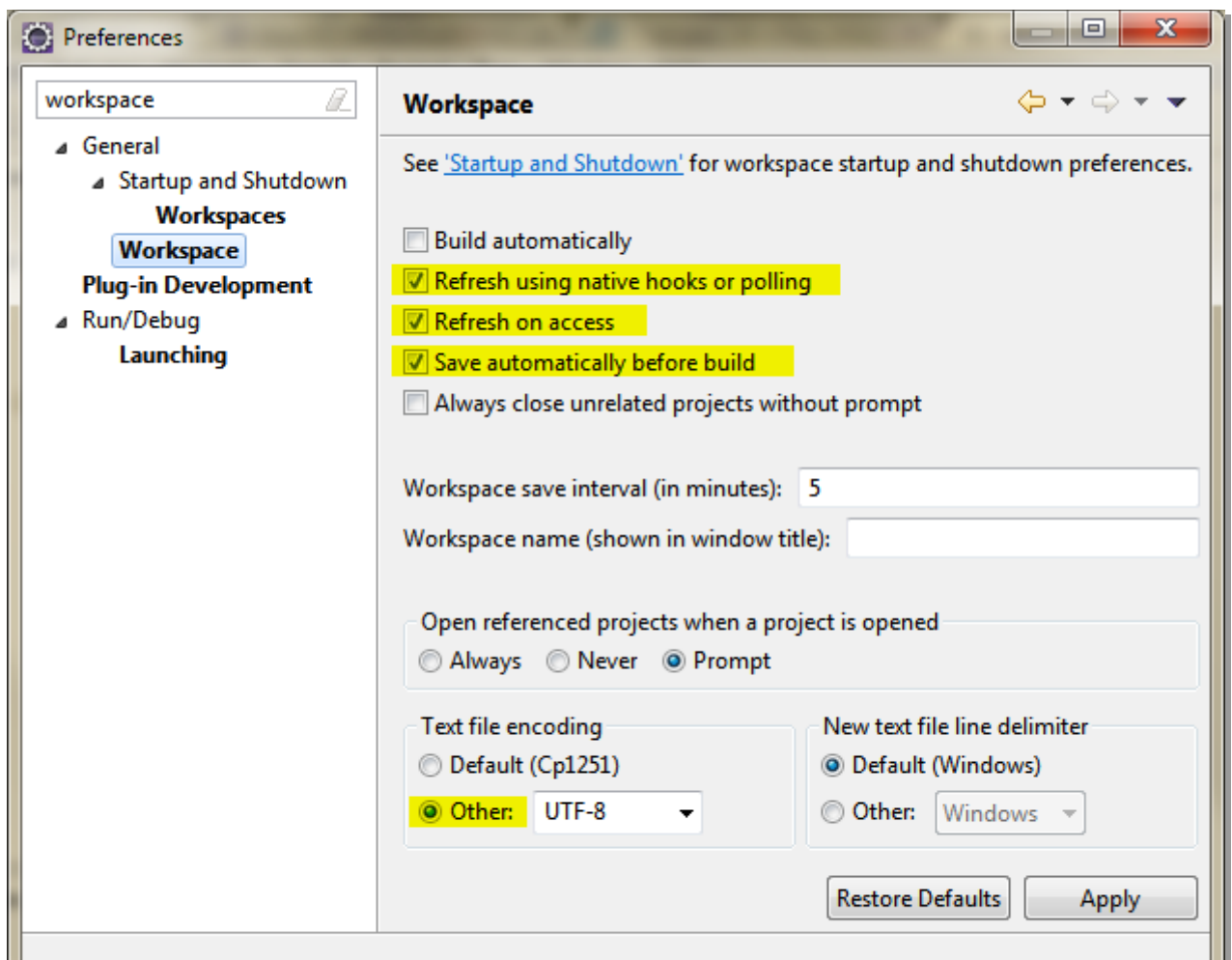


У вікні налаштувань (меню "Windows / Preferences"):

- ліворуч вибрати розділ "General", встановити флаг "Show heap status";



- у підрозділі "Editors / Text Editors" встановити флаг "Show line numbers";
- у підрозділі "Startup and Shutdown" встановити флаг "Refresh workspace on startup";
- у підрозділі "Workspace":
 - встановити флаг "Refresh using native hooks or polling";
 - встановити флаг "Save automatically before build";
 - у групі "Text file encoding" обрати поле "Other" і встановити значення "UTF-8".



2 Додаток В. Основи роботи в середовищі Eclipse

Проект Eclipse був створено в листопаді 2001 року компанією IBM і підтриманий консорціумом постачальників програмного забезпечення.

Eclipse являє собою засновану на Java розширювану платформу розробки з відкритим початковим кодом. По суті – це середовище розробки й набір сервісів для побудови додатків на основі компонентів, що вбудовуються (плагінів). У складі Eclipse є стандартний набір плагінів, у тому числі інструментарій Java Development Tools (JDT).

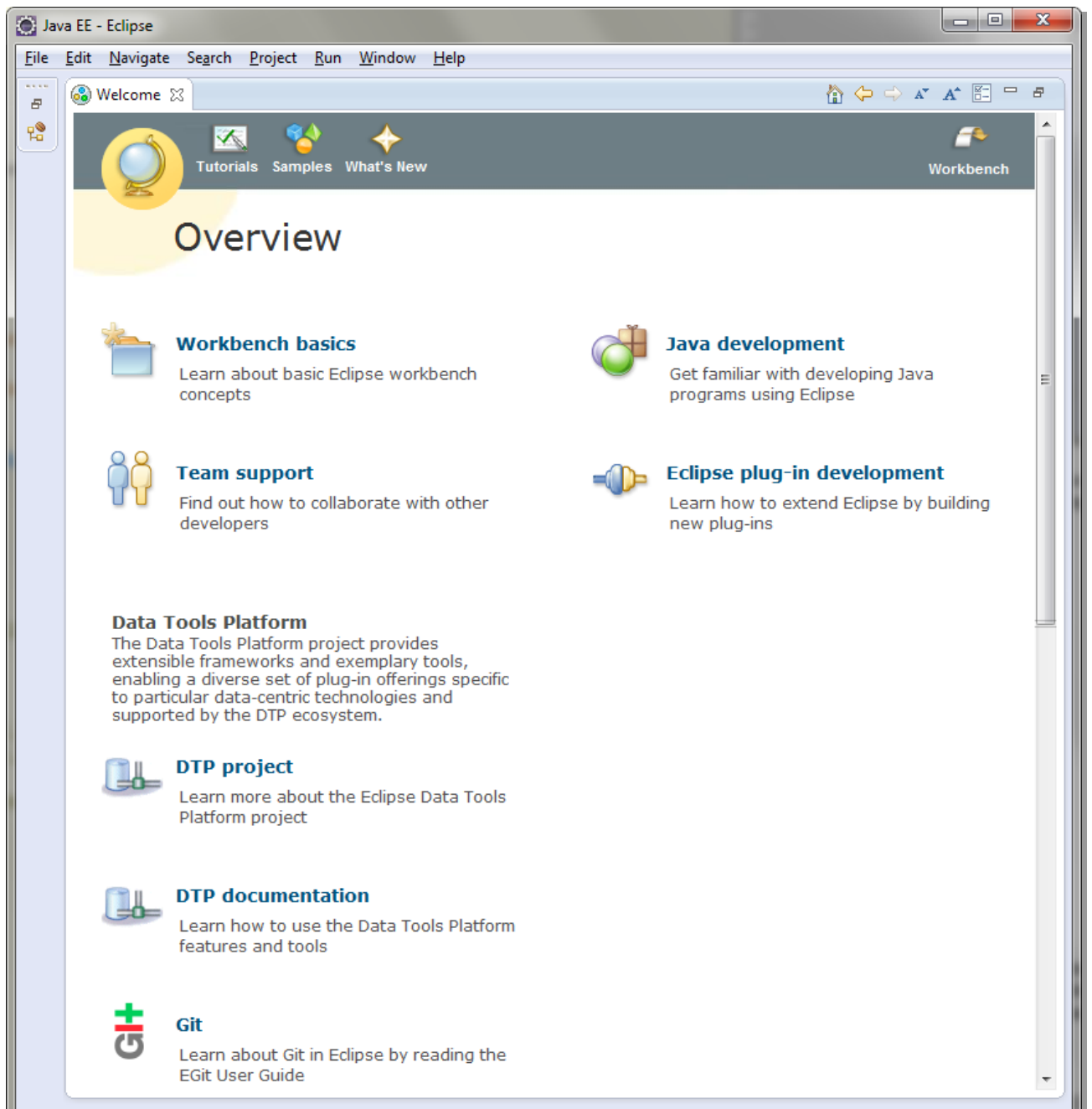
2.1 Робочий простір Eclipse

Робочий простір Eclipse складається з декількох панелей, що називають поданнями (Views), наприклад, навігаційне (Navigator) або схематичне (Outline) подання. Набір таких подань називається перспективою (Perspective). Одна з найпоширеніших перспектив – перспектива "Ресурси" (Resource), яка являє собою базовий набір подань для управління проектами, перегляду й редагування файлів проекту. Обрати перспективу можна за допомогою меню "Window / Open Perspective". Відкрити будь-яке подання в поточній перспективі можна через меню "Window / Show View".

При першому успішному запуску Eclipse буде відображена сторінка вітання (також доступна через меню "Help / Welcome")



Далі рекомендується перейти на сторінку огляду (Overview) де можна ознайомитися з інформацією про нові функції, вивчити деякі приклади або пройти навчальний курс



На сторінці огляду можна відкрити розділ довідкової системи "Основи робочого простору" (Workbench basics), який містить багато корисної початкової інформації про різні компоненти Eclipse і про те, як вони взаємодіють один з одним. Після вивчення цього розділу приступимо до використання інструментів розробки Java Development Tools (JDT) Eclipse.

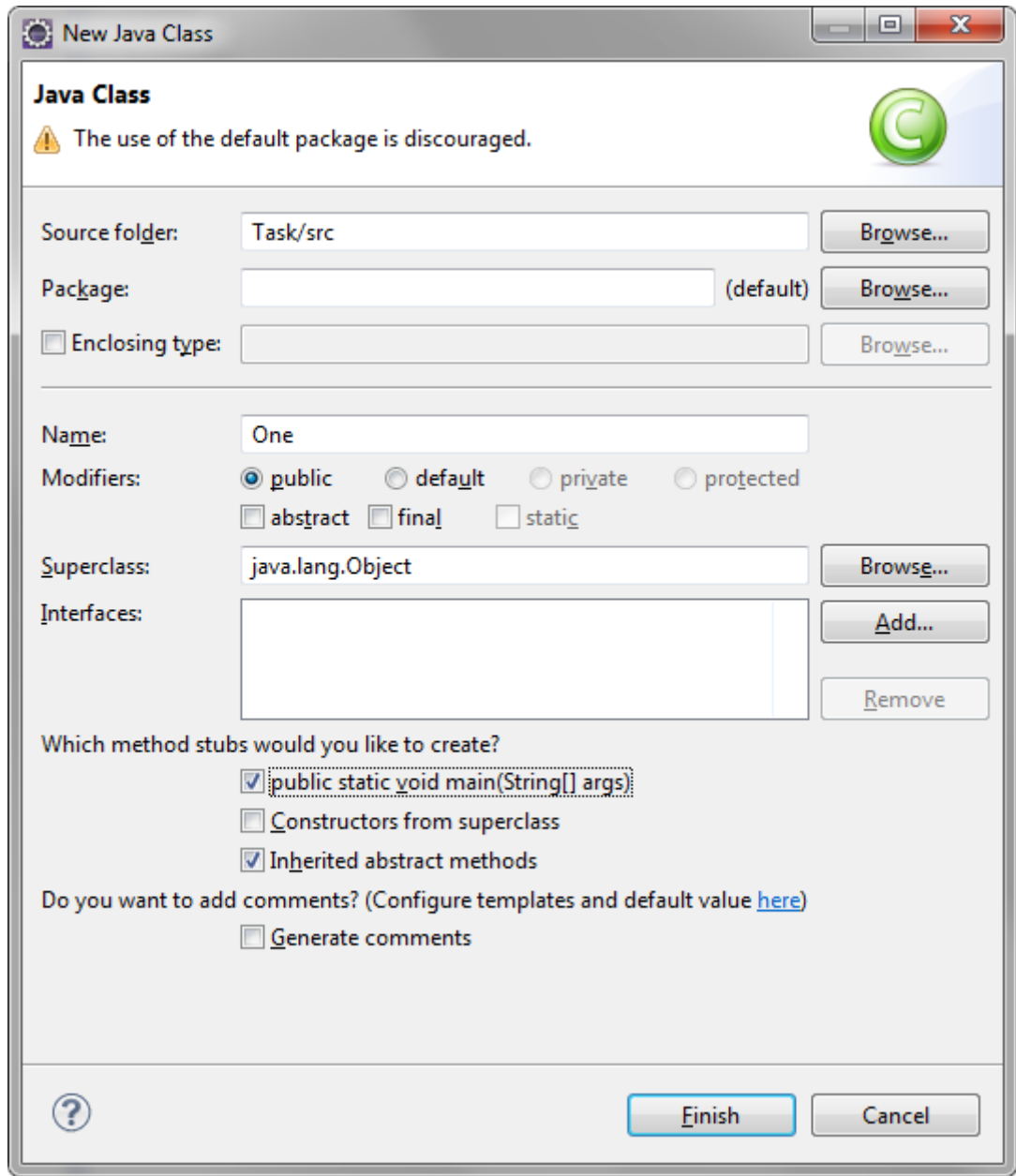
2.2 Створення нового проекту на Java

У меню виберіть "File / New / Java Project" і введіть "Task" у відповідь на запит імені проекту, а потім натисніть "Finish". Відкриється перспектива "Java". Можна або змінити перспективу в поточному вікні, вибравши "Window / Open Perspective / Java", або відкрити нове вікно, вибравши "Window / New Window", і обрати нову перспективу.

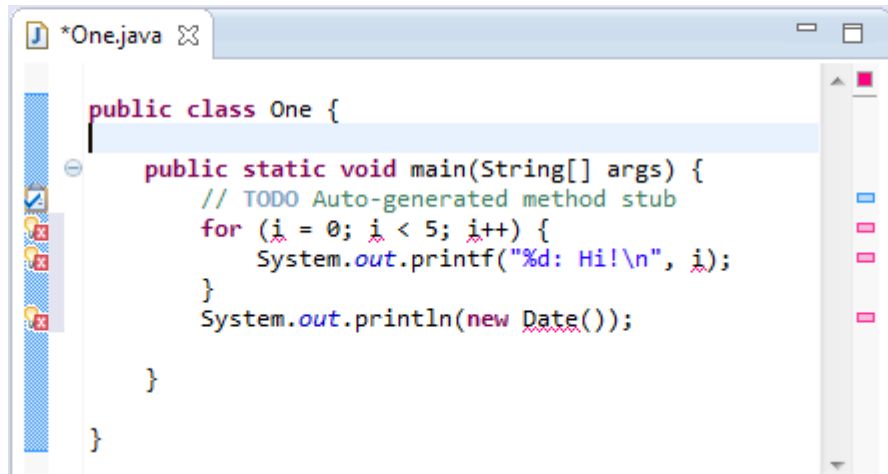
Перспектива "Java" має набір подань, призначених для ведення розробки на Java. Одне з них, розташоване в лівому верхньому куті, відображує ієрархію,

що містить різні пакети Java, класи, Jar-архіви й різноманітні файли. Це подання називається "Package Explorer".

Перебуваючи в перспективі Java, натиснемо правою кнопкою на папці "src" з початковим текстом проекту "Task" і виберемо з меню "New / Class". У діалоговому вікні, що з'явилося, уведемо "One" у якості імені класу. Нижче напису "Which method stubs would you like to create?" (Які заглушки методів ви бажаєте створити?) відзначимо "public static void main(String[] args)" і натиснемо "Finish".



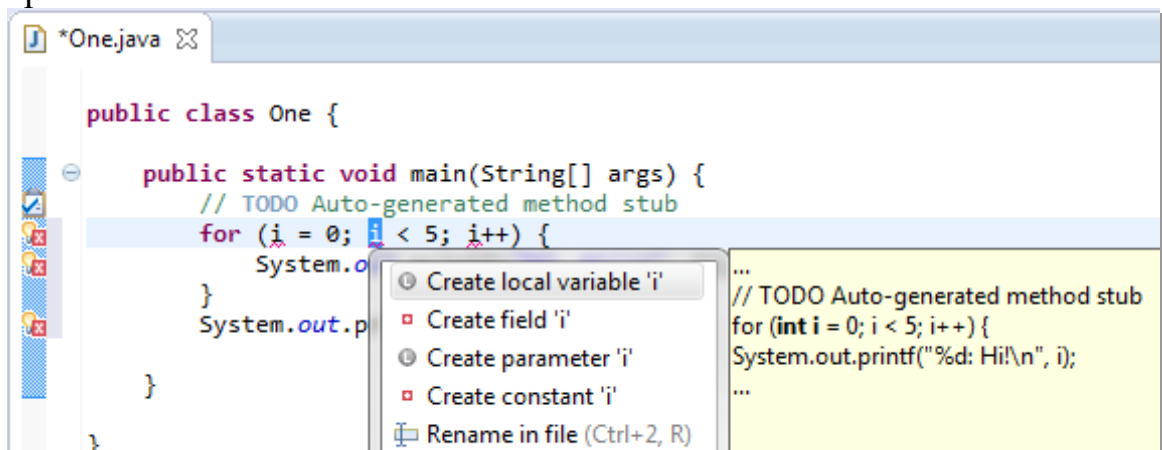
При цьому буде створений файл "One.java" із класом "One" і порожнім методом "main()" в області редактора. Додамо наступний код до методу (зверніть увагу, що опис для змінної "i" був навмисно пропущений):



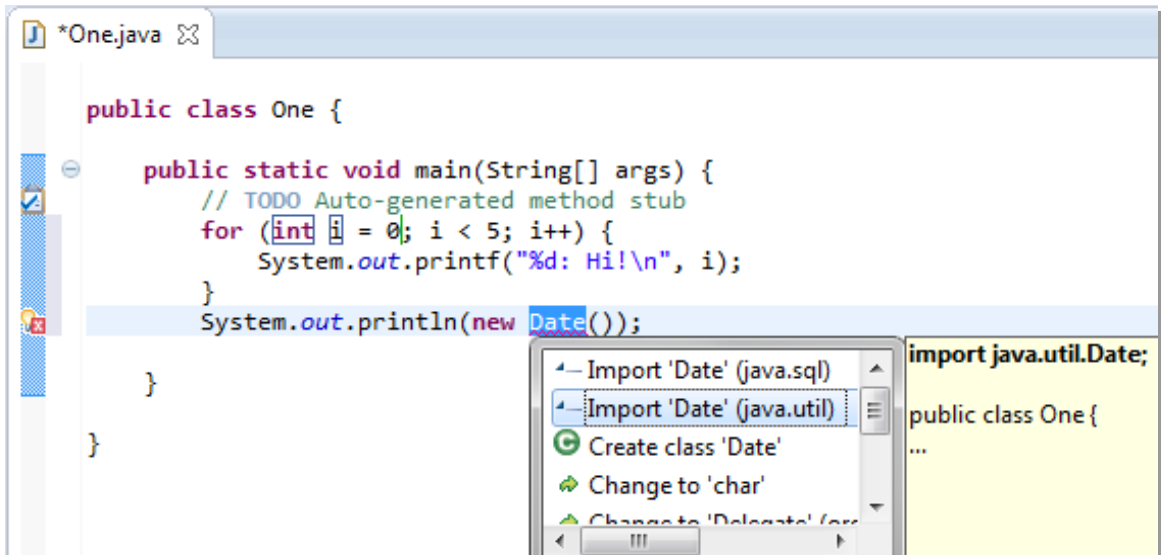
Редактор Eclipse уміє здійснювати перевірку синтаксису й виконувати автоматичне дописування коду. При введенні відкриваючої круглої дужки або подвійних лапок Eclipse автоматично вставляє для них закриваючу пару й поміщає курсор у середину. В інших випадках ви можете викликати автоматичне дописування коду за допомогою комбінації клавіш "Ctrl+I". Функція дописування коду видає контекстно залежний список варіантів, з якого можна здійснювати вибір за допомогою клавіатури або миші. Варіанти можуть являти собою список методів конкретного об'єкту, або фрагмент коду, заснований на різних ключових словах, таких як "for" або "while".

Перевірка синтаксису залежить від процесу інкрементної компіляції. У міру збереження код компілюється у фоновому режимі й перевіряється на наявність синтаксичних помилок. За замовчуванням синтаксичні помилки підкреслюються червоним, а ліворуч на полях з'являється червона позначка. Помилки, позначені на полях символом електричної лампочки, редактор може виправити (функція "Quick Fix").

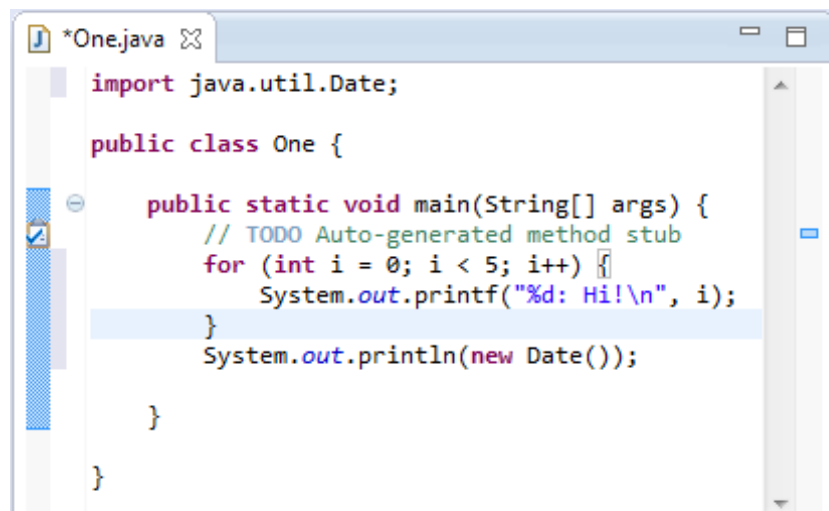
У наведеному коді знак лампочки знаходиться біля оператора for, тому що був пропущений опис для i. Подвійне натискання мишкою на лампочку викликає список пропонуваного виправлення. У нашому випадку буде запропоновано створити поле класу i, локальну змінну i або параметр i для методу; вибір мишкою кожного із цих варіантів покаже той код, який буде згенерований.



Для об'єкта класу Date виправимо помилку шляхом автоматичного додавання рядка "import java.util.Date;" у початок програми.

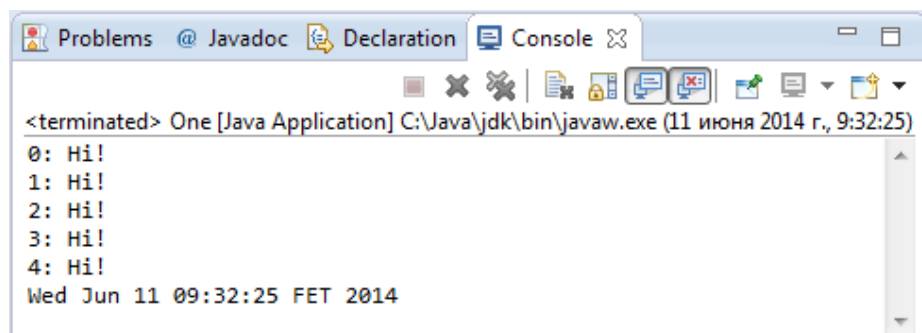


Після всіх виправлень отримаємо код:



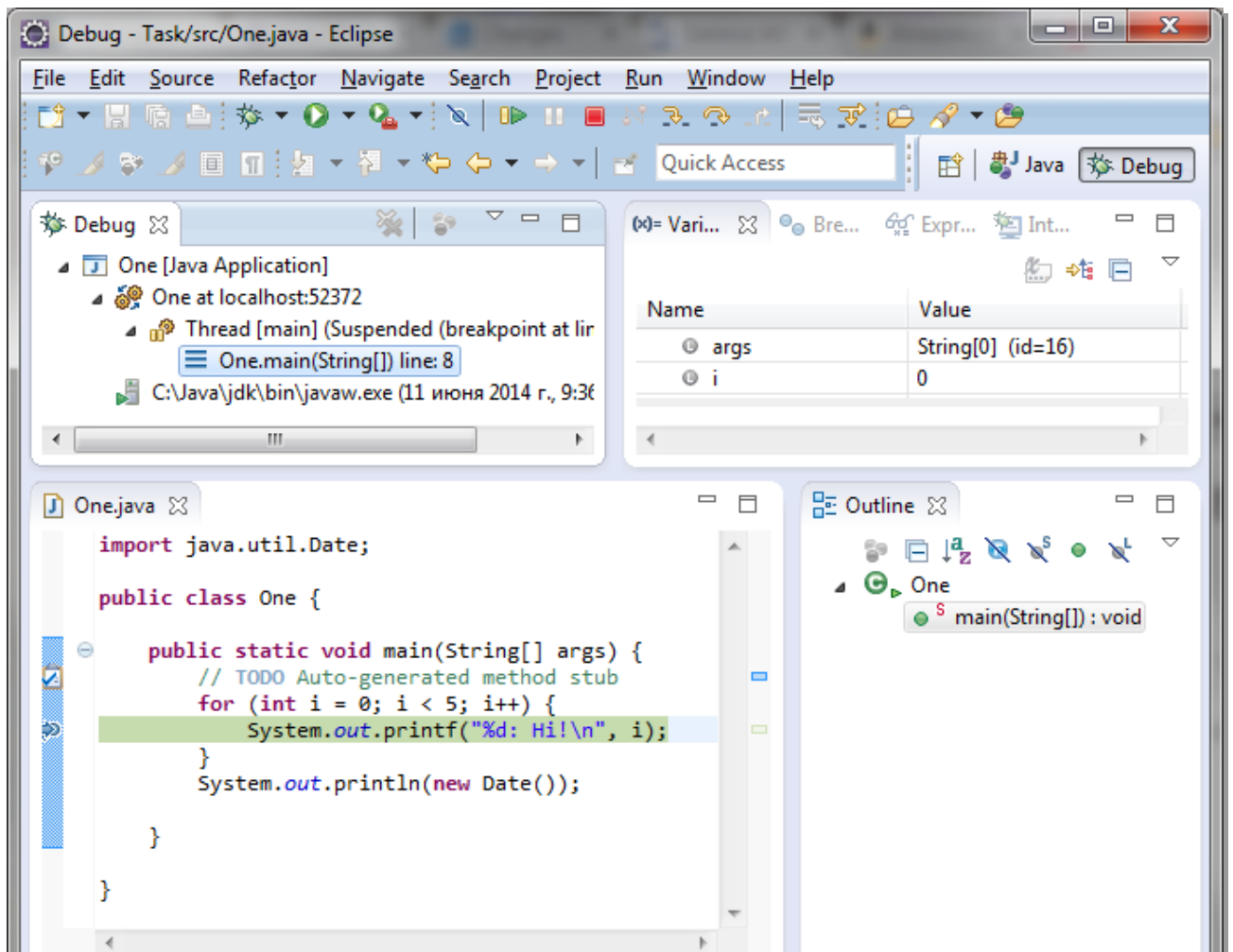
Якщо програма компілюється без помилок, її можна запустити, вибравши Run у меню Eclipse. Зверніть увагу на відсутність окремого кроку для компіляції, тому що компіляція виконується при збереженні коду. Якщо в коді відсутні синтаксичні помилки, він готовий для запуску.

У нижній панелі з'являється нова панель із закладкою "Console" (Консоль), що відображає результат роботи програми.



Можна також запустити програму у відлагоджувальнику Java. Спочатку потрібно встановити контрольну точку в main() за допомогою подвійного клацання мишки на сірому полі з лівої сторони вікна редагування поруч із

викликом "System.out.printf(...)". У меню "Run" виберіть команду "Debug" (Налагодження). Відбудеться активація перспективи налагодження "Debug", яка містить цілий ряд нових подань:



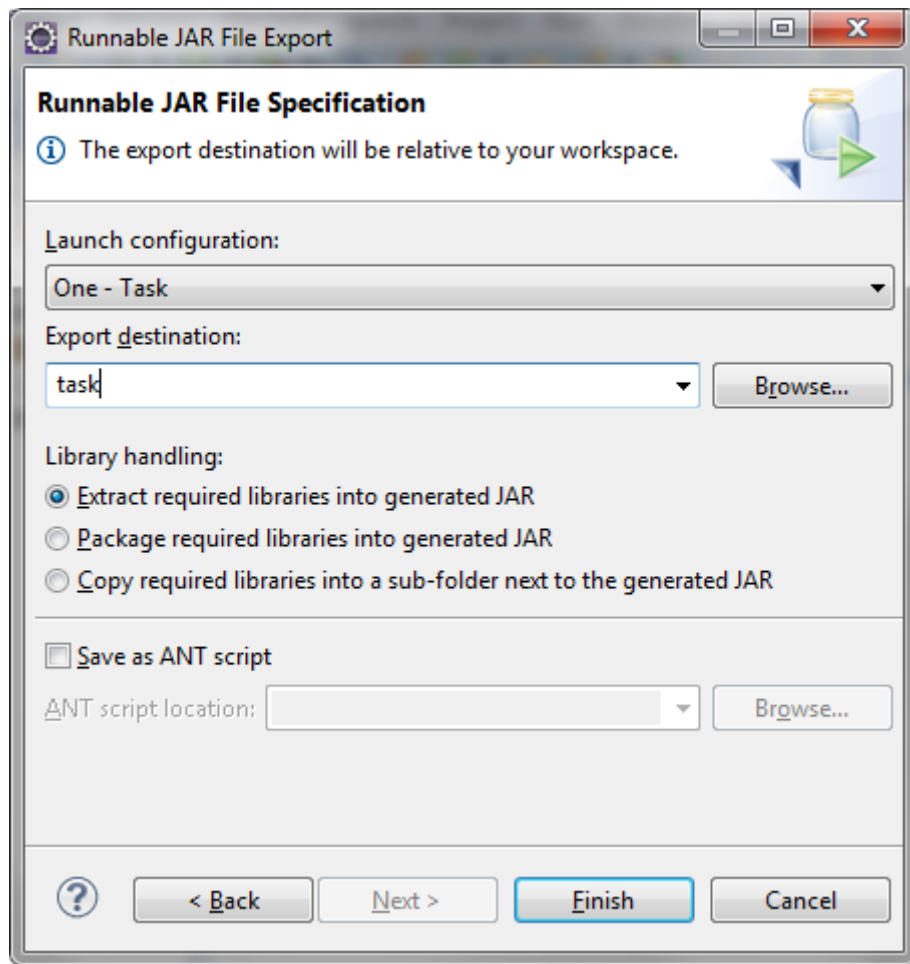
Зверніть увагу на подання "Debug" у лівому верхньому куті перспективи. Це подання показує стек викликів і містить панель інструментів у рядку заголовка, що дозволяє управляти виконанням програми. На панелі є кнопки для продовження (Resume), припинення (Suspend) або завершення програми (Terminate), перехід до наступного оператора (Step Into), переступання наступного оператора (Step Over) або повернення з методу (Step Return).

Панель нагорі праворуч містить подання "Variables" (Змінні), "Breakpoints" (Контрольні точки). З меню "Window" можна відкрити подання "Expressions" (Вирази), "Display" (Відображення) і т.д.

Якщо активувати подання із закладкою "Variables", можна побачити поточне значення змінної `i`.

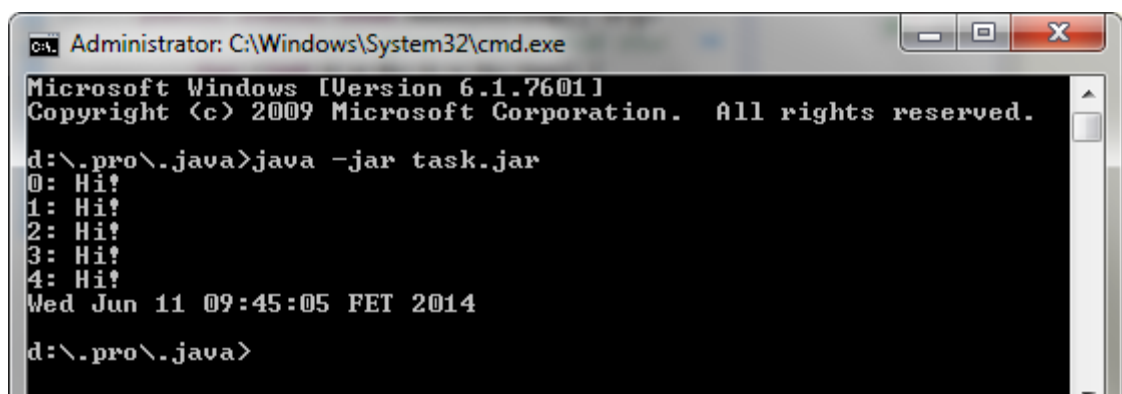
Більш докладну інформацію про кожне подання можна одержати за допомогою контекстної підказки; для цього клацніть мишкою на заголовку подання й натисніть клавішу "F1".

Для одержання JAR файлу, що виконується, можна використовувати команду меню "File / Export / Java / Runnable JAR file", кнопка "Next", "Finish".



Виконати отриманий JAR файл можна з командного рядка в каталозі, де був створений файл task.jar за допомогою команди:

```
java -jar task.jar
```



2.3 Архітектура платформи Eclipse

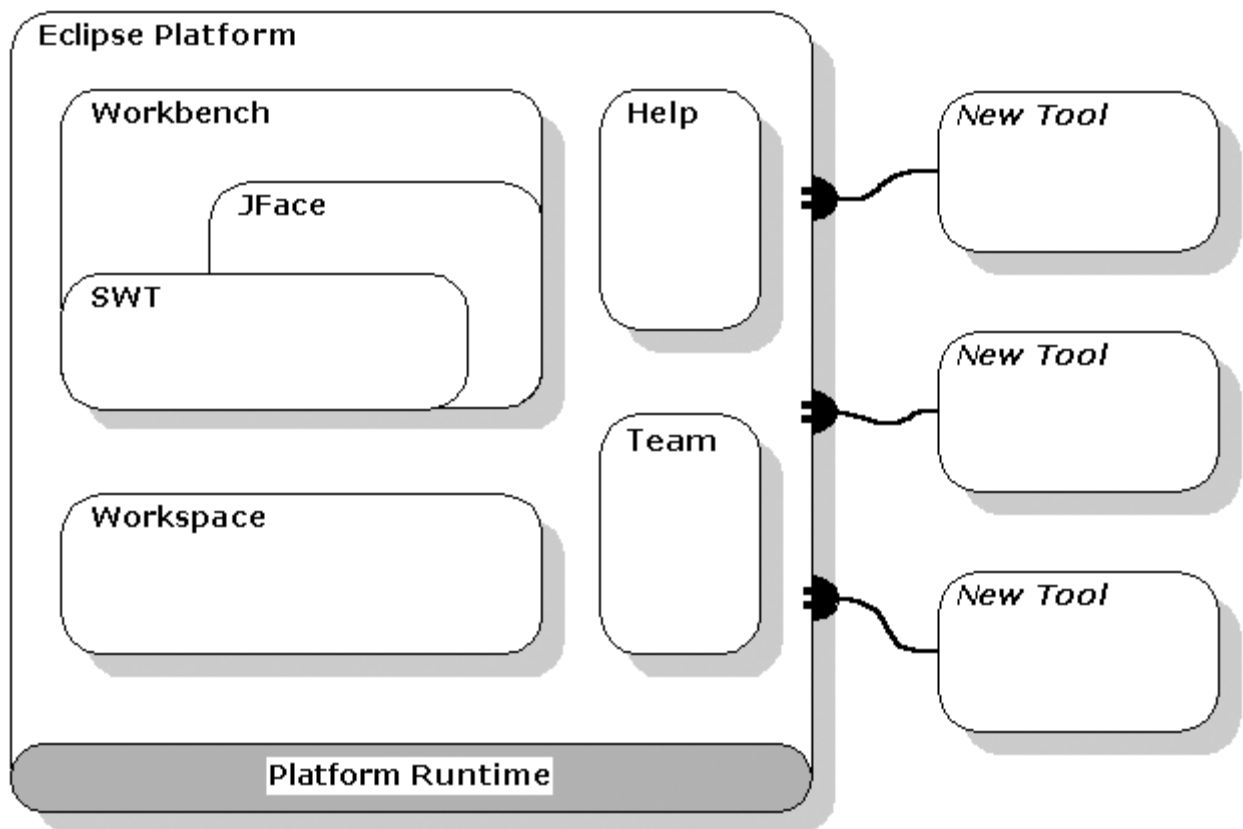
На додаток до плагінів типу JDT, призначеним для редагування, компіляції й налагодження додатків, є плагіни, що підтримують увесь процес розробки: моделювання, автоматизацію побудови, тестування модулів, тестування продуктивності, керування версіями й конфігурацією.

Eclipse містить плагін EGit для роботи із системою управління версіями Git. Плагін з'єднується з Git-сховищем, дозволяючи членам команди

розроблювачів працювати з набором файлів, що містять початкові тексти, не втручаючись у зміни, внесені іншими.

Плагіни, які підтримуються й поширюються співтовариством Eclipse Foundation можна знайти на сайті Eclipse. Найбільш повний список наявних плагінів доступний на сайті Eclipse Marketplace (в Eclipse меню Help/Eclipse Marketplace...).

Платформа Eclipse містить потужний набір плагінів, які підтримують різні види проектів.



Деякі компоненти платформи:

- Runtime. Код, який визначає модель плагінів Eclipse, засновану на специфікації OSGi (Open Services Gateway Initiative – специфікація динамічної модульної шини для створення Java-програм, що розробляється консорціумом OSGi Alliance). Суть полягає в можливості переінсталиювати динамічно компоненти і складові частини програми без необхідності зупиняти і перезапускати її. Runtime також надає додаткові сервіси, такі як ведення системного журналу й паралелізм.
- Workbench – робочий простір. Робочий простір надає Eclipse його індивідуальність. Саме на цьому рівні реалізована концепція подань, перспектив і таких елементів як вікна редагування.
- JFace/SWT. Пакет інструментів Standard Widget Toolkit (SWT) являє собою набір віджетів, відповідальних за користувацький інтерфейс і функції Eclipse. JFace – це просто надбудова над SWT, що надає кілька класів Model-View-Controller (MVC) для полегшення розробки графічних додатків.

- Help – підтримка користувачів. Реалізується через довідкову систему, яка дозволить користувачам шукати довідкову документацію, або за допомогою "шпаргалок", які для кінцевих користувачів можуть виглядати як інтерактивні списки задач.
- Team – команда. Компонент Team – це інфраструктура, що дозволяє фірмам-розроблювачам підключати свої власні системи управління версіями. Зразком реалізації провайдера є плагін EGit, вбудований в Eclipse.

3 Додаток С. Стиль кодування

3.1 Рекомендації з використання імен

За загальноприйнятою згодою ім'я класу пишеться з великої літери. Кілька слів пишуться разом, кожне починається з великої літери.

При написанні ідентифікаторів методів і полів, імен посилань на об'єкти використовують той же стиль, однак перша буква записується в нижньому регістрі.

Примітиви `final static` з початковими незмінними значеннями (константи часу компілювання) йменуються великими літерами та слова розділяються підкресленням (на зразок констант мови C, що вводяться директивою `#define`)

```
class VeryLongClassName {
    private static final int DEFAULT_NUM = 10;
    private final ArrayList<Item2d> items = new ArrayList<Item2d>();
    // ...
    public void showBody() {
        for(Item2d item : items) {
            System.out.printf("(%.0f; %.3f) ", item.getX(), item.getY());
        }
        System.out.println();
    }
}
```

Java код у бібліотеках від Oracle відповідає правилу розміщення відкриваючих і закриваючих фігурних дужок, як показано вище. Більш детальніше інформація наведена в документі *Java Code Conventions*, який доступний в електронному вигляді.

3.2 Документування програмного коду

Найбільша проблема, пов'язана з документуванням кода – підтримка цієї документації. Якщо документація й код розділені, виникають труднощі, пов'язані з необхідністю внесення змін у відповідні розділи супровідної документації щораз при зміні програмного коду. Середовище розробки пропонує рішення – зв'язати код з документацією, помістивши все в один файл.

Існує спеціальний синтаксис для оформлення документації у вигляді коментарів і інструмент для виділення цих коментарів у зручну форму. Інструмент називається `javadoc`. Обробляючи файл із вихідним текстом програми, він виділяє позначену документацію з коментарів і зв'язує з іменами відповідних класів або методів. Таким чином, затративши мінімум зусиль на оформлення коментарів, можна одержати документацію до програми.

На виході `javadoc` утворюється HTML файл, який можна переглянути будь-яким веб-оглядачем. Цей інструмент дозволяє створювати й підтримувати файли з початковим текстом програми й, при необхідності, генерувати супровідну документацію. Бібліотеки Java звичайно документуються саме таким способом, саме тому при розробці програм зручно використовувати JDK

з коментованим для javadoc початковим текстом бібліотек замість JRE, де початкові тексти відсутні.

3.2.1 Коментарі документації

Java підтримує три типи коментарів. Перші два типи: `//...` і `/*...*/`. Третій тип називається коментарем документації. Такий коментар починається з послідовності символів `/**` і закінчується послідовністю `*/`. Коментарі документації дозволяють додавати в програму інформацію про неї самої. За допомогою утиліти javadoc (що входить до складу JDK) цю інформацію можна витягати й поміщати в HTML файл.

Утиліта javadoc дозволяє вставляти HTML теги й використовувати спеціальні ярлики (дескриптори) документування. HTML теги заголовків не використовують, щоб не порушувати стиль файлу, сформованого утилітою.

Дескриптори javadoc, що починаються зі знака `@`, називаються автономними й повинні розміщуватися з початку рядка коментаря (лідуючий символ `*` ігнорується). Дескриптори, що починаються з фігурної дужки, наприклад `{@code}`, називаються вбудованими й можуть застосовуватися усередині опису.

Коментарі документації застосовують для документування класів, інтерфейсів, полів (змінних), конструкторів і методів. У кожному випадку коментар повинен перебувати перед елементом, що документується.

Для документування змінної можна використовувати наступні дескриптори: `@see`, `@serial`, `@serialfield`, `{@value}`, `@deprecated`.

Для класів і інтерфейсів можна використовувати дескриптори: `@see`, `@author`, `@deprecated`, `@param`, `@version`.

Методи можна документувати за допомогою дескрипторів: `@see`, `@return`, `@param`, `@deprecated`, `@throws`, `@serialdata`, `{@inheritdoc}`, `@exception`.

Дескриптори `{@link}`, `{@docroot}`, `{@code}`, `{@literal}`, `@since`, `{@linkplain}` можуть застосовуватися де завгодно.

3.2.2 Загальна форма коментарів

Після початкової комбінації символів `/**` розташовується текст, що є головним описом класу, змінної або методу. Далі можна вставляти різні дескриптори. Кожний дескриптор `@` повинен стояти першим у рядку. Кілька дескрипторів того самого типу необхідно групувати разом. Вбудовані дескриптори (починаються з фігурної дужки) можна поміщати усередині будь-якого опису.

Утиліта javadoc у якості вхідних даних приймає файл із початковим кодом програми. Генерує декілька HTML файлів, що містять документацію по цій програмі. Інформація про кожний клас буде розміщуватися в окремому HTML файлі. Крім того, створюється дерево індексів і ієрархії. Можуть бути згенеровані й інші HTML файли.

Item2d.java

```
1 package ex01;
2
3 import java.io.Serializable;
4
5 /**
6  * Хранит исходные данные и результат вычислений
7  * @author xone
8  * @version 1.0
9  */
10 public class Item2d implements Serializable {
11
12     /** Аргумент вычисляемой функции */
13     private double x;
14
15     /** Результат вычисления функции */
16     private double y;
17
18     /** Автоматически сгенерированная константа */
19     private static final long serialVersionUID = 1L;
20
21     /**
22      * Инициализирует поля {@linkplain Item2d#x}, {@linkplain Item2d#y}
23      */
24     public Item2d() {}
25
26
27
28     /**
29      * Устанавливает значения полей: аргумента
30      * и результата вычисления функции
31      * @param x значение для инициализации поля {@linkplain Item2d#x}
32      * @param y значение для инициализации поля {@linkplain Item2d#y}
33      */
34
35     public Item2d(double x, double y) {}
```

Problems

@ Javadoc

Declaration

ex01.Item2d

Хранит исходные данные и результат вычислений

Version:
1.0

Author:
xone

У середовищі Eclipse генерувати документацію можна використовуючи команду меню "Project/Generate Javadoc...".

ex01

Class Item2d

java.lang.Object
ex01.Item2d

All Implemented Interfaces:

java.io.Serializable

```
public class Item2d
extends java.lang.Object
implements java.io.Serializable
```

Хранит исходные данные и результат вычислений

Version:

1.0

Author:

xone

See Also:

Serialized Form

Field Summary

Fields	
Modifier and Type	Field and Description
private static long	<code>serialVersionUID</code> Автоматически сгенерированная константа
private double	<code>x</code> Аргумент вычисляемой функции
private double	<code>y</code> Результат вычисления функции

3.2.3 Дескриптори javadoc

@author опис	Документує автора класу. При виклику утиліти javadoc потрібно задати опцію -author, щоб включити поле в HTML документацію.
{@code фрагмент_коду}	Дозволяє вбудовувати в коментар текст (фрагмент коду). Цей текст буде відображатися за допомогою шрифту коду без наступної обробки в HTML.
@deprecated опис	Визначає, що клас, інтерфейс або член класу є застарілим. Рекомендується включати дескриптори @see або {@link} для того, щоб інформувати програміста про доступні альтернативні варіанти. Може використовуватися для документування змінних, методів і класів.
{@docroot}	Визначає шлях до кореневого каталогу поточної документації.
@exception ім'я_виключення пояснення	Описує виключення для даного методу. Тут ім'я_виключення вказує повне ім'я виключення, а пояснення – це рядок, що описує у яких випадках може виникнути дане виключення. Може використовуватися тільки для документування методів.
{@inheritdoc}	Успадковує коментар від безпосереднього суперкласу.
{@link пакет.клас#елемент текст}	Вбудовує посилання на додаткову інформацію. При відображенні текст (якщо є присутнім) використовується в якості імені посилання.
{@linkplain пакет.клас#елемент текст}	Вбудовує посилання. Посилання відображається шрифтом основного тексту.
{@literal опис}	Дозволяє вбудовувати текст у коментар. Цей текст відображається "як є" без наступної обробки HTML.
@param ім'я_параметра пояснення	Документує параметр для методу або параметр-тип для класу або інтерфейсу. Може використовуватися тільки для документування метода, конструктора, узагальненого класу або інтерфейсу.
@return пояснення	Описує значення, що вертається, методу.
@see посилання @see пакет.клас#елемент текст	Забезпечує посилання на додаткову інформацію.
@serial опис	Визначає коментар для поля, що серіалізується за замовчуванням.
@serialData опис	Документує дані, записані за допомогою методів writeObject і writeExternal.
@serialField ім'я тип опис	Для класу, що реалізує Serializable, дескриптор забезпечує коментарі для компонента ObjectOutputStreamField. Тут ім'я становить ім'я поля, тип становить його тип, а опис – коментар для даного поля.
@since випуск	Показує, що клас або елемент класу був уперше представлені в певному випуску. Тут випуск становить рядок, у якому зазначений випуск або версія, починаючи з якої ця особливість стала доступною.
@throws ім'я_виключення пояснення	Має те ж призначення, що й дескриптор @exception.
{@value}	Відображає значення наступної за ним константи, якою повинне бути поле static.
{@value пакет.клас#поле}	Відображає значення певного поля static.
@version інформація	Представляє інформацію про версію (як правило, номер). При виконанні утиліти javadoc потрібно вказати опцію -version, щоб цей дескриптор включити в HTML документацію.

3.3 Структура проекту

Для лабораторних робіт слід використовувати назву проектів (projects) у вигляді "surnameXX", де:

- "surname" – прізвище розробника латиницею в нижньому регістрі (<http://translit.kh.ua/?lat&passport>);
- "XX" – номер роботи.

Кореневий пакет повинен мати вигляд:

```
ua.khpi.project
```

де "project" – назва проекту.

Початковий код всіх класів слід розташовувати починаючи з цього кореневого пакета.

Приклад структури проекту в Eclipse:

