ЗАТВЕРДЖУЮ
Начальник кафедри інформаційних технологій та телекомунікаційних систем, к.т.н., доцент
Олександр ПРИДАТКО
«_ _ »_____20___ р.

МЕТОДИЧНА РОЗРОБКА

для проведення практичного заняття з студентами (курсантами) 1-го курсу спеціальності 122 «Комп'ютерні науки» з дисципліни «Основи програмування»

Практичне заняття 5.4: «Перший за стосунок в IDE Eclipse/ IntelliJ IDEA»

Мета: здобути навики щодо створення нового Java-проекту та написання найпростішої програми із виводу текстової стрічки у консоль. Закріпити теоретичні знання щодо структури програмного коду на Java.

Час: 2 годин

Місце заняття: комп'ютерна лабораторія

Матеріальне забезпечення:
- ПК з відповідним ПЗ та доступом до мережі Internet;
- Методичні рекомендації щодо виконання практичної роботи.

Після практичного заняття студенти (курсанти) повинні:
вміти: реалізовувати найпростіші програми із виводу текстових стрічок у консоль середовища розробки Eclipse.
знати: порядок створення нового Java-проекту та його структури, а також структури програмного коду написаного мовою Java.

Література:
1. Віртуальне навчальне середовище ЛДУ БЖД «Віртуальний університет» [Електронний ресурс]. – Доступний з : http://virt.ldubgd.edu.ua/course/view.php?id=1528
2. Head First Java (изучаем Java): пер. с англ. / Kathy Sierra, Bert Bates. – Москва : «Эксмо», 2012. – 718 с.

План заняття:
1. Перший застосунок в середовищі розробки
2. Структура коду програми
3. Видача індивідуальних практичних завдань

2.1. Перший застосунок в середовищі розробки

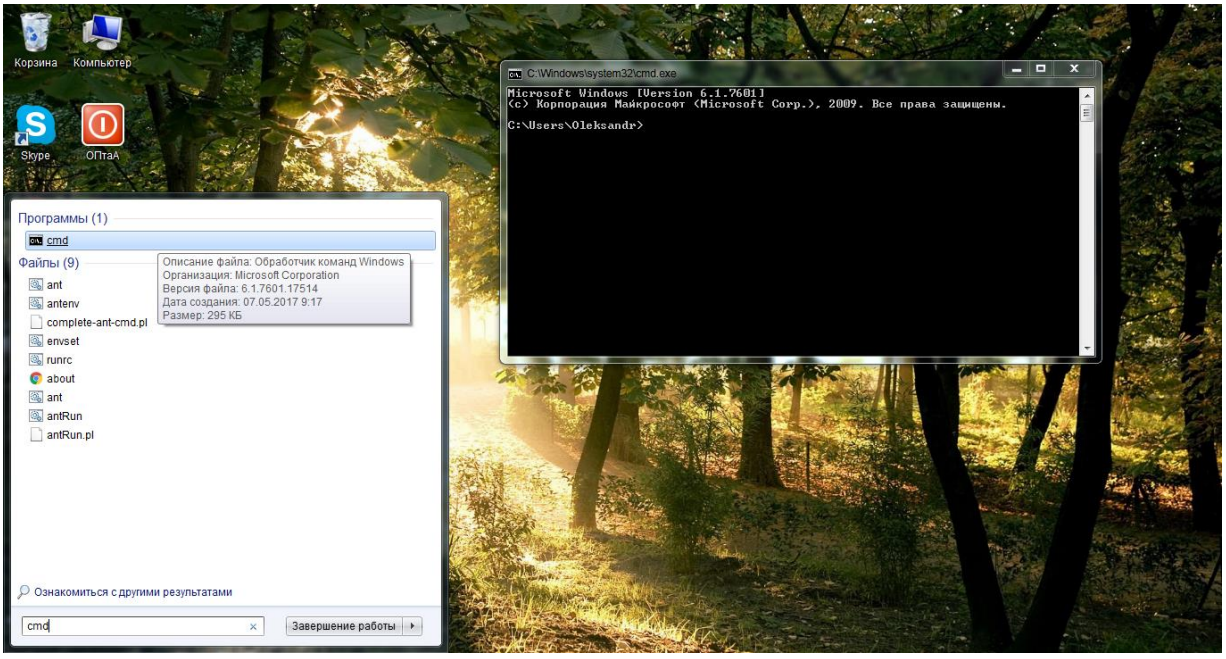
В якості першого застосунку реалізуємо найпростішу програму із виводу на екран стрічки «Hello world!». Спершу реалізуємо застосунок без використання IDE (з метою повноти уяви про трудомісткість процесу програмування без IDE).

Для написання коду програми запускаємо звичайний Notepad (як згадувалось в лекції) та набираємо код програми (про логіку коду пізніше):

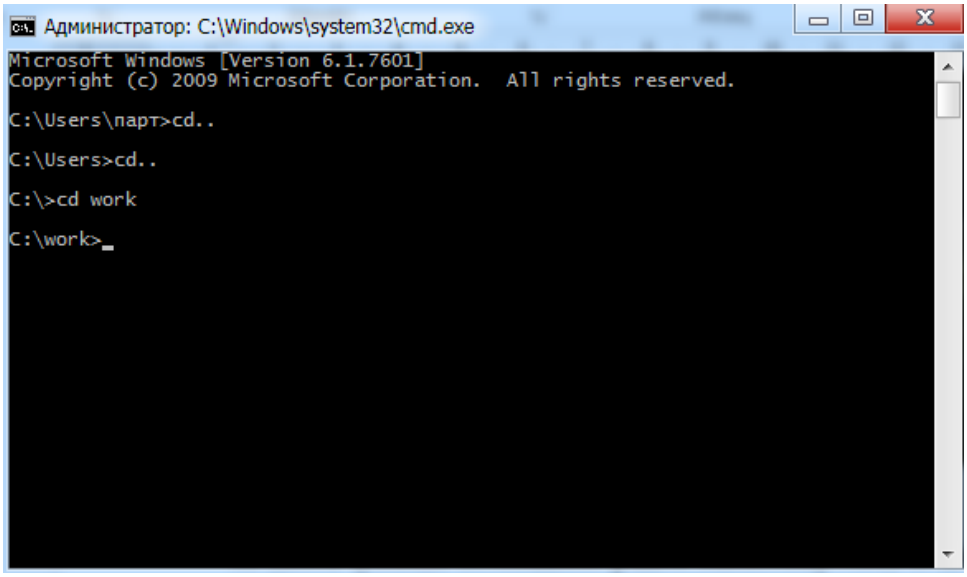
```
public class HelloWorld {
     public static void main(String[] args) {
          System.out.println("Hello world");
     }
}
```

Далі необхідно зберегти цей файл з розширенням .java та назвою, яка відповідає назві класу – HelloWorld. Місце збереження файлу не має значення, проте з метою полегшення процесу доступу до файлу, рекомендовано обрати місце збереження вище до кореневого каталогу. В нашому прикладі створено папку «work» в кореневому каталозі C:\.

Наступним кроком у виконанні програми є запуск командного рядка. Для виконання цієї операції необхідно у вікні пошуку програмного меню «Пуск» ввести «cmd».



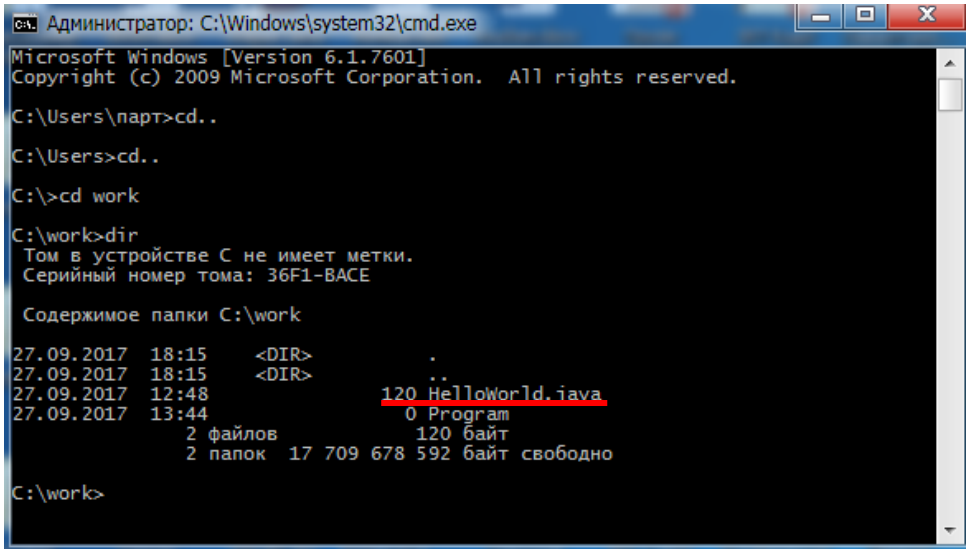
За замовчування командний рядок завантажується із доступом до папки користувача. З метою зміни адреси доступу (переходу до підкаталогу *вищого* рівня) необхідно виконати команду «cd..» потрібну кількість разів. Для переходу до підкаталогу *нижчого* рівня необхідно застосувати команду «cd» після якої вказувати ім'я наступного за ієрархією підкаталогу. В нашому прикладі потрібно здійснити перехід до підкаталогу «work» кореневого каталогу C:\, як це зображено на прикладі:



В папці «work» попередньо збережено файл з кодом програми HelloWorld.java. Переконатись в цьому можливо з допомогою командного рядка. З цією метою застосовують команду «dir»:

```
C:\work>dir
```

В результаті виконання команди буде виведено інформацію про відповідний підкаталог та розміщені у ньому файли:



Як видно з представленого прикладу в зазначеній папці міститься файл HelloWorld.java. Наступним кроком у виконанні програми є компіляція файлу з кодом програми.

Для того, щоб зв’язати зазначений файл з компілятором java, необхідно вказати шлях доступу до javac. Місце розміщення компілятора відповідає директорії встановлення JDK, в зазначеному прикладі це:

```
C:\work>c:\Java\jdk1.8.0_144\bin\javac
```

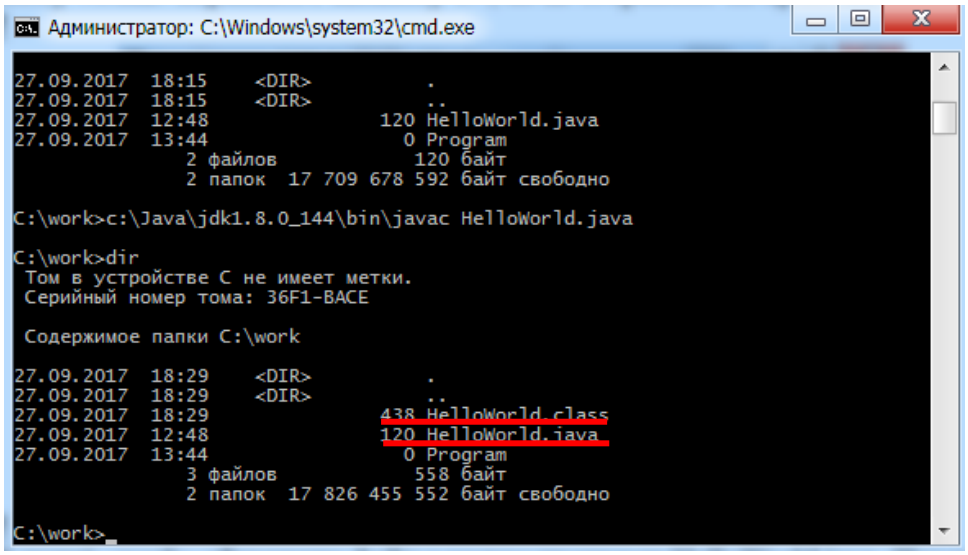
```
або: C:\work> “c:\Program Files\Java\jdk1.8.0_144\bin\javac”
```

Для виконання компіляції, після зазначення адреси розміщення javac, необхідно вказати ім’я файлу з розширенням за прикладом:

```
C:\work>c:\Java\jdk1.8.0_144\bin\javac HelloWorld.java
```

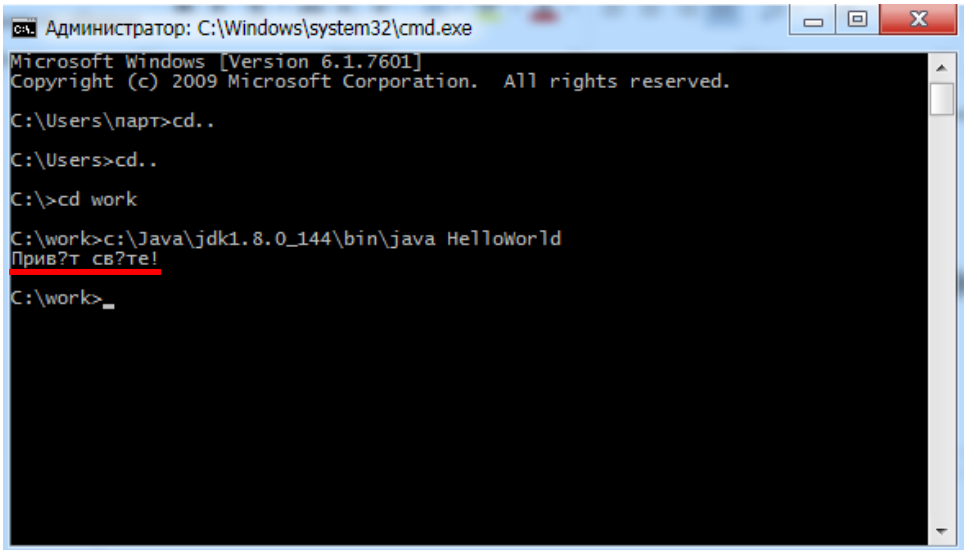
або: `C:\work> “c:\Program Files\Java\jdk1.8.0_144\bin\javac” HelloWorld.java` (ОДНА команда!)

Після виконання компіляції, в командному рядку не відбудеться жодних змін, проте за адресою підкаталогу з вихідним кодом програми з’явиться ще один файл з ідентичною назвою та іншим розширенням (.class) – це файл з відкомпільованим байт-кодом програми. Переконатись в цьому можливо шляхом повторного застосування команди «dir»:



З метою виводу в командний рядок результатів компіляції необхідно виконати дії із зазначенням місця розташування JVM, яка виконуватиме відповідний байт-код:

```
C:\work>c:\Java\jdk1.8.0_144\bin\java HelloWorld
```



Результатом роботи програми буде вивід стрічки без відтворення деяких символів (або взагалі набір незрозумілих символів). Це результат виконання невідповідної кодової сторінки, для заміни якої необхідно повернутись до каталогу користувача, в нашому випадку це `C:\Users\парт` та використати команду:

```
C:\Users\парт>chcp 1251
```

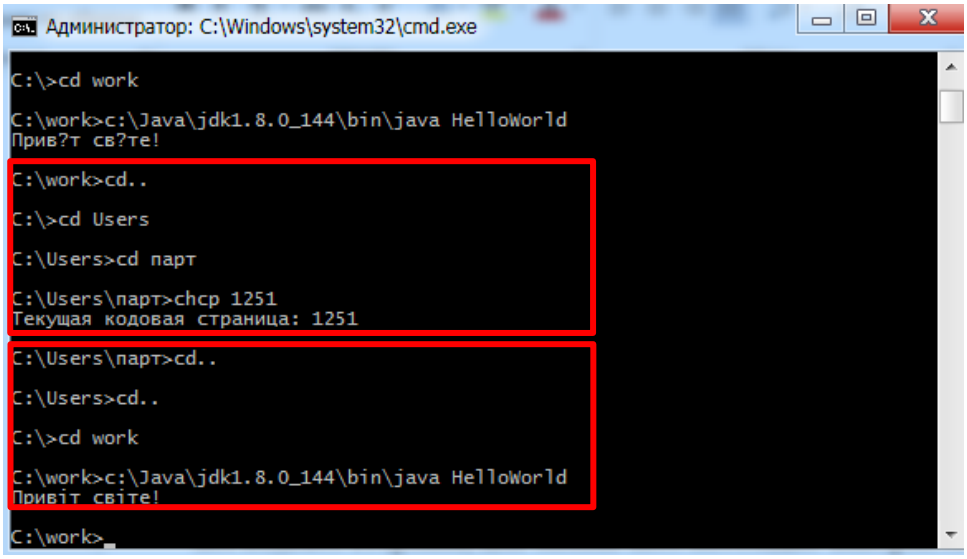
Або ж реалізувати виконання цієї команди перебуваючи безпосередньо в каталозі «work»:

```
C:\work>chcp 1251
```

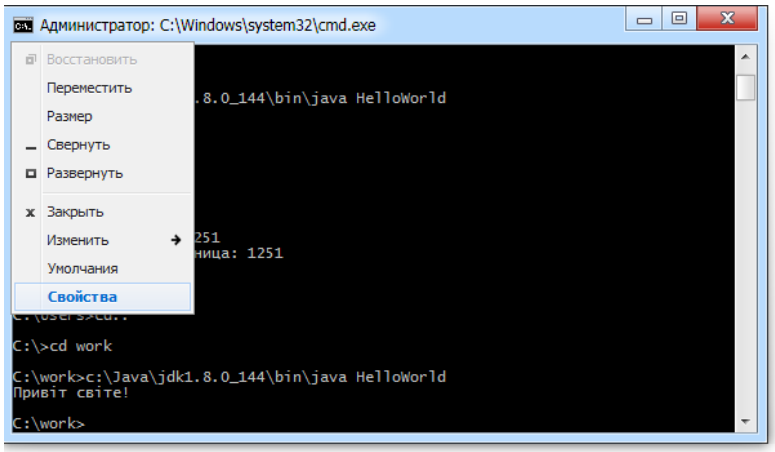
Після чого повернутись повторно виконати програму:

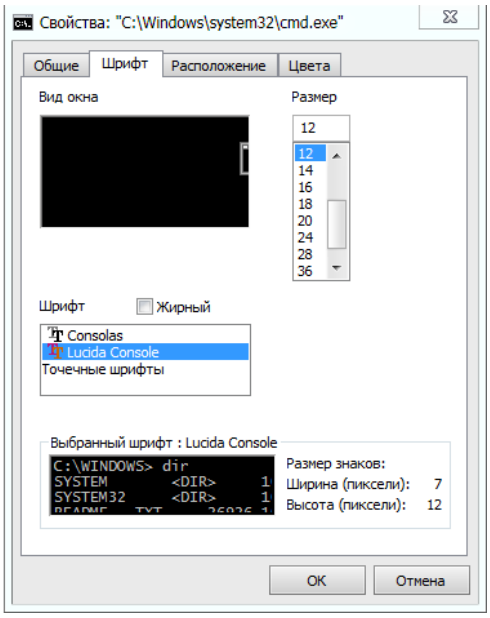
```
C:\work>c:\Java\jdk1.8.0_144\bin\java HelloWorld
```

Результатом роботи програми буде вивід стрічки «Привіт світе!»



Якщо в результаті роботи програми все ж таки одержана стрічка з незрозумілих символів, потрібно переконатись у відповідності шрифтів виводу тексту в консоль. Для цього необхідно перейти в меню командного рядка та обрати категорію «властивості». За обраною категорією в закладці «шрифт» необхідно обрати шрифт Lucida Console або Consolas.

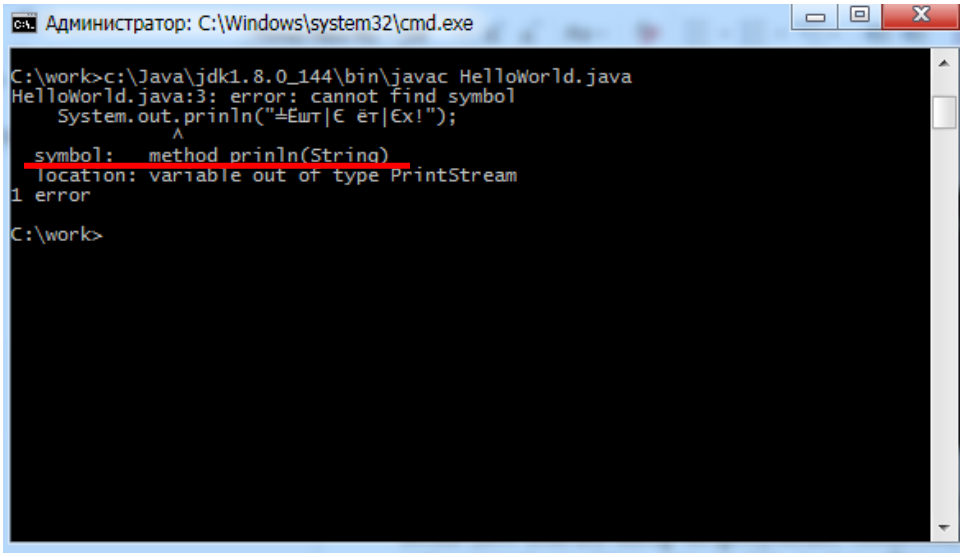




Слід зауважити, що в разі допущення в первинному коді програми будь-якої помилки, текстовий редактор цього не повідомить. Відстежити посилку можливо лише після компіляції програми з допомогою командного рядка. До прикладу, допустимо помилку в методі `println` :

```
public class HelloWorld {
     public static void main (String [] args){
          System.out.prinln("Привіт світе!");
     }
}
```

В результаті компіляції буде отримано:

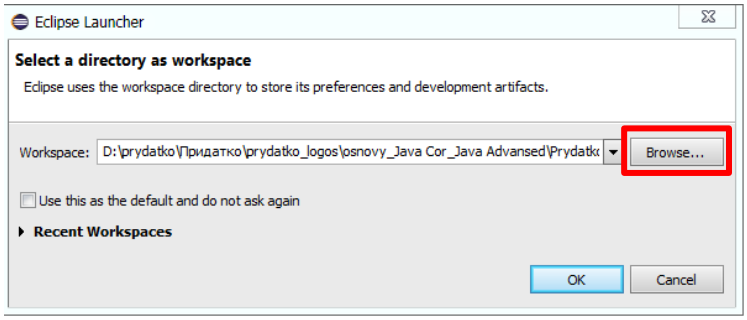


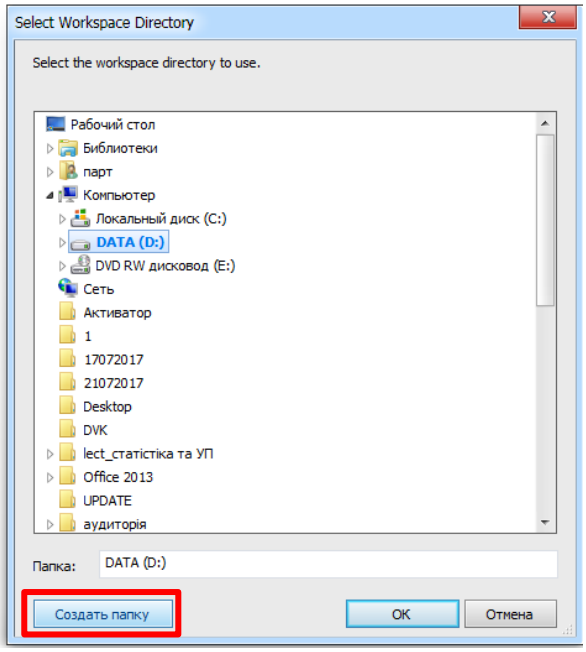
Для виправлення помилки необхідно повернутись до текстового редактора, виправити помилку та повторно запустити програму на компіляцію з допомогою командного рядка.

Власне незручність при роботі з помилками та трудомісткість процесу компіляції є основними недоліками такого методу програмування. З метою полегшення цього процесу використовують інтегровані середовища розробки, які орієнтовані на автоматичну перевірку синтаксису коду під час його написання та швидку компіляцію програми із виводом результату в консоль.

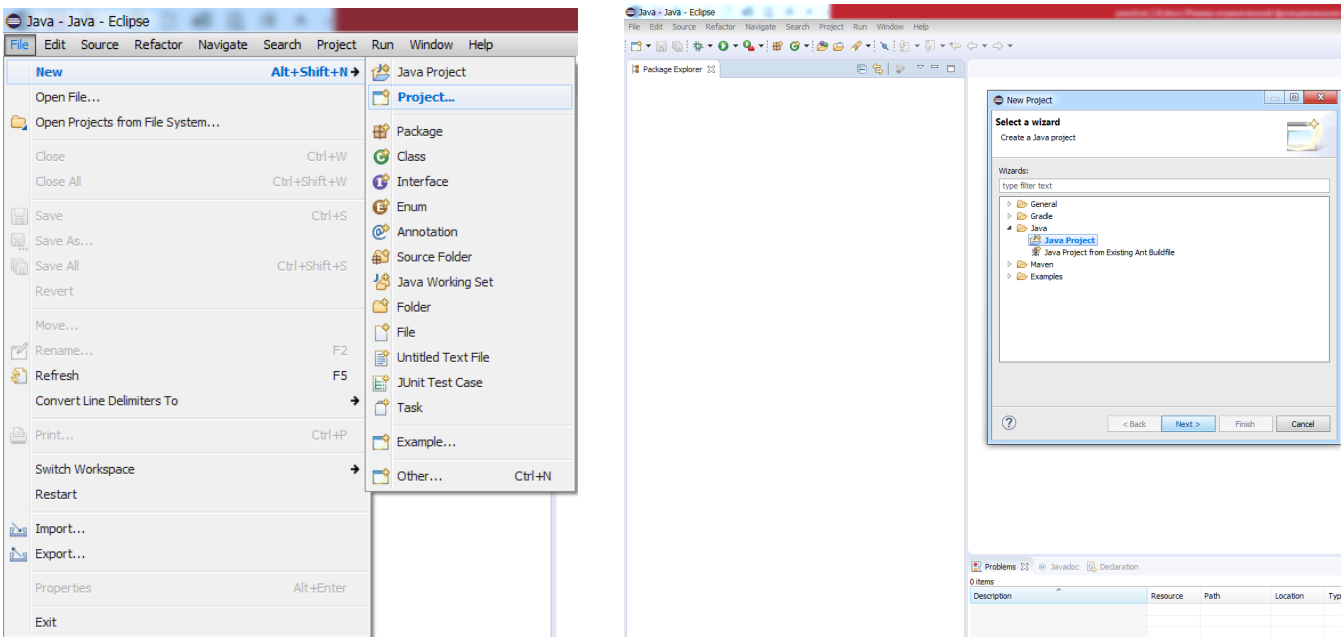
З метою наочності реалізуємо програму виводу стрічки «Привіт світе!» з використанням IDE.

Під час запуску інтегрованого середовища розробки Eclipse необхідно зазначити каталог де зберігатимуться усі Java-проекти. Якщо середовище щойно встановлене та раніше не використовувалось, то таку директорію необхідно створити та вказати під час першого запуску середовища (створення каталогу доступне під час запуску IDE).

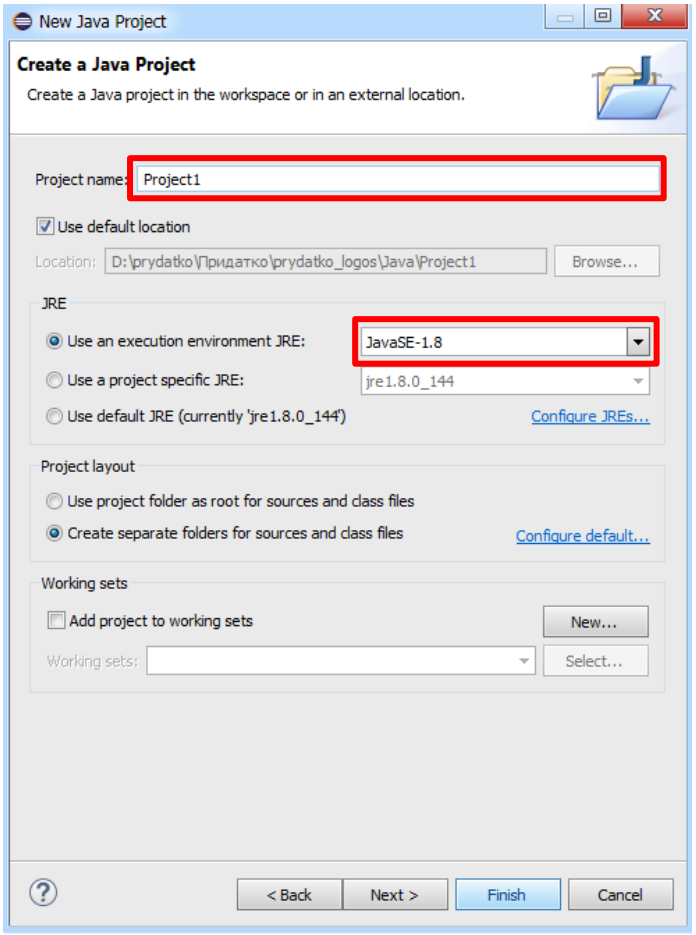




Після першого запуску Eclipse область Packege Explorer, звичайно, буде пустою. Для створення першого проекту необхідно перейти в меню File>New>Project. В вікні створення нового проекту необхідно вибрати Java>Java Project та пройти далі.

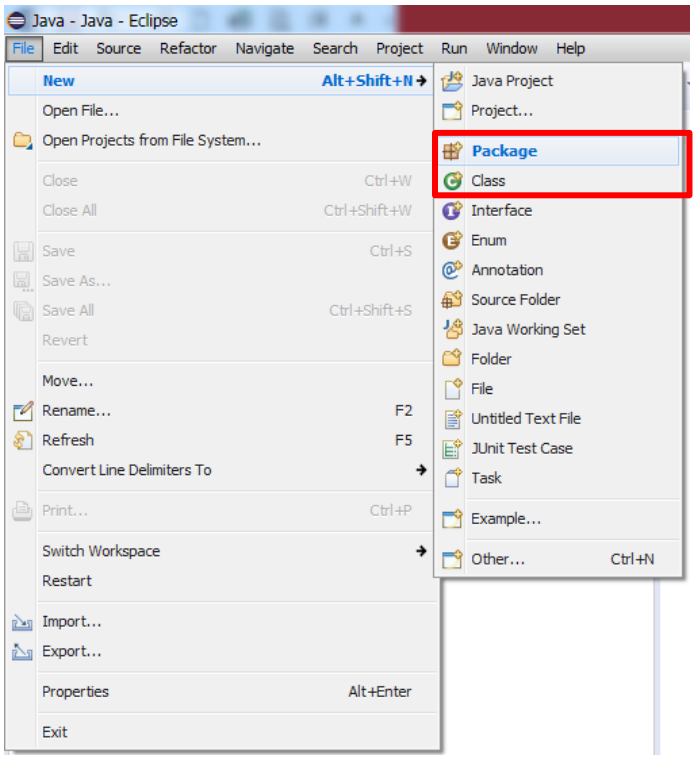


Наступним кроком є присвоєння імені для проекту. В цьому ж вікні можна вибрати версію Java, залежно від особливостей процесу розробки. За замовчуванням буде запропоновано останню версію JDK, яка встановлена на комп'ютері.

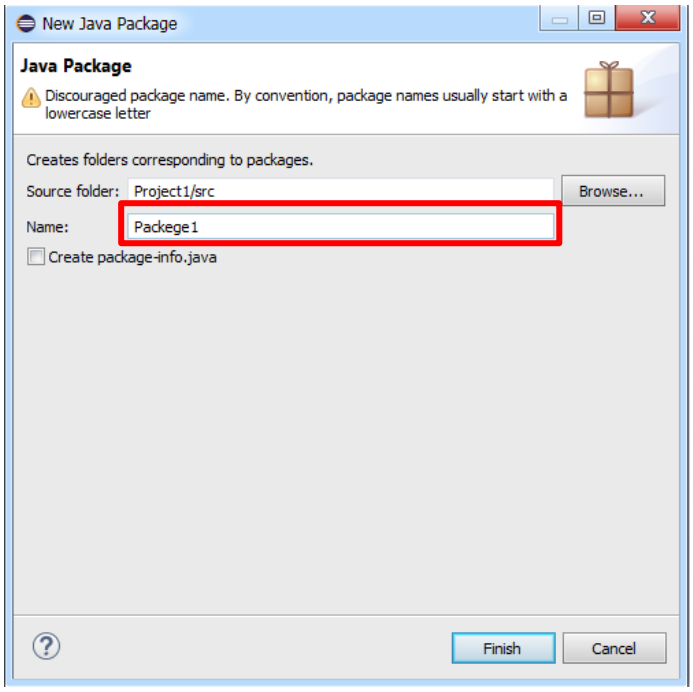


Java-проекти створюють з метою розміщення класів, проте деякі проекти можуть мати велику кількість класів (особливості об'єктно-орієнтованого програмування), тому з метою їх логічної структуризації в Java-проект введено елемент пакету. Для створення пакету необхідно знову звернутись до меню File>New>Packege (за

умови виділеної папки з проектом) або натиснути на директорію пакету правою клавішею маніпулятора (миші) та виконати New>Packege. Аналогічним чином в межах пакету створюється клас. Нагадуємо, що клас являється основним файлом, що містить первинний код програми.



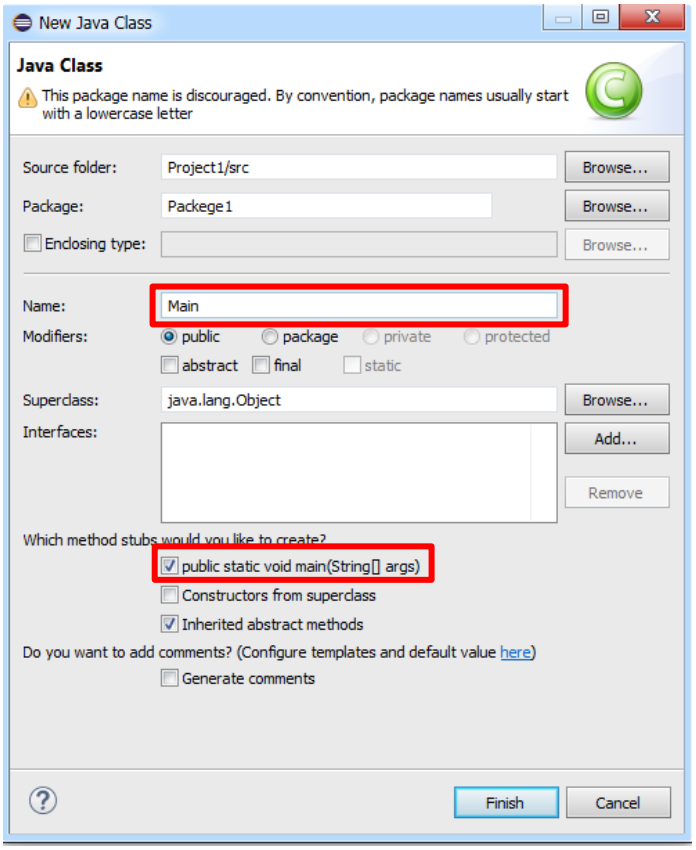
Після вибору позиції «створити новий проект», середовище запропонує дати назву новостворюваному пакету.



Новостворений пакет буде автоматично розміщений в каталозі «src». Слід зауважити, що в каталозі «src» буде збережено усю структуру Java-проекту. Крім того,

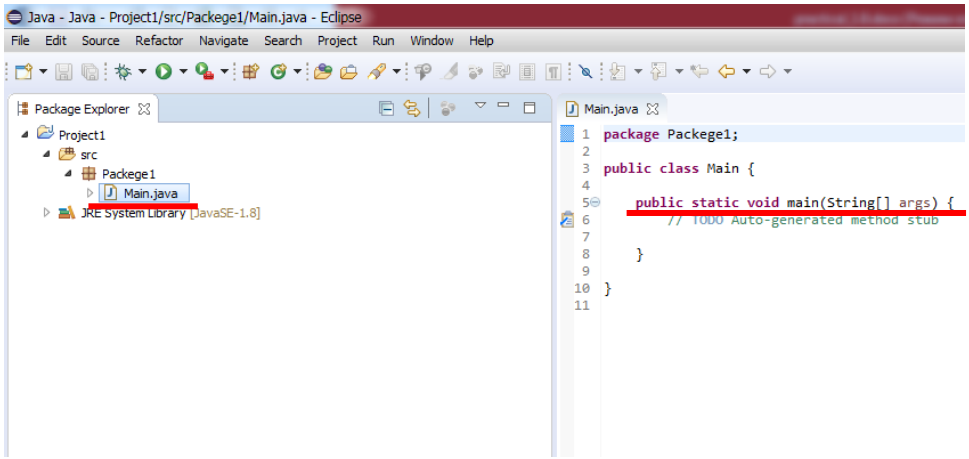
при створенні нового проекту автоматично створиться бібліотека JRE System Library, яка потрібна для запуску програми.

Дещо інший порядок створення класу. Виконавши аналогічні дії за умови виділеного новоствореного пакету (File>New>Class), середовище виведе вікно де необхідно зазначити назву класу. **Важливо!!!** Усі класи прийнято називати з великої літери. В прикладі це буде клас Main (головний). В зазначеному вікні користувачеві надається можливість встановити модифікатор доступу (про це пізніше), вибрати в якості місця збереження інший пакет, автоматично створити головний метод, генерувати конструктор або коментарі тощо. Для початку ми не будемо користуватись усіма представленими можливостями, окрім того, що встановимо позначку навпроти автоматичної генерації методу «public static void main (String [] args)» (щоб не набирати її потів власноруч).



Переваги використанняи IDE для створення класів очевидні. В результаті автоматичної генерації головного методу користувачеві не потрібно писати код із назвою класу, оголошувати метод, слідкувати за синтаксисом, щоб не допустити помилки тощо. Усі ці операції Eclipse виконує автоматично.

У результаті проведених дій в Packege Explorer буде відображено новоствореный метод, а в області написання коду – згенеровано головний метод:



В області написання коду Eclipse автоматично генерує стрічку:

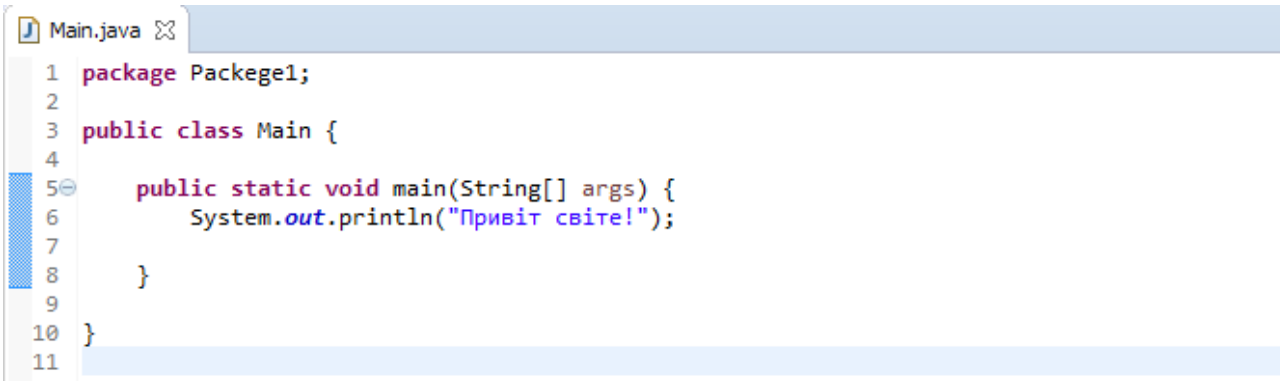
```
// TODO Auto-generated method stub
```

Це коментар, який не відіграє в програмі жодного значення окрім інформування про певні особливості того чи іншого методу. В нашому випадку цю стрічку можна видалити.

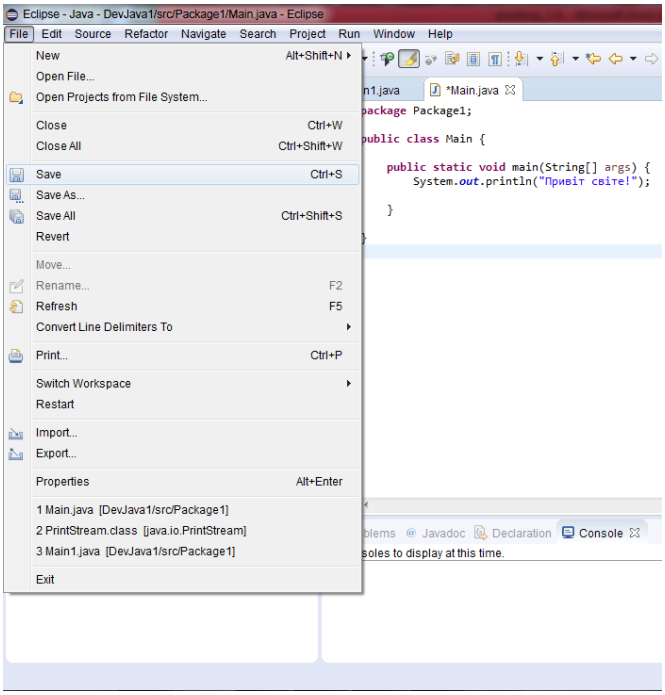
З метою формування власного коментаря необхідно в любій довільній області виділити необхідний текст та натиснути комбінацію клавіш Ctrl+/ (за умови англійської розкладки клавіатури).

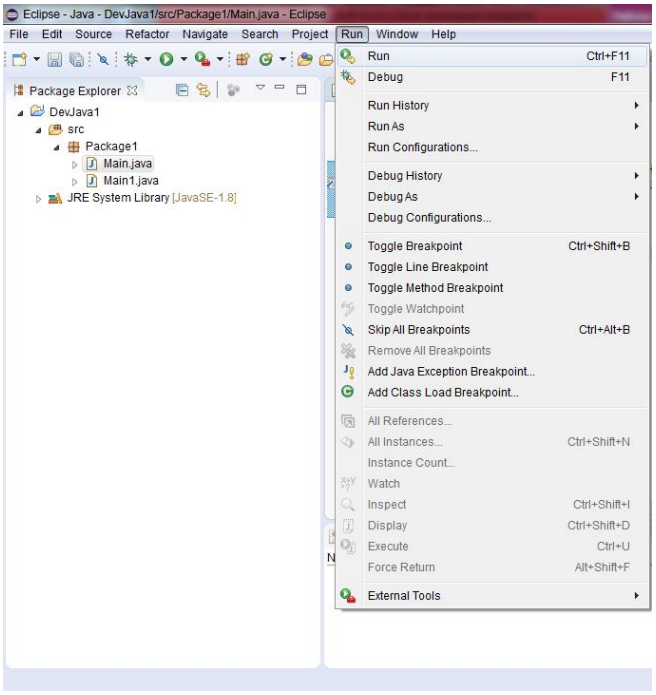
Якщо звернутись до первинного коду програми із виводу стрічки «Привіт світе!» (який було написано раніше), то можна відстежити, що у генерованому в Eclipse варіанті не вистачає лише (потрібно дописати):

```
System.out.println("Hello world");
```

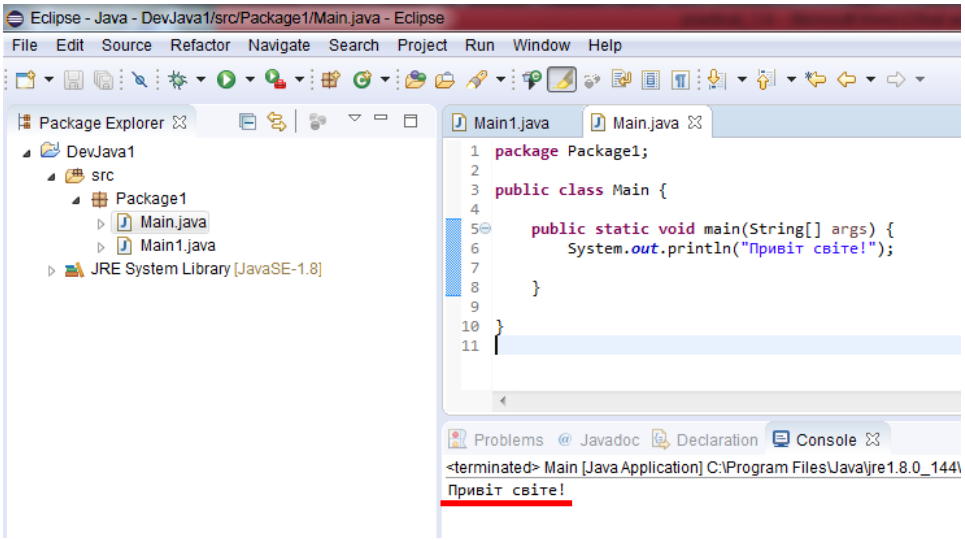


Завершивши написання коду необхідно зберегти напрацювання (File>Save або Ctrl+S) та запустити компілятор. Запуск коду на компіляцією із подальшим виконанням програми та виведенням результату в консоль здійснюється з допомогою натискання кнопки Run, меню Run>run, або комбінацією клавіш Ctrl+F11.

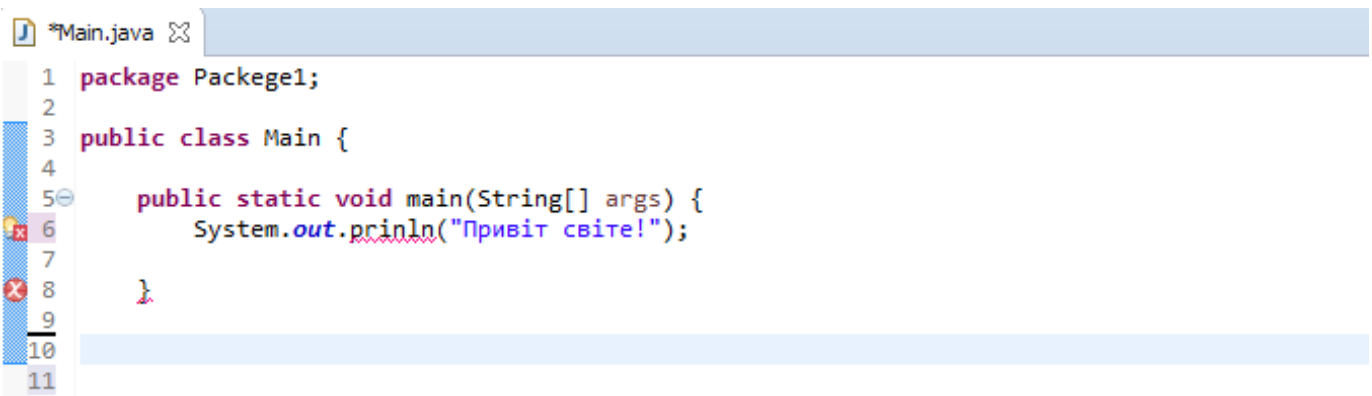




В результаті роботи програми в консоль буде виведено стрічку «Привіт світе!».

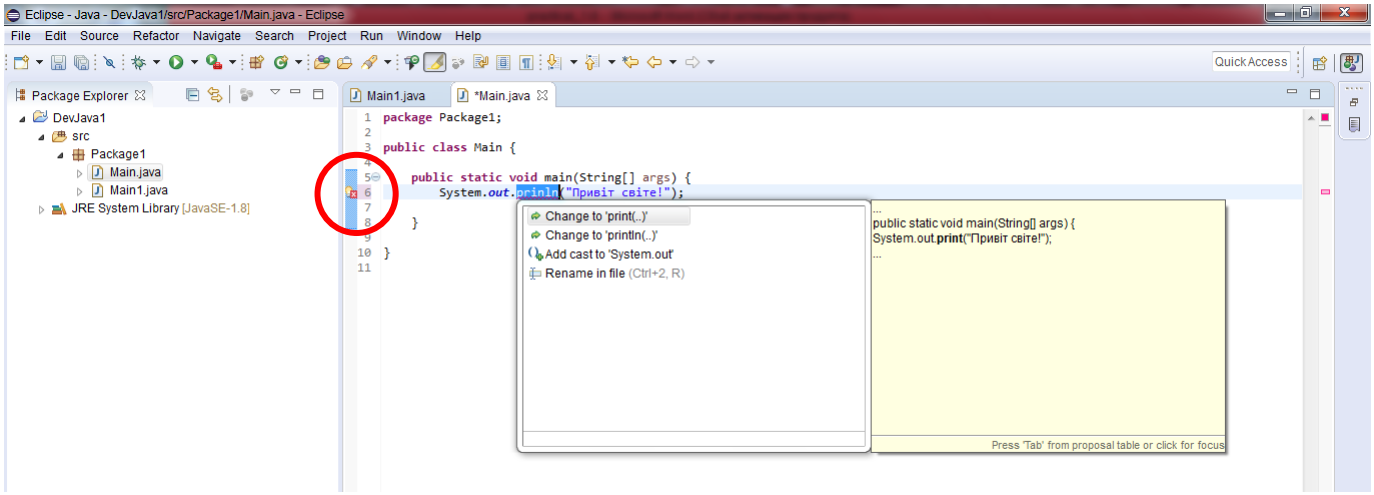


При допущенні помилки в процесі написання коду, компілятор автоматично повідомлятиме користувача про це:



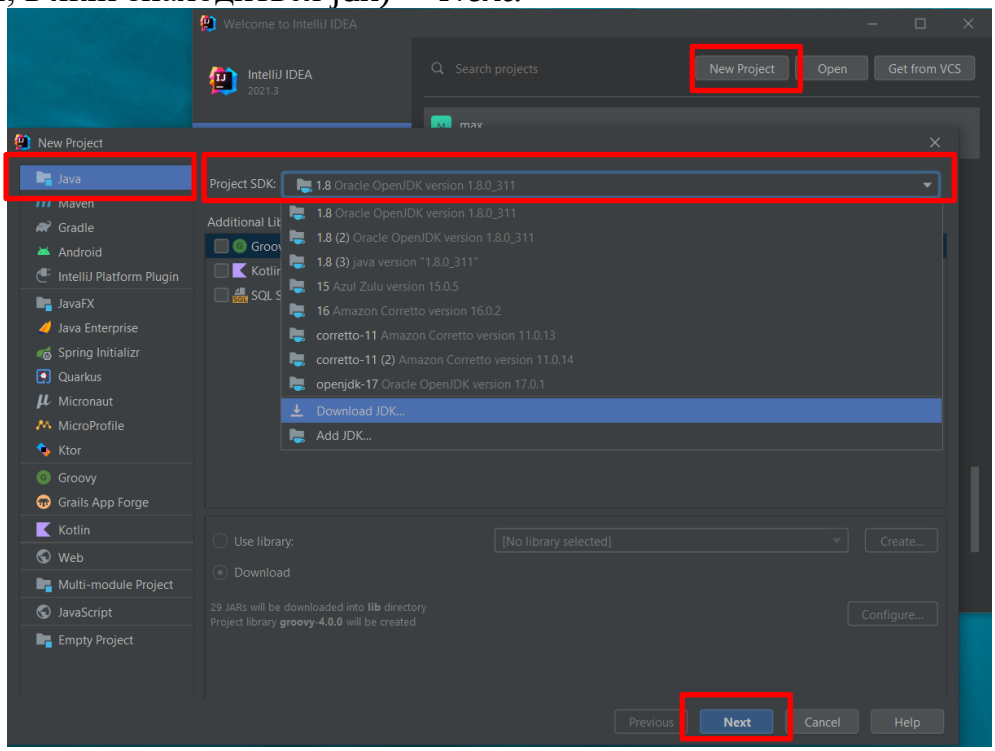
Існує два типи помилок, синтаксична та логічна. Синтаксичні помилки пов'язані недотриманням правил написання коду, наприклад відсутня дужка, крапка з комою, лапки тощо. Логічні помилки це помилки у написання логіки коду, наприклад застосування не того методу, невірна ініціалізація змінної, або ж помилка у написанні методів (як зроблено в прикладі з методом println). В обох варіантах компілятор надає допомогу щодо можливих варіантів виправлення помилок.

Для того щоб скористатись допомогою компілятора у виправлені помилки достатньо натиснути на піктограму, яка сигналізує про неї. В результаті чого середовище надає перелік можливих варіантів виправлення помилки у вигляді контекстного меню:

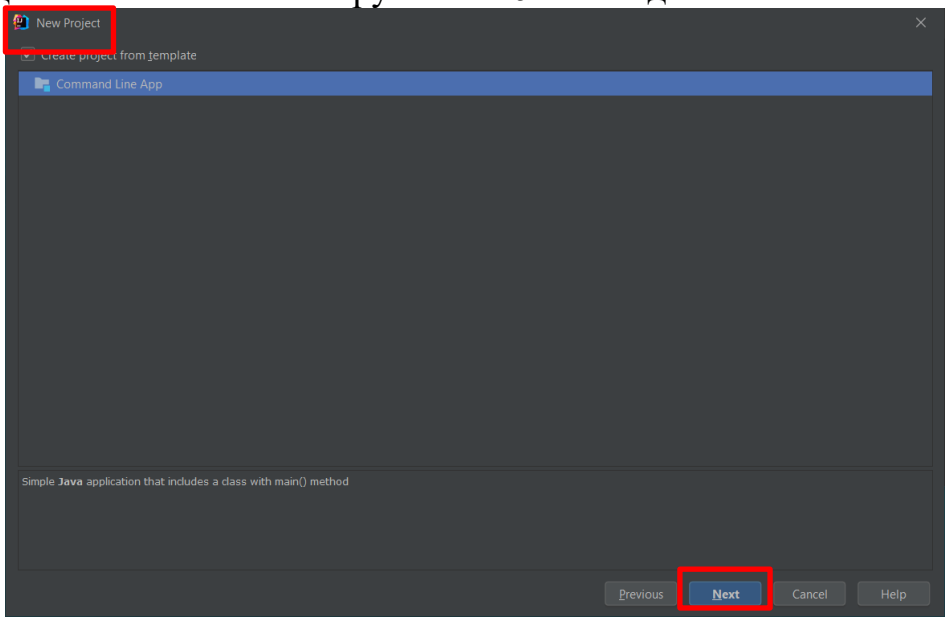


Приклад реалізації програми у інтегрованому середовищі IntelliJ IDEA

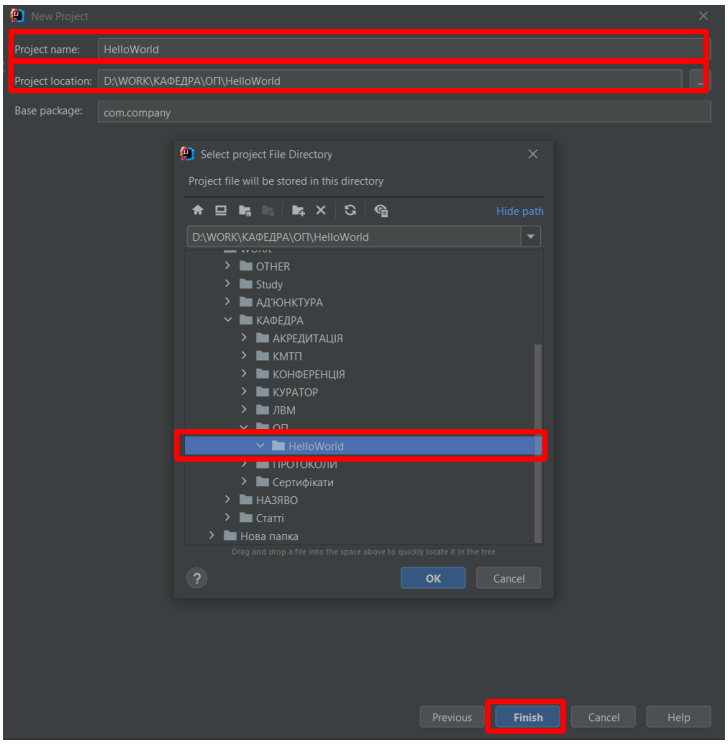
Для того, щоб створити проект у IntelliJ IDEA вибираємо ***New Project*** -> ***Java*** -> ***Select SDK вказати jdk*** (перший раз програма може попросити вказати прямий шлях до папки, в якій знаходиться jdk) –>***Next***.



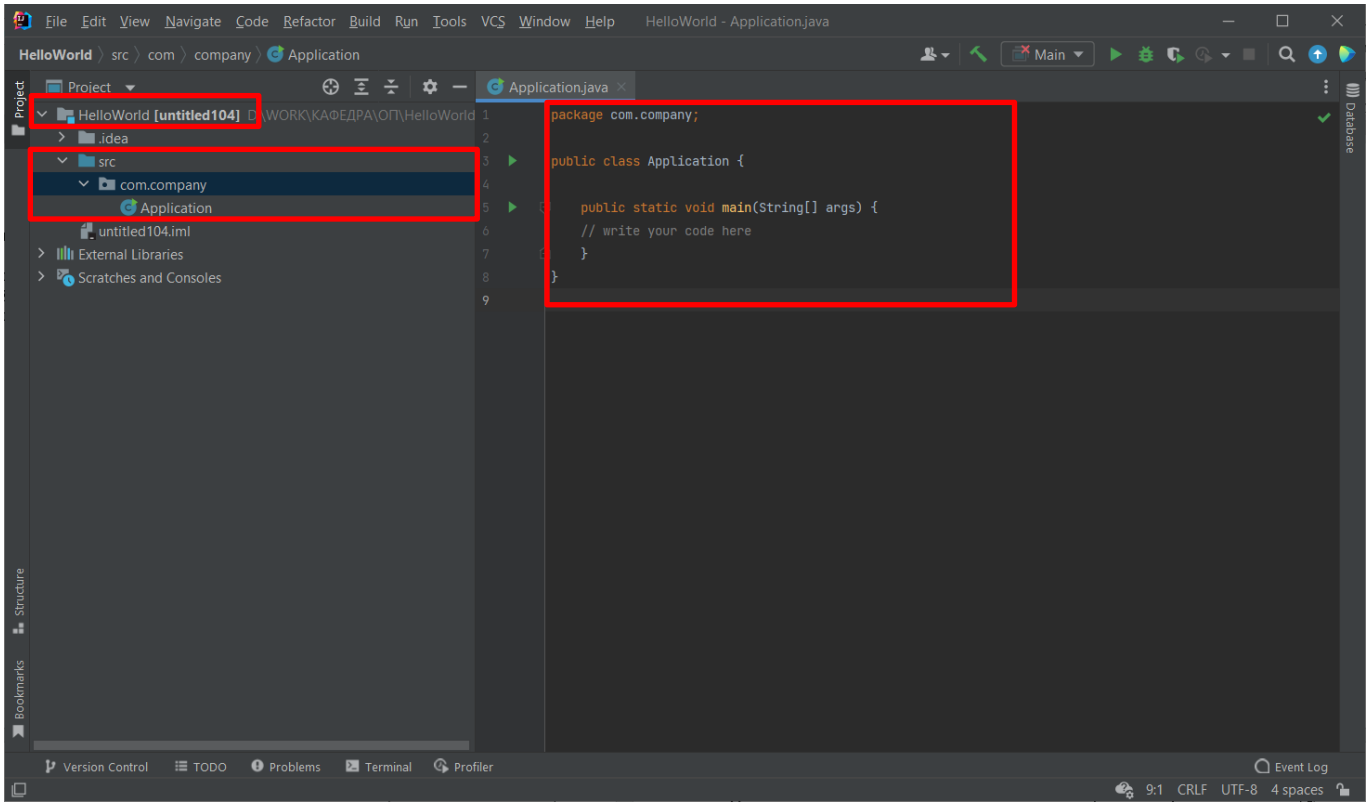
Після чого відкриється нове вікно, де потрібно буде вибрати ***Create project from template***, щоб автоматично згенерувався main-метод.



В новому вікні вказуємо назву проекту та місце, де він буде зберігатись та вибираємо ***Finish***.



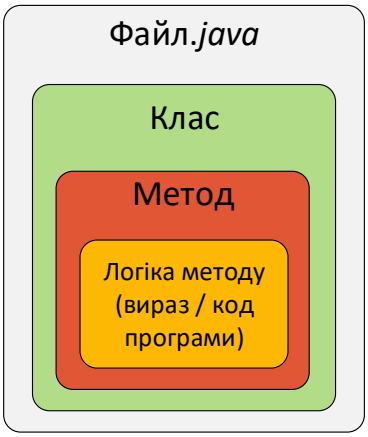
Все проект створений, залишилось реалізувати логіку. Вся логіка проекту реалізується у класах, які знаходяться у папці ***scr***. При створенні проекту у вас мав би згенеруватись клас ***Main.java*** (у моєму випадку – Application.java) та каркас методу ***main().***



2.2. Структура коду програми

Для кращого розуміння ієрархії коду програми проведено його короткий аналіз. Для прикладу візьмемо вже добре відомий код програми із виводу стрічки «Привіт світе!»:

```
public class HelloWorld {
      public static void main (String []
      args){
            System.out.println("Привіт світе!");
      }
}
```



Для кращої уяви про структуру програмного коду поділимо його на три категорії: клас; метод; вираз. Усі з зазначених категорій виділені різними кольорами та відповідають наведеним прикладам.

Файл з первинним кодом (розширення java) містить в собі клас. Власне клас це і є програма, або її частина. Маленькі програми можуть складатись з одного класу. Усе, що віднесено до класу розміщується між фігурними дужками (так званий блок класу):

```
public class HelloWorld {
```

```
    public static void main (String [] args){
        System.out.println("Привіт світе!");
    }
}
```

Клас може вміщувати один або декілька методів. Власне методи містять в собі усі інструкції щодо виконання програми (код програми). Методи обов'язково потрібно розміщувати в середині класів (між фігурними дужками класу).

```
public class HelloWorld {
    public static void main (String [] args){
        System.out.println("Привіт світе!");
    }
}
```

Логіка методу (інструкції щодо виконання програми) має розміщуватись в середині методу. Іншими словами, код методу – це набір його виразів (що та як він робить).

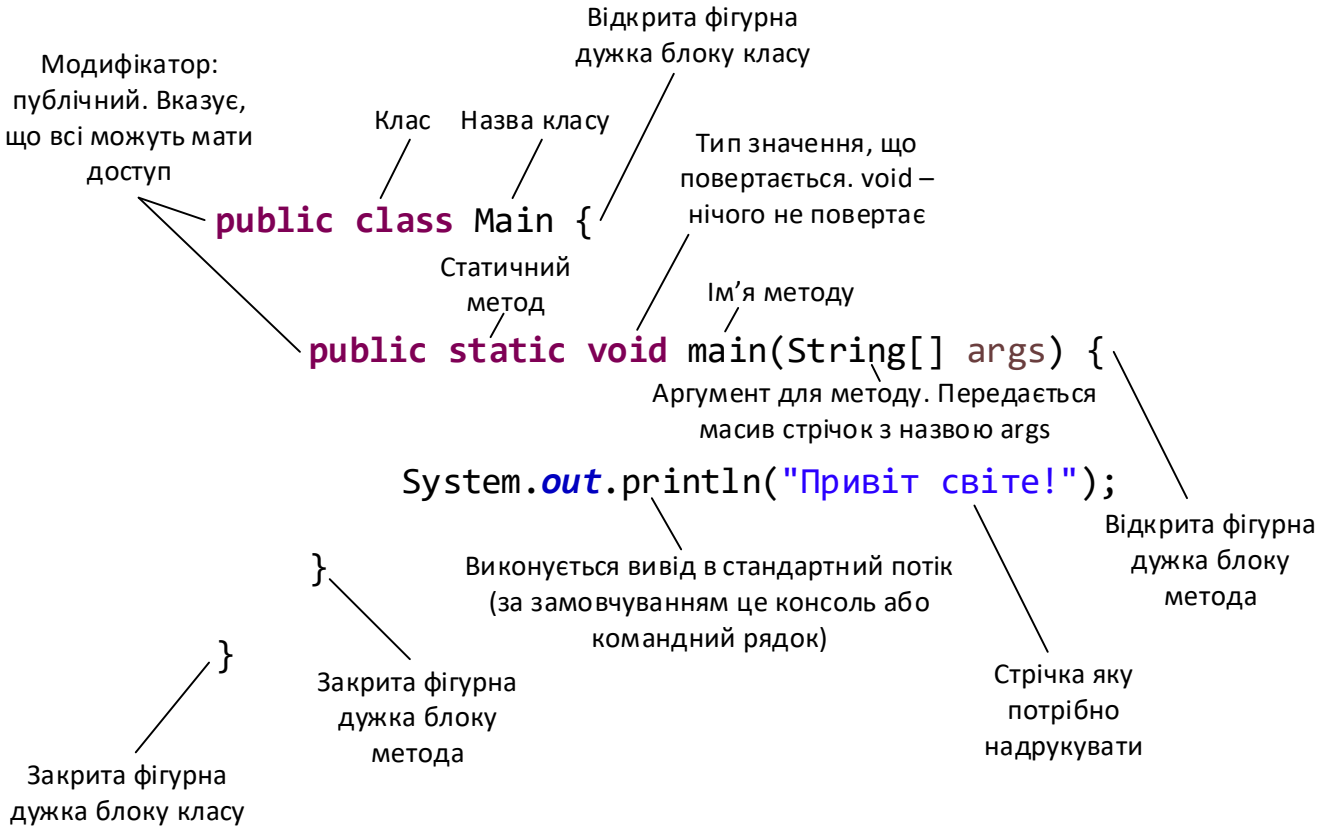
```
public class HelloWorld {
    public static void main (String [] args){
        System.out.println("Привіт світе!");
    }
}
```

В наведеному прикладі вираз методу виводить у консоль стрічку «Привіт світе!».

Отже, узагальнимо ієрархію коду: ***клас → метод → вираз (логіка програми).***

Коли JVM (віртуальна машина Java) починає свою роботу, вона першою чергою шукає клас (власне чому клас в ієрархії коду займає перше місце). Після знаходження класу JVM шукає метод та заходить в нього. Далі JVM почерзі виконує все, що знаходиться між фігурними дужками головного методу. Будь-яка програма написана на Java має що найменше один клас та головний метод.

Тепер зоглянемо детальніше кожен елемент представленого коду:



Не потрібно намагатись відразу запам'ятати всю структуру коду та вивчити, що як називається. Основна мета – це *ознайомлення* з елементами коду. Далі в процесі вивчення основ програмування користувачі неодноразово звертатимуться до структури та ієрархії коду, та більш детально ознайомляться з ним.

2.3. Видача індивідуальних практичних завдань

Вкінці практичного заняття студенти отримують індивідуальне практичне завдання, результат виконанням якого необхідно завантажити в віртуальне навчальне середовище (категорія: контрольна частина).

Завдання полягає у написанні найпростішої програми із виводу інформації про свою родину в такому форматі:

```
Батько: Чумаченко Степан Дмитрович, 1965 року народження, підприємець;
… …
Сестра: Чумаченко Надія Степанівна, 1998 року народження, студентка;
… …
```

Первинний код програми зберігається у файлі з розширенням .java за адресою яку вказано під час запуску середовища розробки **в каталозі «src»**:

…\назва каталогу з Java-проектом\назва проекту**src**\назва пакету**файл.java**

Методичну розробку розробив:
Заступник начальника кафедри УПІТтаТ
підполковник служби цивільного захисту Олександр Придатко

Методична розробка обговорена на засіданні кафедри УПІТтаТ
Протокол № ____ від "_ __" _____ 20__ р.