**Лабораторна робота №5**

**Мета роботи:** засвоїти технологію покрокової перевірки програм; визначати стан змінних; встановлювати умови зупинки програм у точках переривання.

**Постановка задачі**

Налагодження - це процес покрокової перевірки програм і додатків з метою виявлення помилок.

Процес налагодження характеризується тим, що програма зупиняється щоразу в точках переривання. Якщо точки переривання не встановлені, то налагодження не буде відрізнятися від звичайного виконання програми.

Розглянемо налагодження на прикладі. Створимо новий клас з ім'ям Debug і введемо запропонований нижче код.

**publicclass** Debug

{

**publicstaticvoid** main(String[] args)

{

**for** (**int**i=0;i<10;i++)

{

System.***out***.print("Шаг "+i+"\n");

}

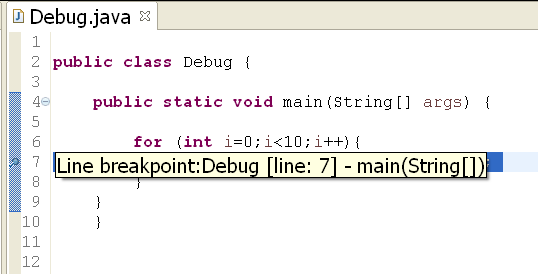
}

}

Для запуску відладчика натиснемо кнопку із зображенням жука.

При натисканні на цю кнопку включається режим налагодження поточної програми, якщо вікно коду програми активно. Якщо активна панель **PackageExplorer**, то налагоджується обраний (виділений) клас, при цьому класи можна додатково вибрати, якщо їх декілька.

На рис. 2.1 виділено поле, в якому користувач може встановити точки переривання.



Ріс.2.1.Лінійка з точкою переривання

Для того щоб поставити крапку, потрібно двічі клацнути мишкою в даному полі навпроти потрібного рядка, де планується зупинка програми. Маркери точок переривання є маленькі квадратики.

Встановимо точку переривання навпроти рядка

System.***out***.print("Шаг "+i+"\n");

Після того, як була встановлена точка переривання і натиснута кнопка, на екрані з'явиться діалогове вікно (рис.2.2), що пропонує перемкнути режим робочого столу з Java в Debug.

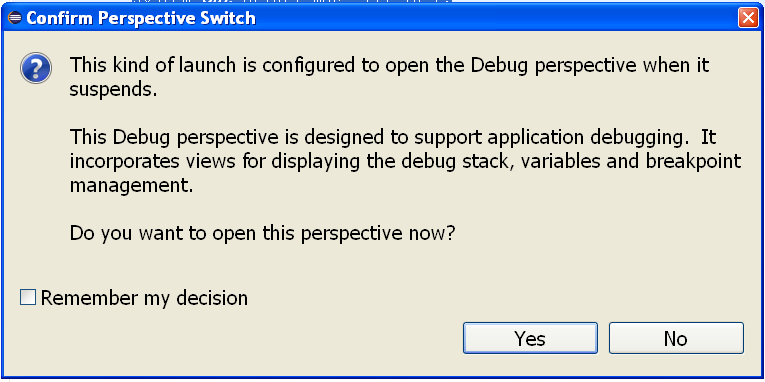


Рис. 2.2. Пропозиція переключити робочий стіл

Для того щоб переключитися назад в режим Java, використовують кнопку, що знаходиться у верхньому правому куті робочого столу. Далі, приймаємо дану пропозиція, натиснувши кнопку «Yes», і в результаті робочий стіл прийме вигляд, представлений на рис. 2.3.

Це стандартний, за замовчуванням, вид компонування Debug, але його можна змінити: видаляючи, додаючи або перетягуючи панелі мишею.

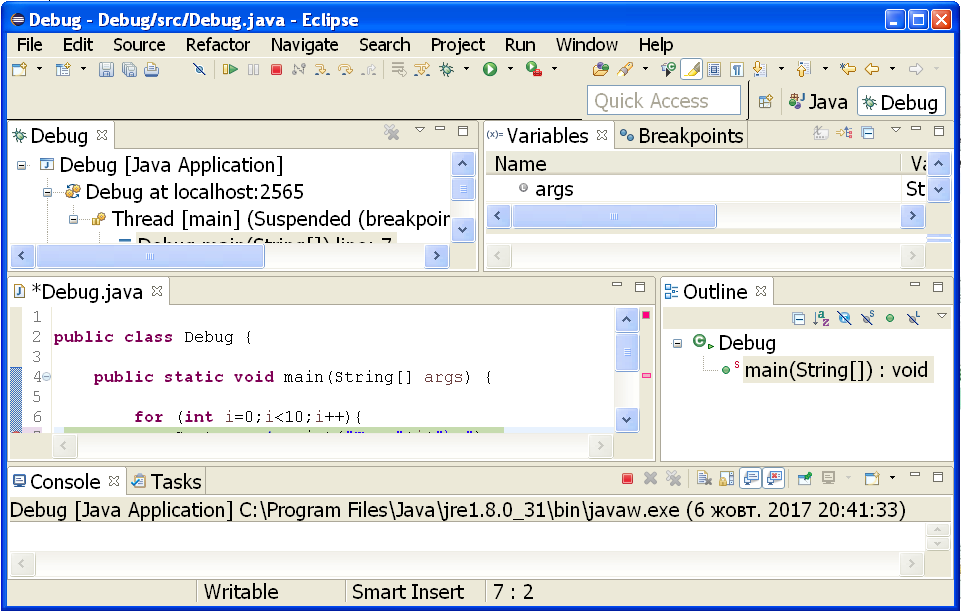


Рис. 2.3. Робочий стіл Debug

У компонуванні Debug можна виділити наступні основні компоненти:

● Вікно **Debug**. В даному вікні відображаються задіяні в налагодженні елементи, а також панель управління процесом налагодження (рис. 2.4).

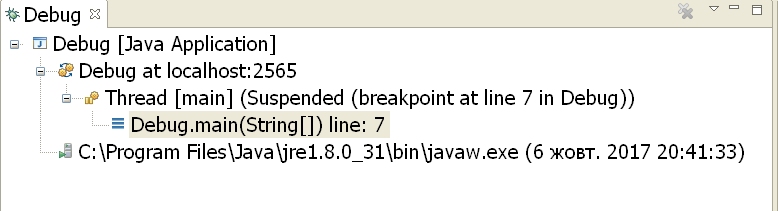


Рис.2.4. Вікно Debug

● Вікно стану змінних**Variables** і точок переривання **Breakpoints**, рис. 2.5. Вкладка **Variables** містить список змінних, задіяних в поточній точці переривання. Дане вікно є найбільш важливим в процесі налагодження, воно здійснює моніторинг значень змінних на кожній точці переривання. У лівому стовпчику вказані ідентифікатори (імена) змінних, в правому - їх поточні значення

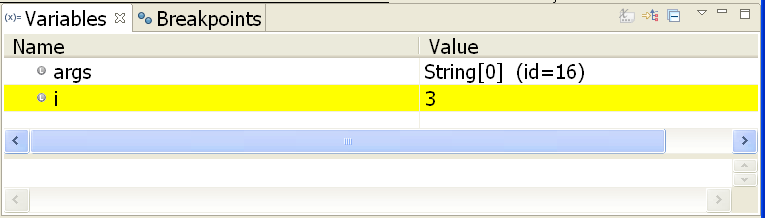


Рис.2.5. Вікно змінних

Вкладка **Breakpoints** містить список точок переривання, які встановив програміст. Точки переривання можна відключати і включати в процесі налагодження, встановлюючи або видаляючи маркер на проти потрібної точки.

Після переходу в режимі налагодження та запуску програми отладчик послідовно виконує оператори програми до першої точкипереривання і призупиняє хід виконання. Програміст має можливість подивитися значення змінних програми на момент зупинки. Після того як стан програми був проаналізован, її можна продовжувати, натиснувши кнопку ( «Resume») або клавішу F8 для продовження.

У нашому випадку програма була зупинена на операторі виведення даних на консоль. Після продовження, виконання тіла циклу продовжиться. Так як в тілі нашого циклу тільки один оператор, він виконується, і на консоль виведеться перший рядок. На першому кроці змінна i = 0, в цьому можна переконатися переглянувши вкладку **Variables** вікна стану (рис. 2.6).

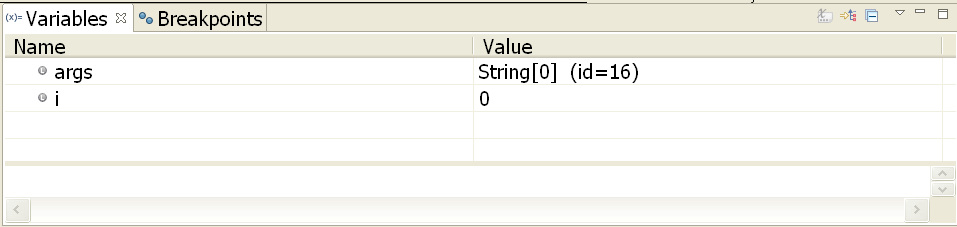


Рис. 2.6. Стан змінних на першому кроці

Після виведення рядка цикл буде виконуватися ще раз, змінна зміниться і буде дорівнювати 1.

І так далі, натискаючи клавішу F8 або кнопку, програма буде переходити від однієї точки переривання до іншої.

У отладчику Eclipse є ще одна корисна функція, за допомогою якої можна побачити значення складних виразів. Наприклад, нас цікавить значення логічного виразу (умова), при якому виконується цикл. Для цього виділімо потрібний фрагмент коду, в нашому випадку це **i <10** в блоці умови циклу, і натиснемо клавіші **<Ctrl> + <Shift> + <D>**. З'явиться спливаюче вікно, в якому буде відображений тип виразу (в нашому випадку це boolean) і його стан, рис. 2.7.

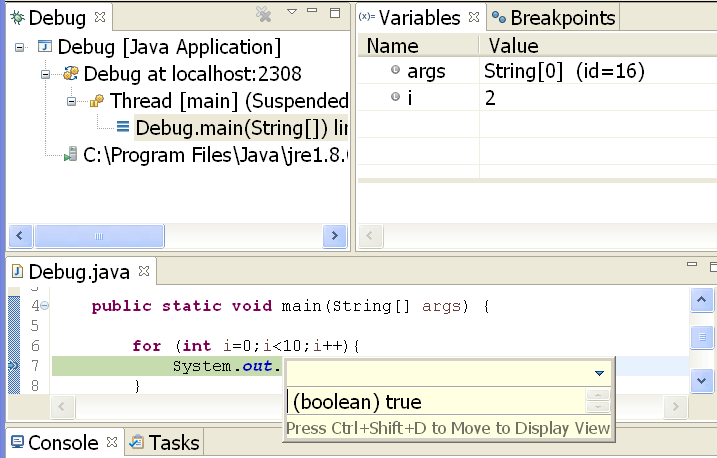


Рис. 2.7. Визначення логічного виразу

Коли програма дійде до кінця, налагодження закінчиться. Для повтору потрібно ще раз натиснути кнопку.

Для переривання роботи відладчика потрібно натиснути кнопку в панелі Debug.

Крім налагодження із використанням безумовних точок переривання, є можливість додатково встановлювати умови зупинки на точках переривання.

Для того, щоб задати умову, за якої буде відбуватися зупинка, потрібно відкрити вікно властивостей точки переривання шляхом наведення курсору мишки на потрібну точку. Далі натиснути праву кнопку миші, і вибрати в меню пункт **BreakpointProperties**. Відкриється вікно властивостей обраної точки переривання (рис.2.8).

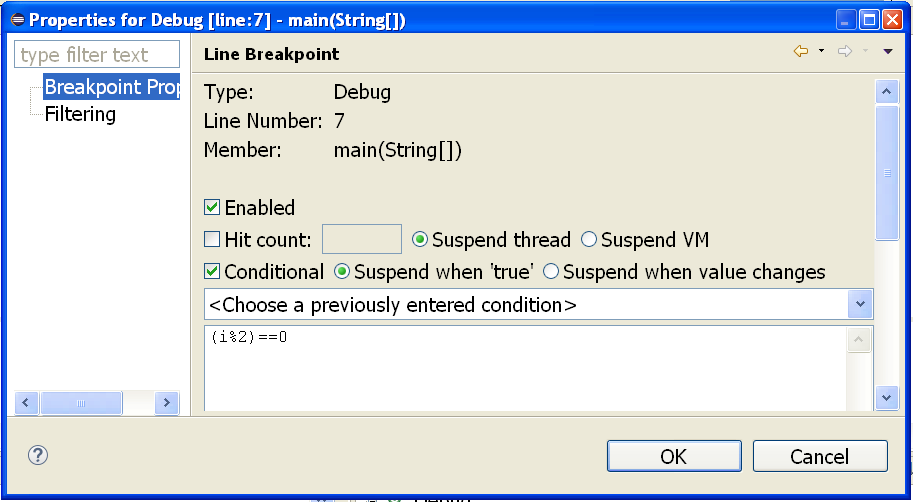


Рис. 2.8. Властивості точки переривання

Встановимо галочки на перемикачах вибору **Enable**та **Conditional** (ВключітьУсловіе) і в полі введемо необхідну умову. Для прикладу, встановимо логічний вираз **(i%2) == 0**. Цей вираз буде приймати значення «істина» кожен раз, коли залишок від ділення i на 2 буде дорівнює 0, тобто кожен раз, коли i буде приймати парні значення. Запустимо процес налагодження та переконаємося в тому, що тепер зупика відбуватиметься через кожні два проходи циклу, а не один, як було раніше.

Так само можна вибрати тип умови переривання. При виборі радіо кнопки «**Suspend when “true”»,**зупинка відбуватиметься кожного разу, коли введений логічний вираз буде приймати значення «істина». При виборі другої радіо кнопки «**Suspend when value changes**», зупинка відбуватиметься кожного разу, коли введене значення або константа буде змінюватися.

При виборі значення «**Suspend thread»**зупинка відбуватиметься на рівні поточного потоку thread, при виборі «**Suspend VM**» -на рівні самої віртуальної машини Java.

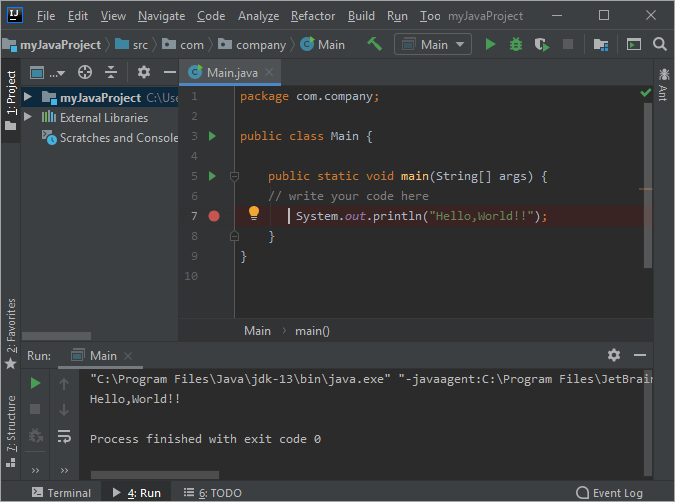
**Налагодження програми в IntelliJ IDEA**

За допомогою налагодження ви зможете простежити запущений код, перевірити стан, в якому знаходиться код, а також перевірити потік виконання. Налагоджувач, запропонований IntelliJ IDEA, допомагає вам налагоджувати все і все, починаючи від найпростішого коду і закінчуючи складними програмами.

Використовуючи вбудований налагоджувач IntelliJ IDEA, ви можете виконувати різні операції, такі як зупинити виконання програми в будь-який момент, ввійти та вийти з будь-якої функції, встановити різні точки зупинку, перевірити та стежити за змінними, оцінити вираз програма тощо.

Точки зупинки в програмі дозволяють програмі зупинятися на певних точках. Точки зупинку спрацьовують, коли виконання програми досягає рядка, де встановлена ​​точка зупинку.

На скріншоті нижче точка зупинку вказана в першому рядку коду в головній функції. Він позначається червоним колом проти номера рядка.

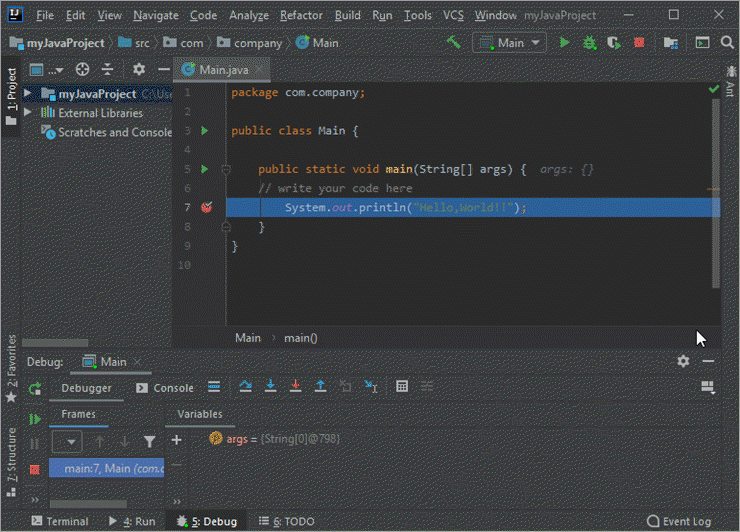


Встановити точку зупинки

Ви можете вказати точку зупинки у своєму коді, вибравши відповідний пункт меню (перемикати точку зупинки) в меню “Виконати” або використовуючи комбінаційні клавіші “Ctrl + f8”.

Після встановлення точки зупинки ви можете розпочати налагодження програми, вибравши опцію «Налагодження» в меню «Виконати», або скориставшись комбінаційною клавішею «Alt + shift + f8». Це запропонує вам варіанти налагодження для налагодження основної або будь-якої іншої функції.

На панелі інструментів також є параметр налагодження для запуску налагодження. Після запуску налагодження відображається наступний екран.



Налагоджувач

Як показано на екрані вище, відображається вікно вкладки налагоджувача. Ви можете виконувати різні інші операції, такі як перевірка значень змінних, оцінка виразів, крок у функцію, вихід із функції, зупинка налагодження, відновлення налагодження тощо.

За допомогою цих операцій ви можете ефективно налагодити програму та виявити проблему з програмою.

**Завдання до роботи**

За допомогою налагоджувача вашого середовища розробки встановіть Breakpoints, знайдіть помилки у коді та виправте їх згідно наступної умови.

Вася замовив посилку з інтернет-магазину в поштомат. Через деякий час йому прийшло повідомлення з пін-кодом, який, в тому числі, може починатися з 0. Поштомат дає користувачу 5 спроб вводу пін-кода, і якщо користувач вводить неправильний пін-код, то система надсилає СМС з новим пін-кодом. Визначте, чи зможе Вася забрати свою посилку?

На вхід програмі даються рядки попарно: пароль, який прийшов Васі і пароль, який він ввів. Якщо Вася вводить неправильний пароль, то на екран виводиться "INCORRECT n" (де n - номер спроби), а якщо правильний, то "CORRECT". Якщо Вася вичерпав усі спроби, то з’являється рядок "Error".

Початковий код:

import java.util.Scanner;

class A{

public void main(String[] args){

Scanner sc = new Scanner(System.in);

int counter = 1;

boolean pin\_code\_coincided = true;

while (true){

String correct = sc.nextLine();

String vasya = sc.nextLine();

counter++;

if(correct.equals(vasya)){

System.out.println("СORRECT");

break;

}

System.out.println("INCORRECT" + counter);

}

if (!pin\_code\_coincided){

System.out.println("Error");

}else{

System.out.println("CORRECT");

}

}

}

Sample Input 1:

1234

1235

3478

1234

5623

5623

Sample Output 1:

INCORRECT 1

INCORRECT 2

CORRECT

Sample Input 2:

1234

1235

3478

1234

5623

5624

0534

5623

4512

3214

Sample Output 2:

INCORRECT 1

INCORRECT 2

INCORRECT 3

INCORRECT 4

INCORRECT 5

Error