Nội dung

[Link1.Các cách để Debug Java project trong Eclipse 2](#_Toc147601872)

[Link1.Các phím tắt để Debug Java project trong Eclipse 2](#_Toc147601873)

* [Link2.Khái quát 2](#_Toc147601874)

[**1.** **Debug là gì?** 2](#_Toc147601875)

[**2.** **Các phương pháp Debug** 2](#_Toc147601876)

[**3.** **Debug code Java trong Eclipse** 2](#_Toc147601877)

[1. Tạo BreakPoint. 2](#_Toc147601878)

[2. Start chương trình ở chế độ Debug. 2](#_Toc147601879)

[3. Sử dụng các phím tắt để theo dõi giá trị các biến, theo dõi flow thực thi của chương trình. 2](#_Toc147601880)

[4. **Debug như thế nào mới hiệu quả?** 2](#_Toc147601881)

<https://viettuts.vn/eclipse/debug-java-project-trong-eclipse>

<https://gpcoder.com/3621-huong-dan-debug-code-trong-eclipse/>

Link1.Các cách để Debug Java project trong Eclipse

Dưới đây là các cách để Debug Java project trong Eclipse:

* Bằng cách chọn menu Run -> Debug As -> Java Application.
* Bằng cách click chuột phải vào class chứa phương thức main() -> Debug As -> Java Application.
* Bằng cách click vào button Debug trong thanh công cụ -> Debug As -> Java Application.
* Nhấn phím F11.

## Link1.Các phím tắt để Debug Java project trong Eclipse

**F11**: thực thi debug.

**F6**: thực thi câu lệnh hiện tại và nhảy đến câu lệnh tiếp tiếp.

**F8**: thực thi câu lệnh hiện tại và nhảy đến breakpoint tiếp theo.

**F5**: nhảy vào phương thức đang debug.

***(Có hướng dẫn cụ thể cách debug 1 bài đơn giản).***

Link2.Khái quát

1. **Debug là gì?**

Bug (lỗi) có thể là từ chương trình của bạn viết ra hoặc là từ chương trình mà bạn kế thừa từ người khác. Khi chương trình chạy và kết quả không phải là những gì mà bạn mong đợi, tức là đã có lỗi. Lỗi có thể do dự đoán của bạn đã bị sai hoặc là có gì đó phát sinh khi chương trình được thực thi mà bạn không lường trước được.

**Debug**(gỡ lỗi) là một kỹ năng nền tảng của lập trình viên. Mục đích chính của debug là để dò tìm lỗi (error) của chương trình, ngoài ra nó còn giúp lập trình viên hiểu rõ hơn cách chương trình hoạt động.

1. **Các phương pháp Debug**

Có nhiều phương pháp để debug, thông thường chúng ta sử dụng các phương pháp sau:

* **Print Console**: là cách thêm vào source code của bạn những dòng lệnh để in ra những thông tin mà bạn cần theo dõi trong quá trình thực thi. Ví dụ: System.out.println(), hay System.error.println(), … Cách này hơi phiền phức vì bạn phải thay đổi code của chương trình, những dòng code này vô nghĩa và sau khi debug bạn phải xóa khỏi source code của chương trình.
* **Logging**: tạo ra một tập tin để ghi (log) lại những thông tin sau khi chương trình thực thi. Chúng ta sẽ dựa vào thông tin này để phân tích nguyên nhân lỗi. Kỹ thuật này thường được áp dụng trong môi trường thực, khi mà ứng dụng của bạn đang được mọi người sử dụng, bạn không thể thay đổi code hay sử dụng công cụ để dò tìm lỗi.
* **Debugging Tool** (dùng công cụ để Debug) là phương pháp debug đi sâu vào source code nhất. Các công cụ này thông thường được tích hợp vào các IDE, ví dụ công cụ debugger của eclipse, Microsoft Visual Studio Debugger, … Cách này thường được sử dụng nhất khi bạn đang phát triển chương trình, bạn dễ dàng theo dõi, kiểm tra các giá trị, thay đổi các giá trị để kiểm tra tính đúng đắn của chương trình.

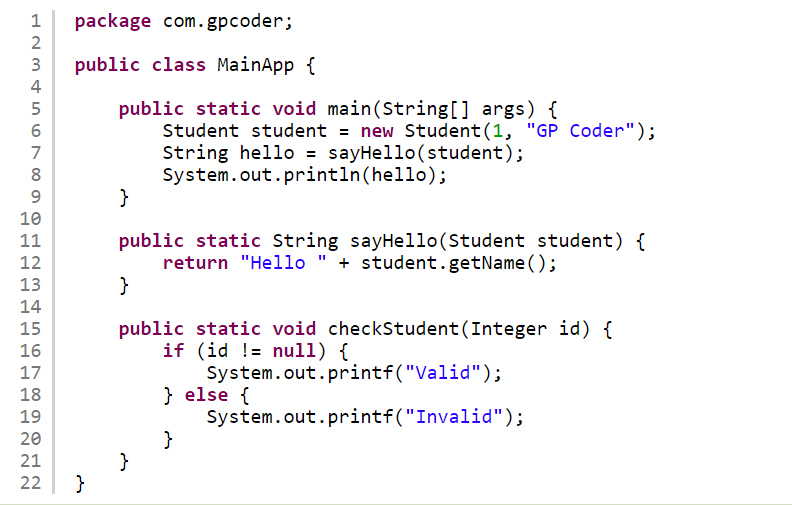
1. **Debug code Java trong Eclipse**

**Debug code Java** trong **Eclipse**cũng giống như việc chạy một ứng dụng Java. Khi debug chúng ta có thể tạm ngưng (pause) chương trình để xem giá trị của các biến, từ đó biết được luồng chạy của chương trình, thậm chí là thay đổi giá trị của các biến. Đặc biệt debug được sử dụng để tìm kiếm bug, đây là một việc quan trọng trong phát triển phần mềm.

**BreakPoint** (điểm ngắt): một BreakPoint trong mã nguồn xác định nơi thực thi của chương trình nên dừng lại trong quá trình gỡ lỗi. Một khi chương trình đã dừng, bạn có thể kiểm tra giá trị các biến, thay đổi nội dung, …

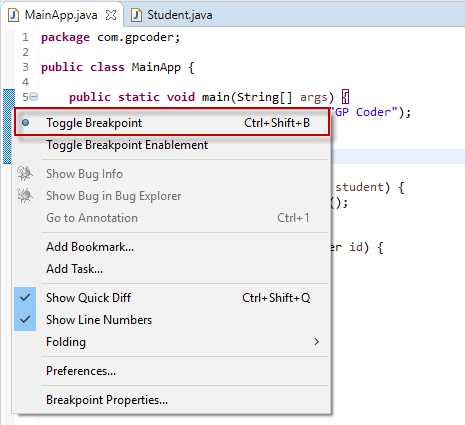
1. **Các bước debug code Java trong Eclipse**

Các bước thực hiện debug code trong Java: (code minh hoạ)

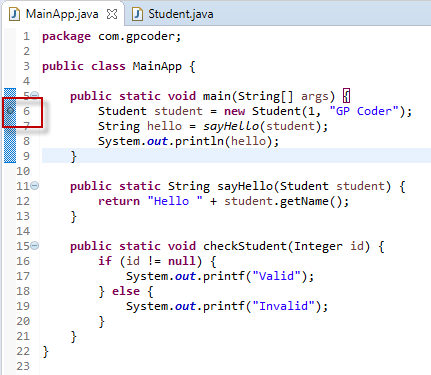


### Tạo BreakPoint.

Để xác định điểm ngắt trong mã nguồn, bạn nhấp chuột phải vào lề trái trong trình soạn thảo Java và chọn **Toggle Breakpoint**. Hoặc bạn có thể nhấp đúp vào vị trí cần đánh dấu BreakPoint.



Ví dụ tôi thiết lập một BreakPoint tại dòng code số 6, ta có kết quả sau:



### Start chương trình ở chế độ Debug.

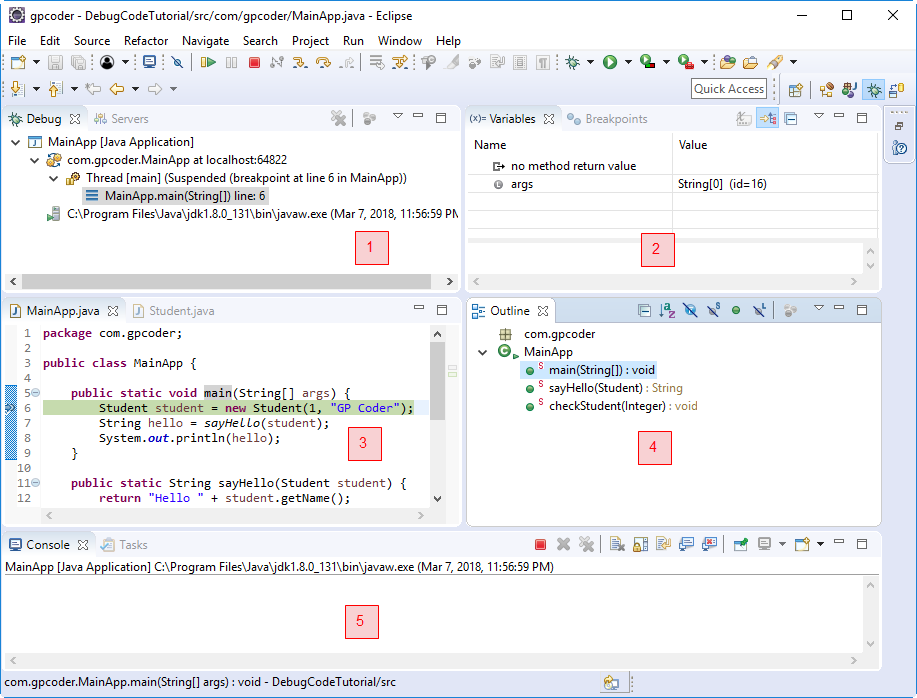
Có thể thực hiện một trong các cách bên dưới để Start ở project ở chế độ debug trong Eclipse:

* Chọn menu Run -> Debug As -> Java Application.
* Click chuột phải vào class chứa phương thức main() -> Debug As -> Java Application.
* Nhấn phím F11.
* Click vào button Debug trong thanh công cụ -> Debug As -> Java Application



Lần đầu tiên debug hộp thoại sau được hiển thị ra, check vào “**Remember my decision**” nếu bạn không muốn nhìn thấy nó lần nữa -> chọn **Yes**.

Nếu khung làm việc Debug không được mở, bạn có thể vào Window -> Perspective -> Open Perspective -> Others -> Debug.



Khung nhìn BreakPoint gồm có 5 cửa sổ nhỏ:

* Cửa sổ 1 (Debug): hiển thị thông tin Class của dòng lệnh đang thực thi.
* Cửa sổ 2 (Variables/ BreakPoints, Expressions): hiển thị giá trị các biến, hiển thị tất cả các BreakPoint đánh dấu của project, thiết lập các biểu thức (expression), điều kiện để đánh dấu BreakPoint.
* Cửa sổ 3 (Java Class): cho phép thực hiện điều khiển thực hiện debug.
* Cửa sổ 4 (Outline): hiển thị tên các biến, phương thức của class hiện đang thực thi, bạn có thể di chuyển nhanh đến vị trí của phương thức bằng cách click lên tên phương thức tương ứng.
* Cửa sổ 5 (Console): hiển thị thông tin kết quả của câu lệnh System.out, System.error, … ra cửa sổ Console.

### Sử dụng các phím tắt để theo dõi giá trị các biến, theo dõi flow thực thi của chương trình.

1. Các phím tắt để Debug Java project trong Eclipse

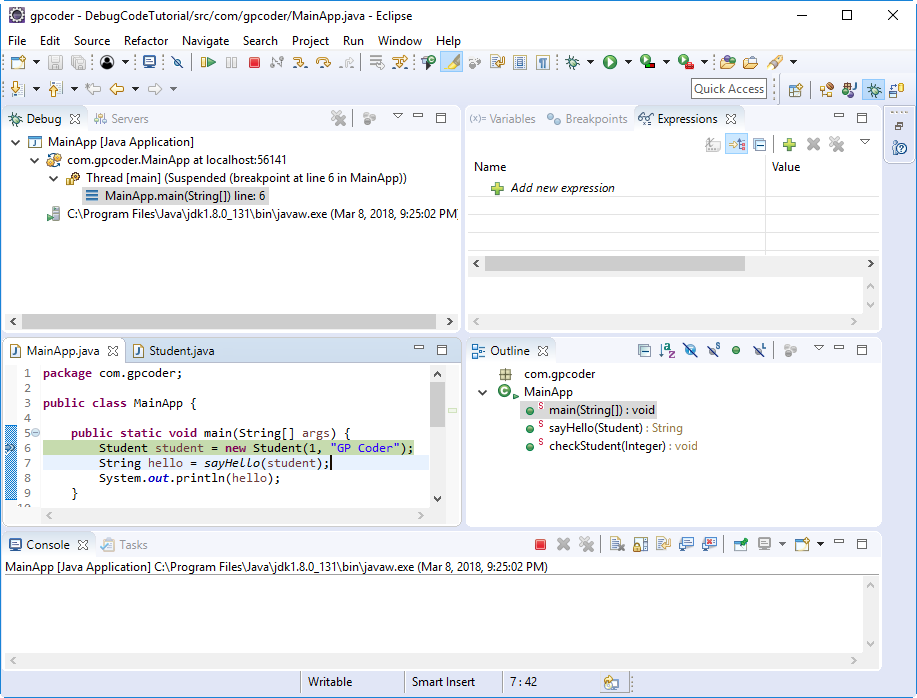


Eclipse cung cấp các nút trên thanh công cụ để kiểm soát việc thực hiện chương trình mà bạn đang debug. Bạn có thể sử dụng cho phép sử dụng phím tắt tương ứng để kiểm soát việc thực hiện này.

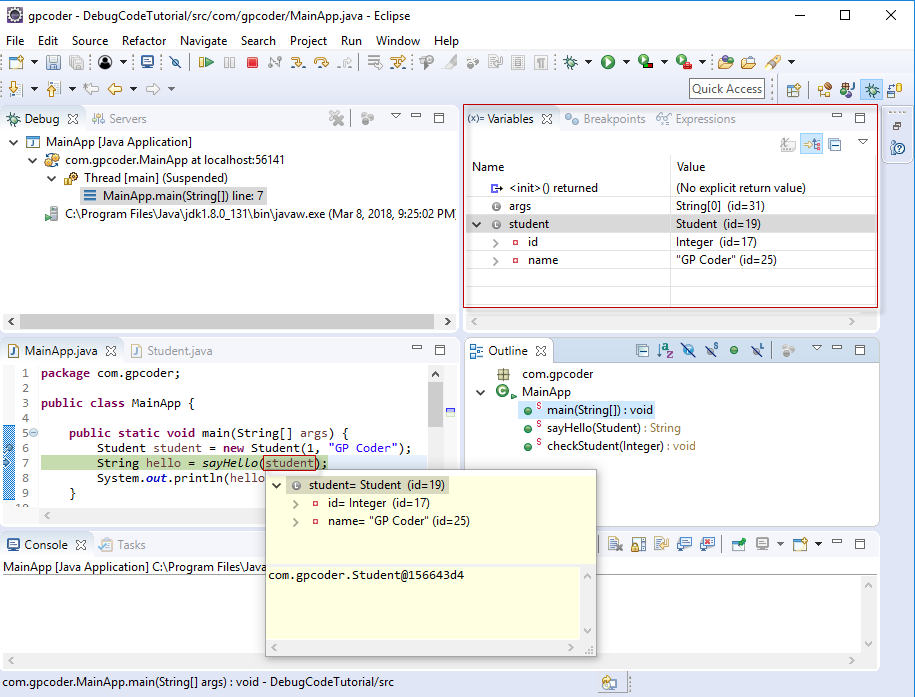
* **F5** (Step Into): nhảy vào phương thức đang debug.
* **F6** (Step Over): thực thi câu lệnh hiện tại và nhảy đến câu lệnh tiếp tiếp.
* **F7** (Step Return): thực thi câu lệnh hiện tại và quay lại lệnh đã gọi phương thức này.
* **F8** (Resume): thực thi câu lệnh hiện tại và nhảy đến breakpoint tiếp theo.
* **Ctrl** + **F2** (Terminate): kết thúc chế độ debug.
* **Ctrl**+ **Shift** + **B**: đánh dấu/ hủy BreakPoint
* **Ctrl** + **Shift** + **I**: mở popup hiển thị thông tin giá trị biến được chọn.

1. Xem các giá trị biến trong khung debugger

Sau khi start chương trình ở chế độ debug, chương trình sẽ thực thi bình thường. Nếu gặp BreakPoint, chương trình sẽ dừng lại để cho phép bạn điều khiển và theo dõi. Như bạn thấy, chương trình sẽ dừng lại ở BreakPoint tại dòng code số **6**:



Nhấn **F6** để thực thi câu lệnh hiện tại và trỏ tới câu lệnh tiếp theo.

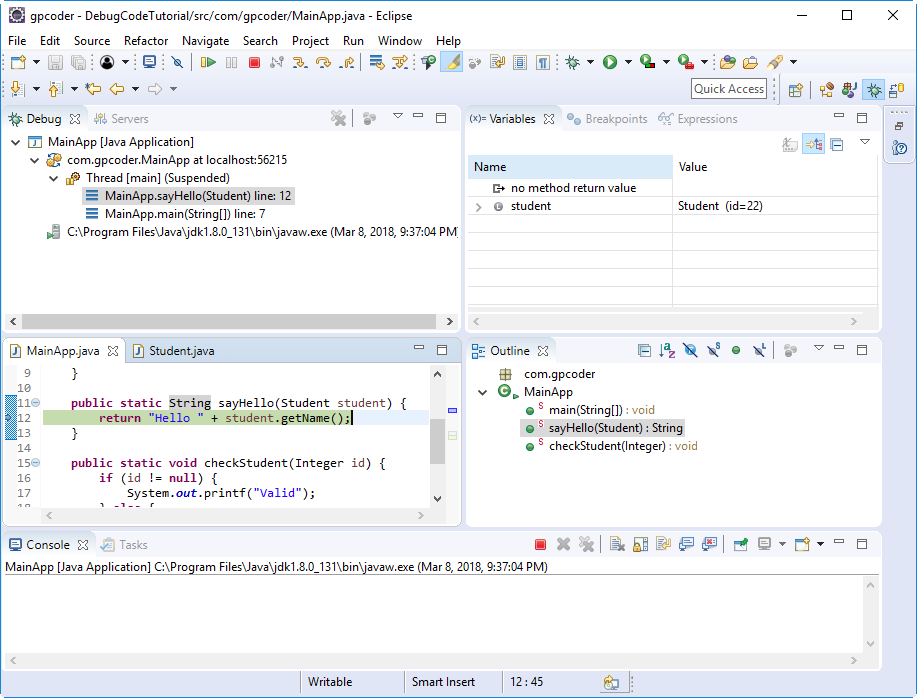


Để xem giá trị của biến, bạn thực hiện một trong các cách sau:

* Hover chột vào biến tương ứng.
* Bôi đen biến và nhấn tổ hợp phím **Ctrl + Shift + I.**
* Mở cửa sổ Variable.

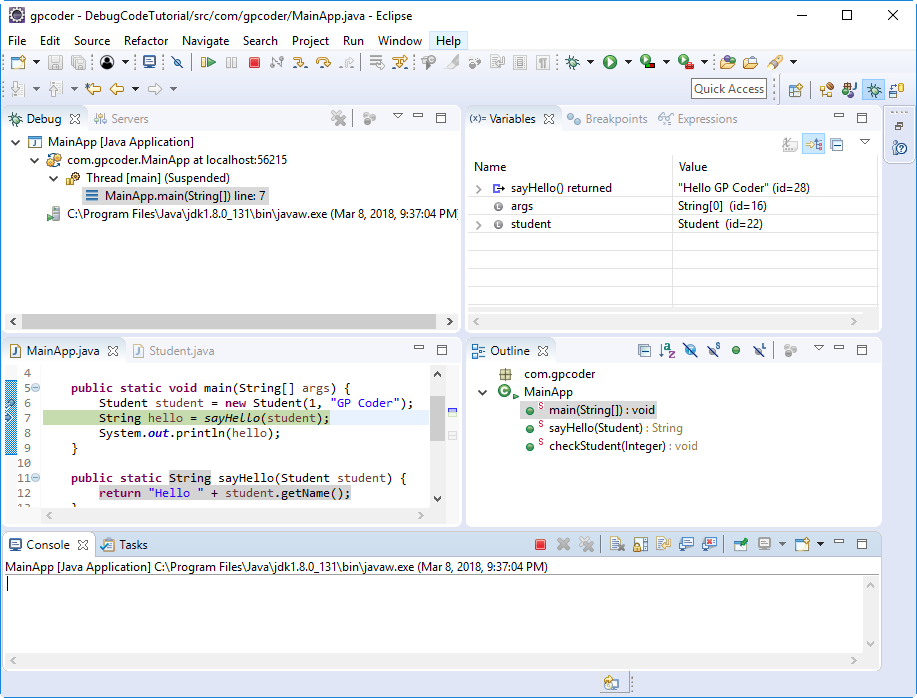
Như bạn thấy hình trên, chương trình dừng tại dòng code số 7, khi bạn hover chuột lên biến student, bạn có thể thấy được giá trị của biến student.

Tại dòng code số 7, bạn có thể nhấn phím **F5** để vào xem nội dụng của phương thức sayHello.

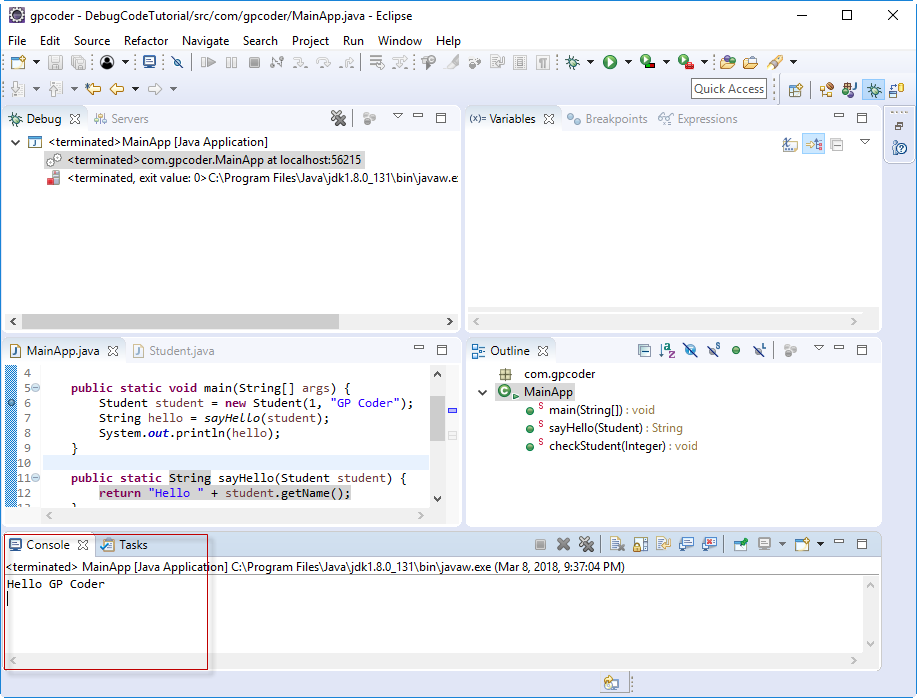


Như bạn thấy, hiện tại chương trình đã nhảy vào phương thức sayHello(). Tại đây, bạn có thể nhấn phím **F6**để chuyển sang dòng code kế tiếp, khi thực thi đến hết phương thức sayHello(), chương trình sẽ quay lại dòng code số 7.

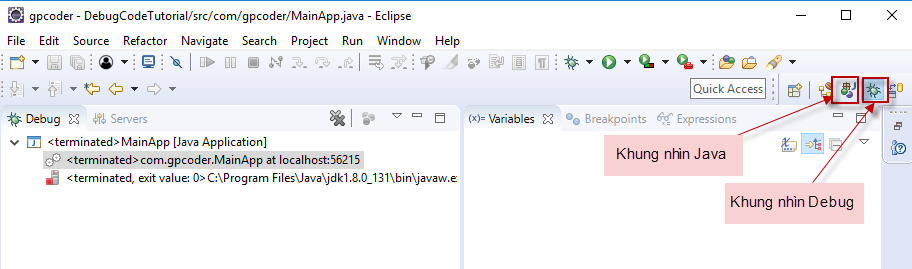
Để quay lại dòng code số 7 mà không cần thực thi từng dòng lệnh (không nhấn phím **F6**), chúng ta sẽ nhấn phím **F7**, khi đó chương trình sẽ thực thi tất cả các câu lệnh còn lại phương thức sayHello() và quay lại lệnh đã gọi phương thức này (tức là dòng code số 7).



Nhấn **F8** để thực thi câu lệnh hiện tại và nhảy đến **BreakPoint** tiếp theo (nếu có). Trong ví dụ của chúng ta là kết thúc chương trình.



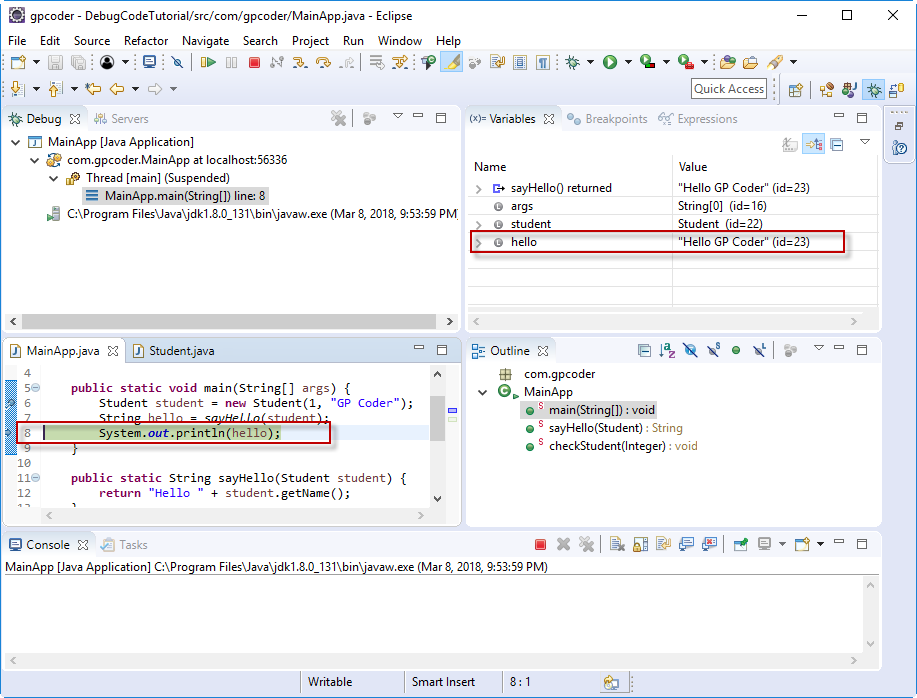
**Lưu ý:** sau khi thực hiện debug xong, bạn có thể chuyển lại khung nhìn Java bằng cách click vào icon bên phải như sau:



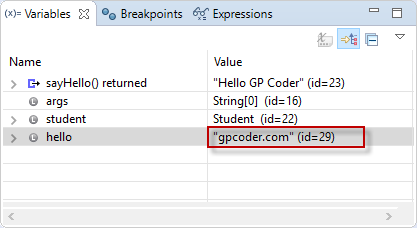
Nếu không có các icon trên, bạn có thể vào menu chính -> **Window** -> **Perspective** -> **Open Perspective** -> **Other** -> chọn **Debug** hoặc **Java** hoặc **Java EE**.

1. Thay đổi/ gán giá trị biến trong lúc debug

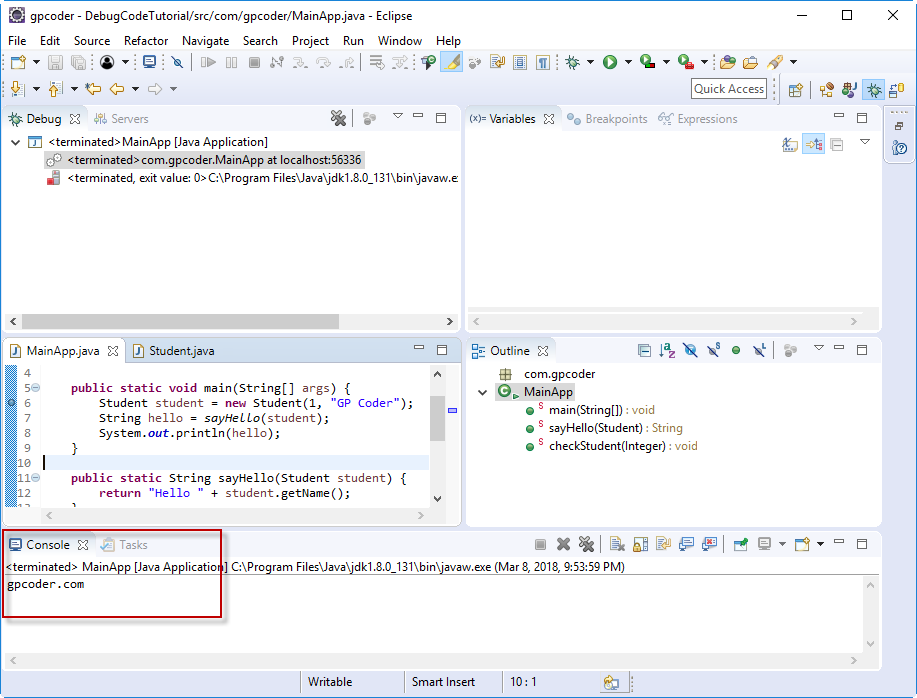
Ví dụ thực hiện debug chương trình trên đến dòng code số 8, khi đó giá trị hello tại cửa sổ **Variables** của khung nhìn **Debug** như sau:



Chúng ta có thể thay đổi giá trị của **biến hello** bằng cách nhấp chuột vào **value** của biến hello, sau đó nhập giá trị mới. Ví dụ, tôi đổi giá trị “**GP Coder**” thành “**gpcoder.com**“:



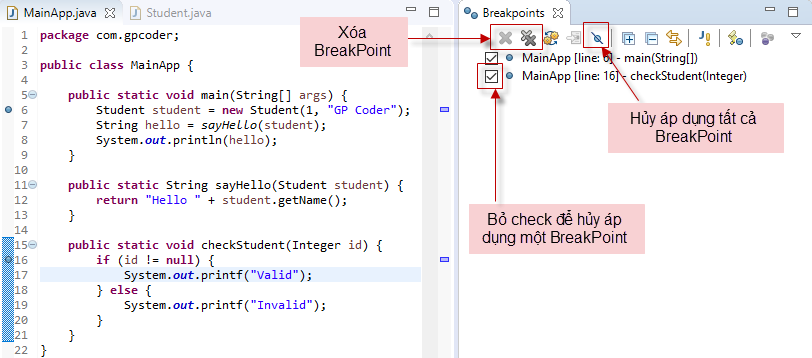
Nhấn phím **F8** để chương trình thực thi tiếp phần còn lại. Tại cửa sổ **Console**, bạn sẽ thấy giá trị “gpcoder.com” được hiển thị:



1. Xoá các BreakPoint hoặc huỷ kích hoạt BreakPoint

Khung nhìn Breakpoints cho phép bạn xóa và hủy kích hoạt các BreakPoint. Để hủy kích hoạt một điểm ngắt, loại bỏ hộp kiểm tương ứng trong khung nhìn Breakpoints. Để xóa nó, bạn có thể sử dụng các nút tương ứng trong thanh công cụ.

Mở khung nhìn BreakPoint: Trên menu chính, chọn Window -> Show view -> Others -> BreakPoints

[](https://gpcoder.com/wp-content/uploads/2018/03/breakpoint-perspective.png)

Như bạn thấy khung nhìn BreakPoint sẽ hiển thị tất cả các BreakPoint được đánh dấu trong chương trình của bạn. Trong hình trên, tôi đã đánh dấu 2 BreakPoint tại dòng code 6 và 16, khung nhìn view cũng thể hiện 2 BreakPoint này.

Trên đây là những hướng dẫn cơ bản nhất để thực hiện debug trong Java. Thao tác debug khá đơn giản nhưng rất quan trọng, mỗi lập trình viên đều cần biết và vận dụng nó để dễ dàng dò tìm lỗi, cũng như hiểu sâu hơn về cách hoạt động của chương trình. Tôi xin dừng bài viết ở đây, cám ơn các bạn đã quan tâm và theo dõi. Hẹn gặp lại ở các bài viết tiếp theo.

* 1. **Debug như thế nào mới hiệu quả?**

Nếu ta chịu khó debug chi tiết thì sẽ hiểu rõ hơn về hệ thống và biết được chương trình đang bị lỗi gì. Tuy nhiên, đôi khi chúng ta không có đủ thời gian để debug chi tiết từng dòng code. Dưới đây là một vài điểm giúp quá trình debug nhanh hơn, dễ phát hiện lỗi xảy ra hơn mà tôi muốn chia sẽ với các bạn:

* Đọc cẩn thận các dòng thông báo lỗi khi chạy đoạn mã, đôi khi những dòng báo lỗi đã chỉ ra cho ta biết nguyên nhân lỗi là gì.
* Khi ta chạy chương trình debug ngoài xem các giá trị của biến cần ta cần xem thêm các comment để hiểu hơn nghiệp vụ xử lý trong chương trình. Từ đó có thể phán đoán được khả năng xảy ra lỗi trong chương trình nhanh hơn.
* Phân vùng lỗi thông qua các BreakPoint. Nếu ta chạy từng dòng lệnh một cách chi tiết sẽ tốn nhiều thời gian. Thường thì tôi đoán lỗi khả năng xảy ra ở đâu, sau đó đặt các BreakPoint ở đó. Chạy debug để loại bỏ các dự đoán khả năng lỗi. Rồi phân vùng khả năng lỗi để tiến hành debug sâu dưới dạng Step into để gỡ lỗi. Theo kinh nghiệm của tôi thì làm cách này tôi thấy phát hiện lỗi được nhanh hơn. Nhưng nhanh hay chậm cũng phụ thuộc kinh nghiệm, nếu kinh nghiệm tốt sẽ đoán được phân vùng lỗi nhanh và đỡ tốn thời gian hơn.