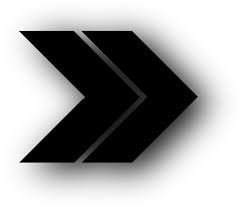
**LỘ TRÌNH ĐÀO TẠO CHƯƠNG 4 PHẦN 2**

1. Giới thiệu

- Trong quá trình làm việc do nhu cầu cần thiết, người lập trình viên cần phải thực hiện chạy nhiều  
lần những đoạn mã code giống nhau.  
Vd1: Viết chương trình in ra màn hình 10 câu chào: Hello World  
Vd2: Viết chương trình in ra màn hình 1000 câu chào: Hello World  
Có 2 giải pháp để thực hiện:  
- Giải pháp 1: Viết 1 đoạn mã Hello World ra và liên tục copy paste cho đến khi đủ 10, 100 …

Sẽ không ai rãnh mà làm như vậy cả.

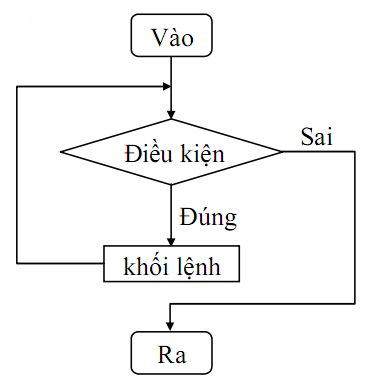
 Giải pháp 2: Dùng câu lệnh lặp  
- Câu lệnh lặp sẽ liên tục lặp đi lặp lại một hành động nào đó nếu như nó còn thỏa điều kiện gì đó, chỉ khi nào không còn thỏa được điều kiện thì nó mới dừng lại, lúc này người ta gọi nó là “điều kiện dừng vòng lặp”.  
Bản chất bên trong 1 câu lệnh lặp luôn tồn tại 4 thành phần chính như sau:

1. Khởi tạo ban đầu
2. Điều kiện để được lặp
3. Hành động sẽ thực hiện
4. Quá trình lặp

Vd: Cuộc sống của con người chính là 1 vòng lặp, trong đó:  
1. Khởi tạo ban đầu => Sinh ra  
2. Điều kiện để được lặp => Còn sống  
3. Hành động sẽ thực hiện => Hít thở không khí, trao đổi chất  
4. Quá trình lặp => Mỗi giây, mỗi phút trôi qua

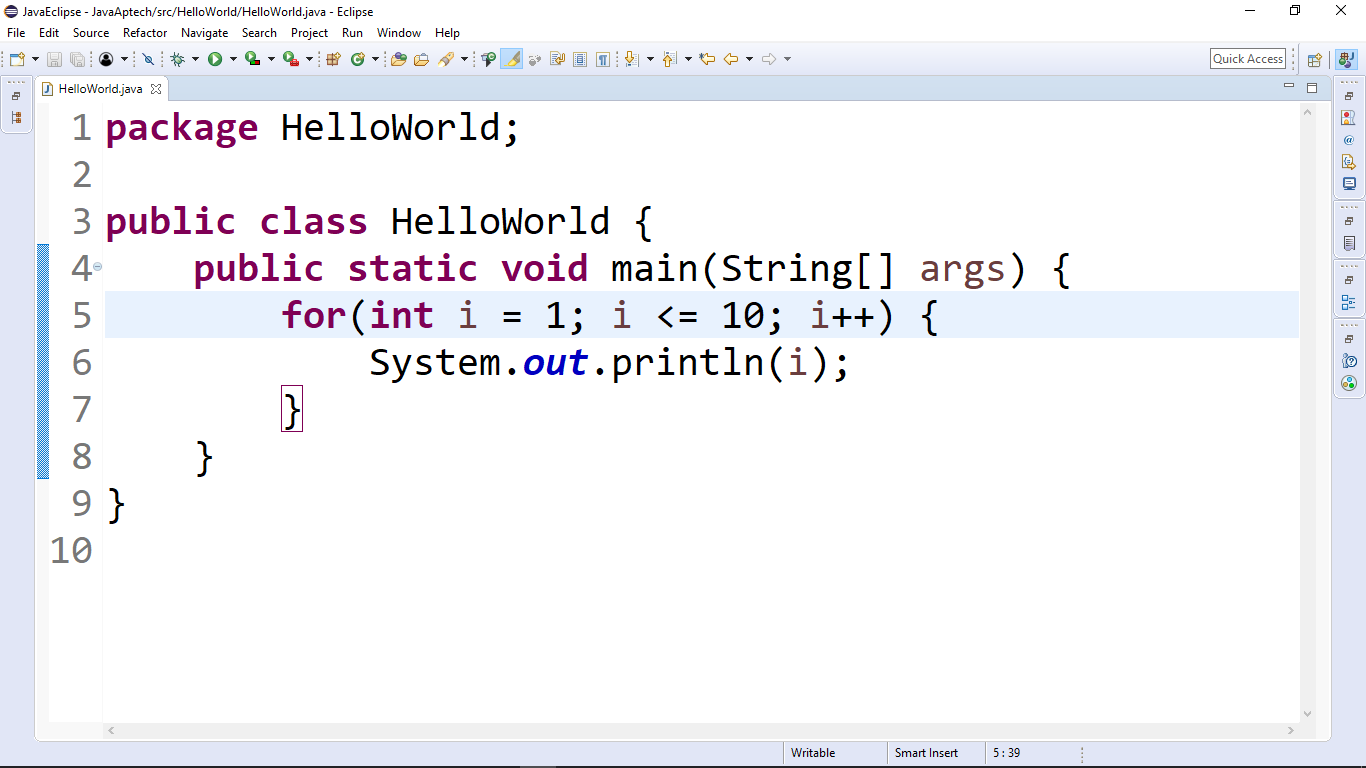
Trong Java có 4 câu lệnh lặp: for, while, do – while, for – each.

1. Câu lệnh lặp
   1. Câu lệnh lặp for

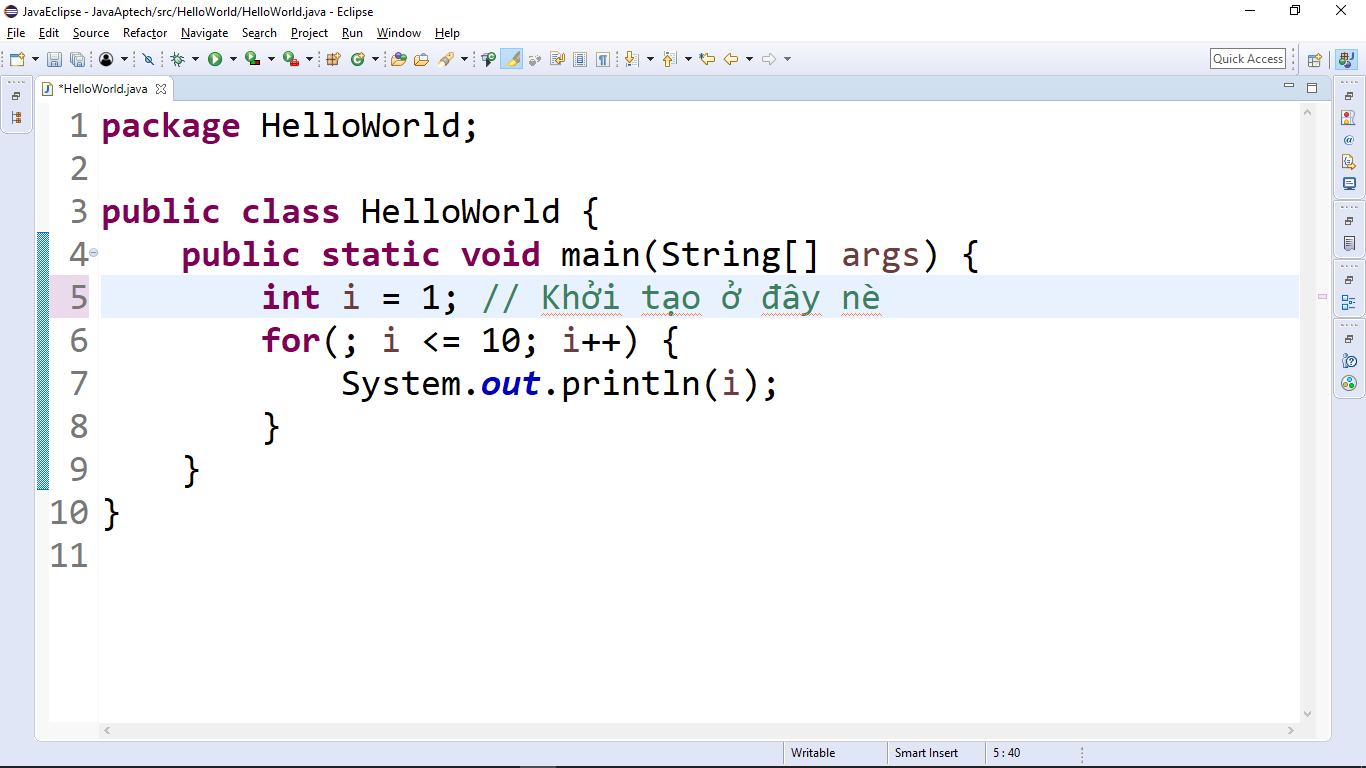
5

**for**(<Khởi Tạo>; <Điều Kiện Lặp>; <Bước Lặp>)  
<Hành Động>;  
Trong đó <Hành Động> có thể là “câu lệnh đơn” (chỉ có duy nhất 1 câu lệnh) hoặc là “câu lệnh phức” (có từ 2 câu lệnh trở lên).  
Nếu là “câu lệnh phức” thì phải có dấu ngoặc nhọn {} kèm theo bọc câu lệnh đó bên trong. Nếu là “câu lệnh đơn” thì không cần cũng được.

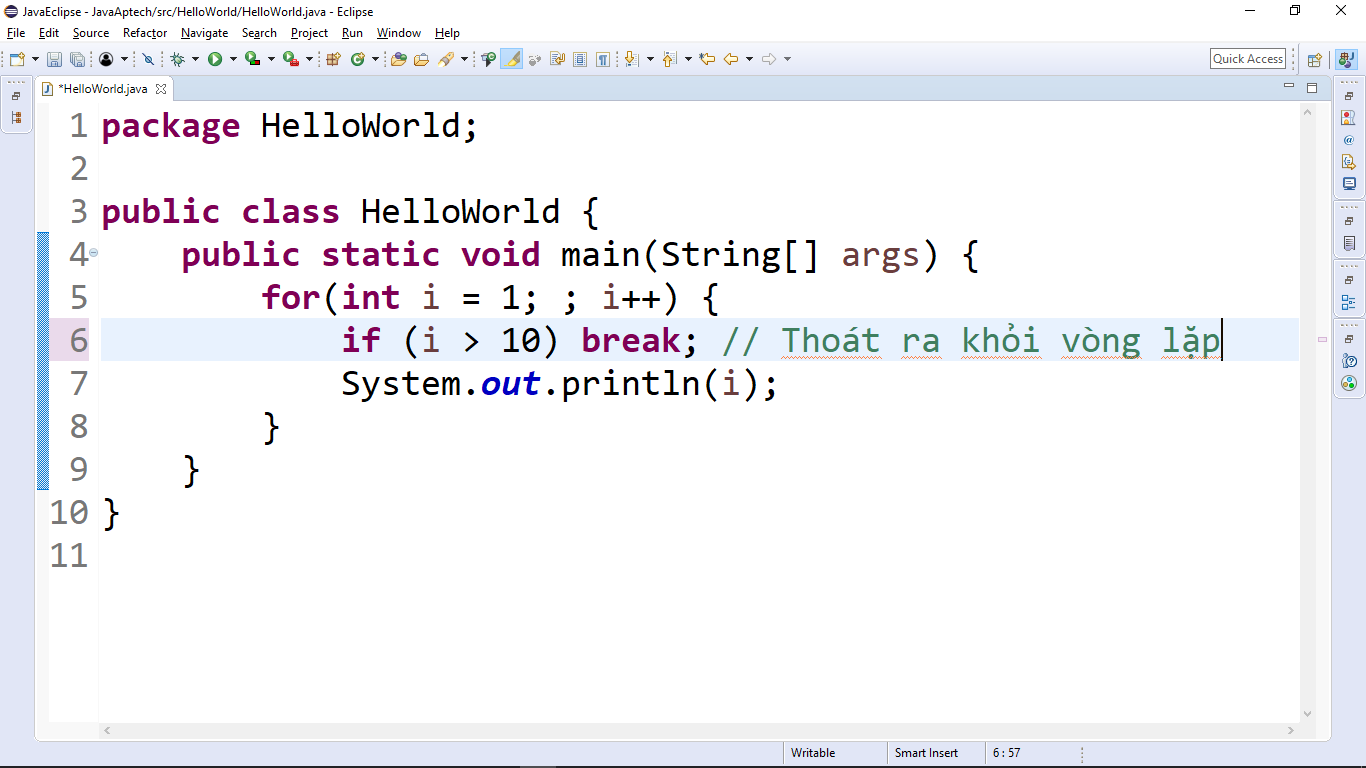
VD: Viết chương trình in ra từ 1 đến 10.



-Bản chất trong câu lệnh lặp for có thể sẽ không có phần “Khởi Tạo”. Thực ra là nó khởi tạo ở 1 chỗ khác chứ không khởi tạo ngay bên trong vòng for.

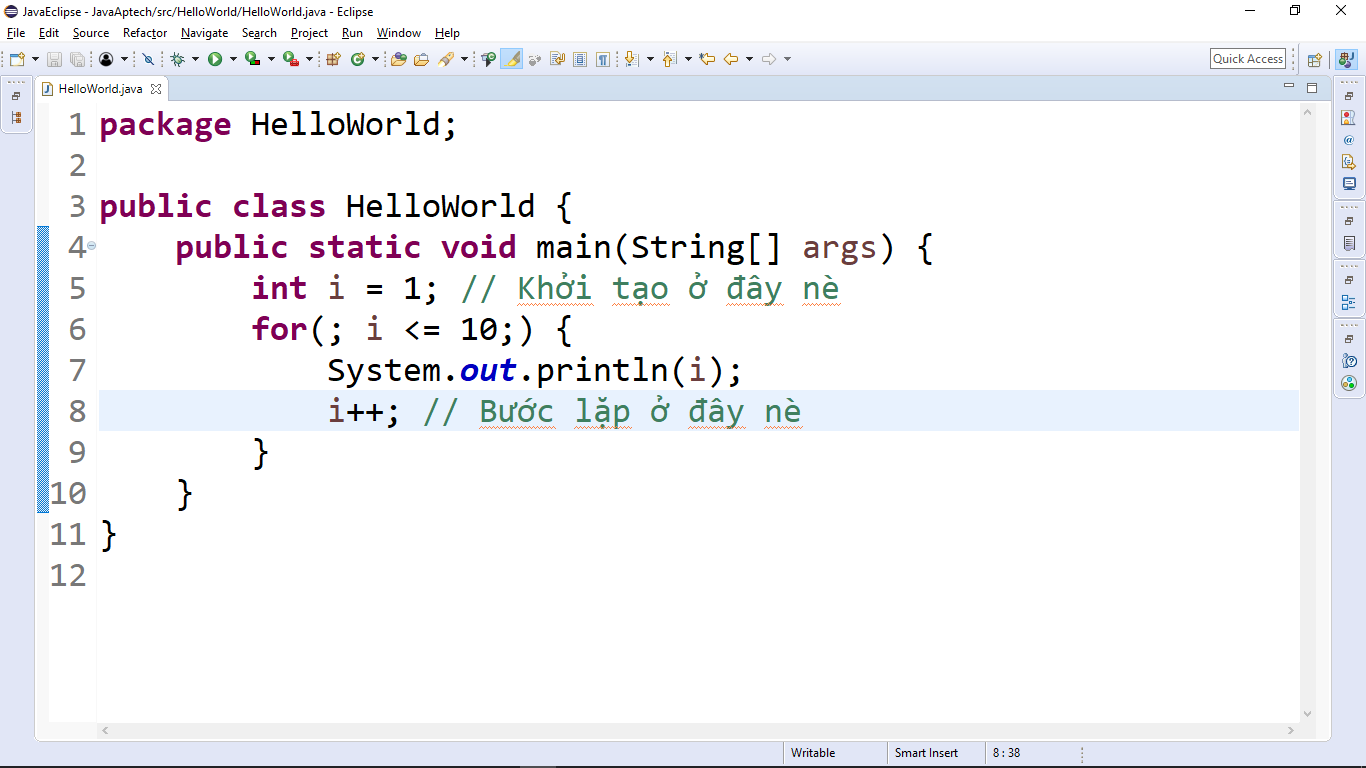


Bản chất trong câu lệnh lặp for có thể sẽ không có phần “Điều Kiện Lặp”. Chúng ta hiểu  
“Điều Kiện Lặp” chính là để biết được khi nào vòng lặp sẽ dừng lại. Vậy trong tình huống  
này ta sử dụng 1 lệnh để thoát ra khỏi vòng lặp đó là câu lệnh **break**



* Các bạn thử suy nghĩ nếu hàm in ra biến i được đặt phía trên câu lệnh if thì phải sửa lại như thế nào để in ra từ 1 đến 10.

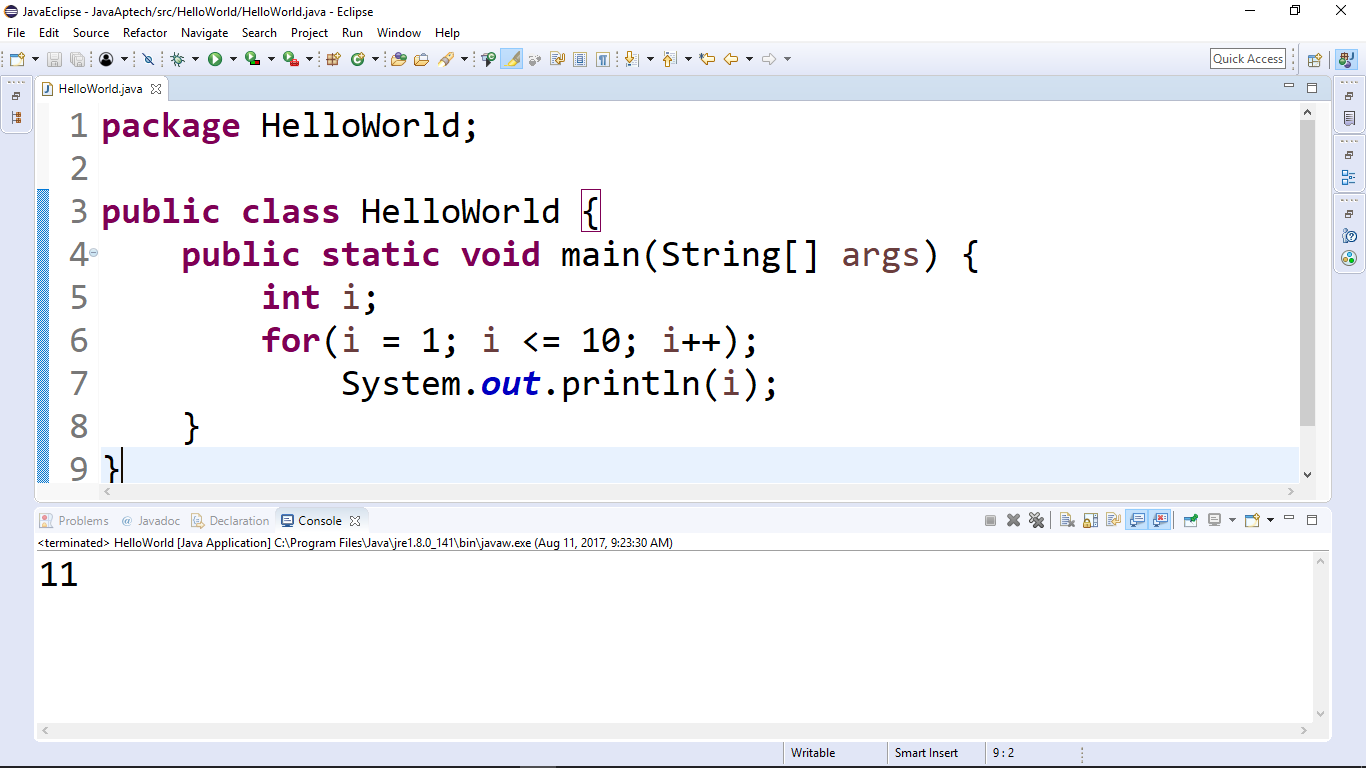
Bản chất trong 1 câu lệnh lặp for có thể sẽ không có phần <Bước Lặp>. Thực ra nó thực hiện  
<Bước Lặp> ở ngay bên trong thân vòng lặp.



Tóm lại ta hiểu như sau:  
- Bất kỳ câu lệnh lặp nào cũng luôn hội tụ đủ 3 yếu tố: <Khởi Tạo>, <Điều Kiện Lặp>, <Bước  
Lặp>. Nếu không có chẳng qua là nó sẽ nằm ở một chỗ khác.  
- Nếu như không có phần nào thì ta vẫn phải khai báo ra dấu ; tương ứng phần đó, nếu không sẽ  
bị **báo lỗi cú pháp**.

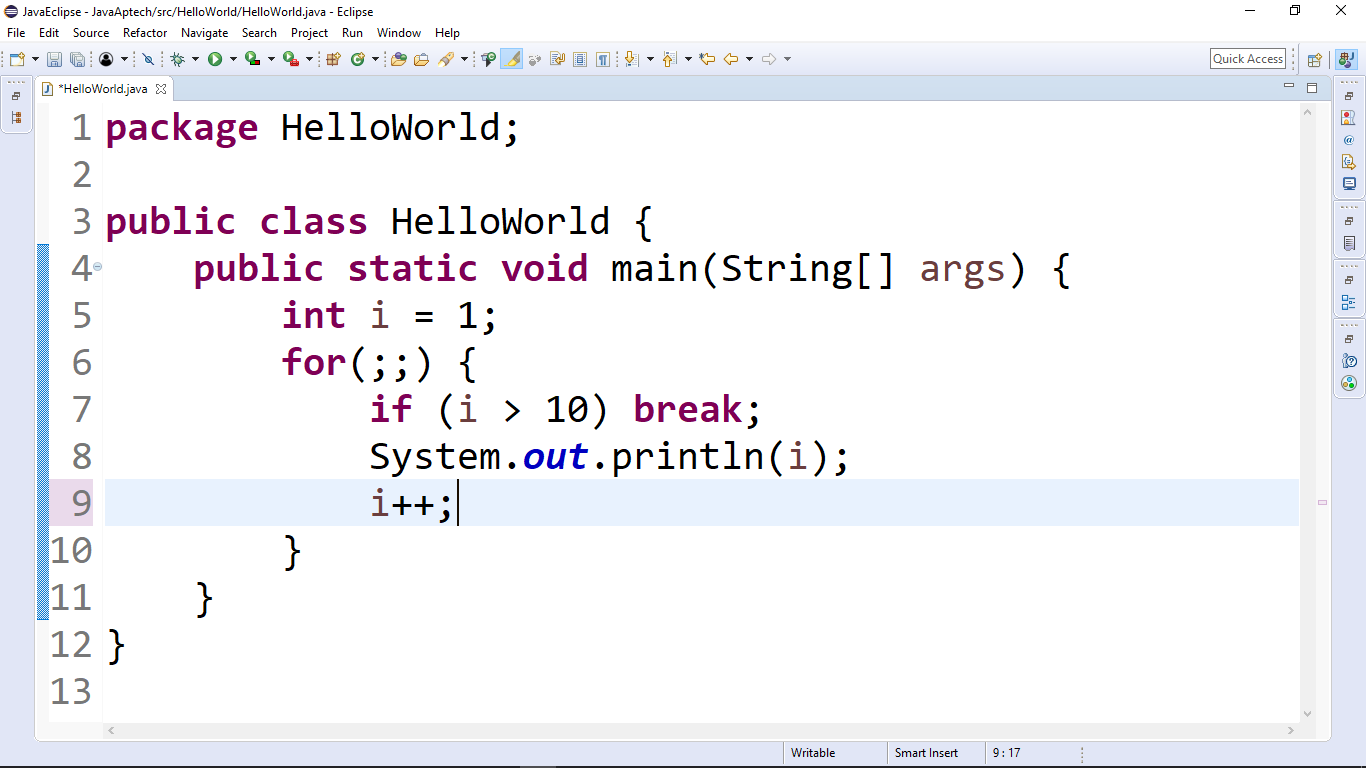
* Một số lưu ý:

Ta không được thêm ; đằng ngay sau câu lệnh lặp for vì như thế chương trình sẽ hiểu  
câu lệnh for đó bị **rỗng**



- Vòng lặp bị rỗng tức là nếu như trong bên trong thân của nó ta có cài đặt các câu lệnh thì lúc này nó sẽ hiểu là những câu lệnh đó nằm ở bên ngoài, tức là theo tiến trình thì vòng lặp sẽ lặp liên tục với hành động là hành động rỗng (Tức là không làm gì cả) rồi sau đó khi vòng lặp đã dừng lại rồi mới thực hiện các câu lệnh ban đầu.

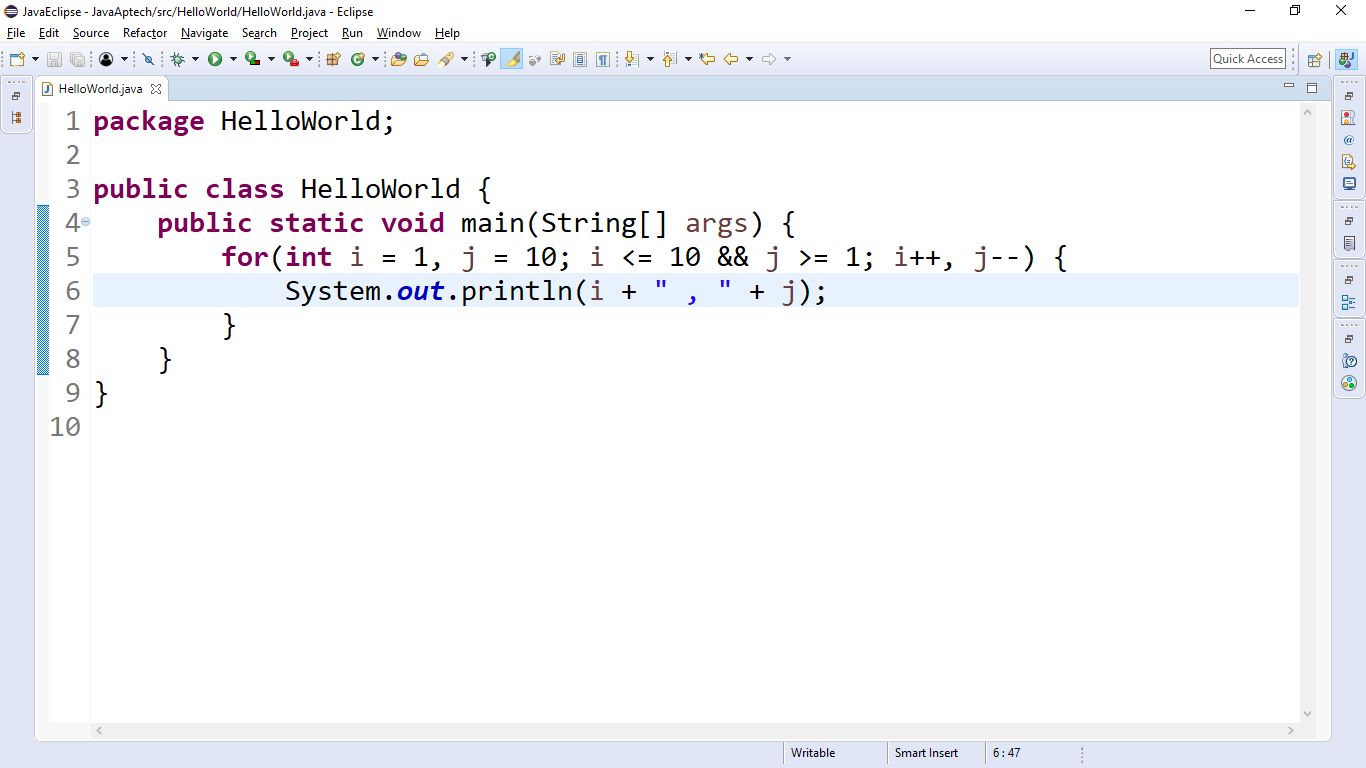
VD: Trong vòng lặp for không có phần tử nào:



* Một số lưu ý:

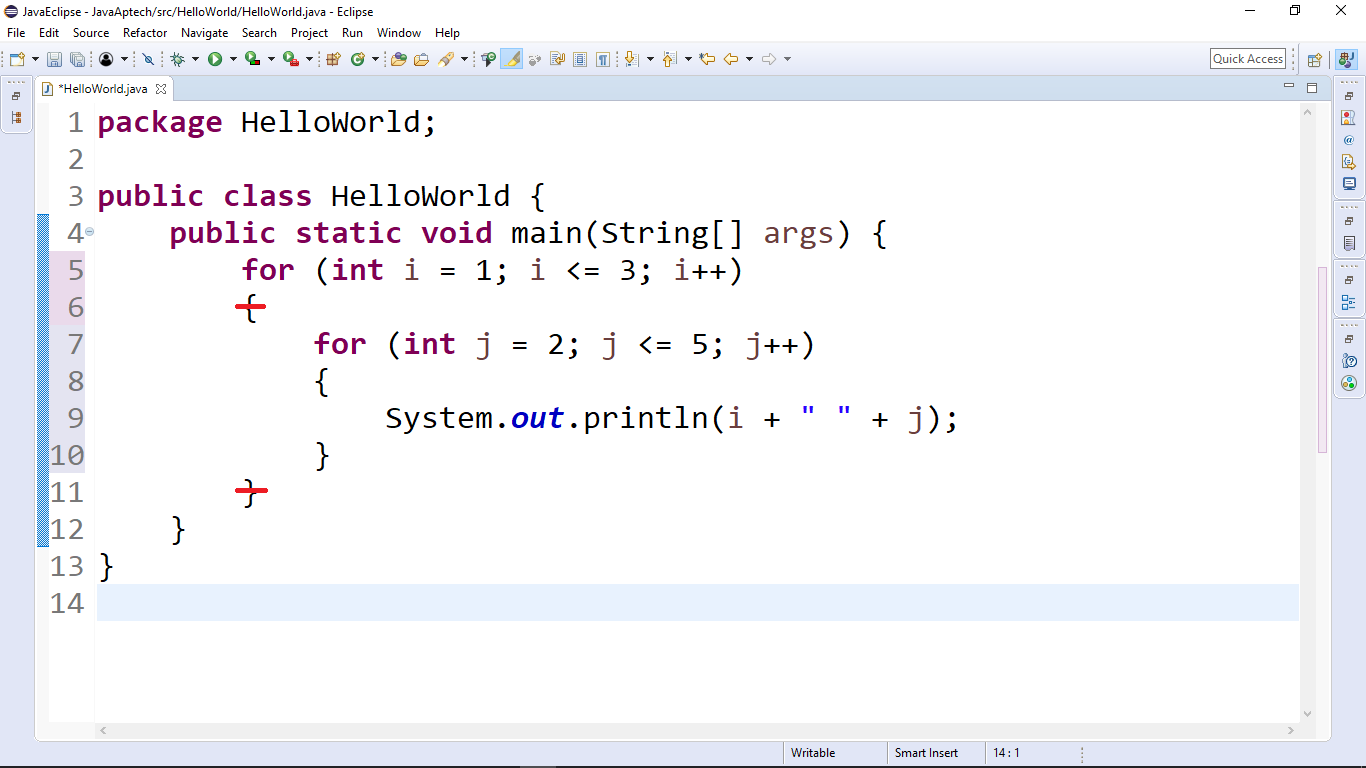
-Ta có thể có nhiều <Khởi Tạo>, có thể có nhiều <Điều Kiện Lặp> và có nhiều <Bước Lặp>. Mỗi phần như thế ta sẽ ngăn cách với nhau bởi dấu “,”

- Ngoài ra riêng phần <Điều Kiện Lặp> do bản thân nó là phép luận lý cho nên ta có thể sử dụng các toán tử kết hợp, ví dụ ta có thể sử dụng &&.



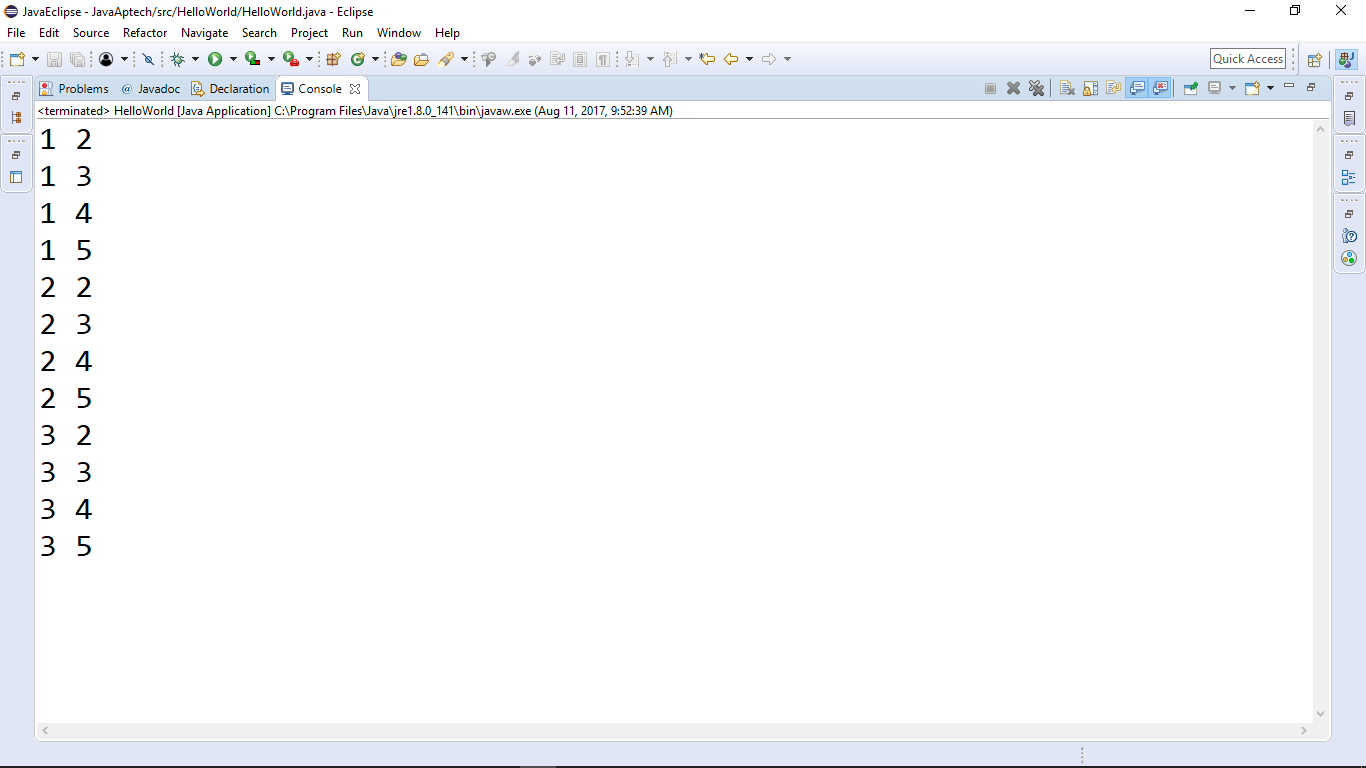
* Vấn đề câu lệnh lồng nhau

-Bản chất cả câu lệnh for cũng chỉ là 1 câu lệnh đơn nên nếu như nó có lồng vào nhau thì ta cũng  
không cần để bên trong dấu ngoặc nhọn {}



Và khi có 2, 3 hoặc nhiều vòng lặp lồng vào nhau ta phải hiểu quá trình chạy của nó như sau:  
- Vòng lặp nằm bên ngoài ta gọi nó là vòng lặp **cha**- Vòng lặp nằm bên trong ta gọi nó là vòng lặp **con  
=> Với mỗi trường hợp của vòng lặp cha nó sẽ chạy hết mọi trường hợp của vòng lặp con.**

**Theo ví dụ trên ta có được kết quả sau:**



* Vòng lặp vô tận
* Trong quá trình lập trình, nếu không cẩn thận người lập trình viên có thể sẽ tạo ra những **vòng lặp vô tận**
* Điều kiện lặp lun đúng, hoặc để trống tất cả trong for.

TH1:

for (int I = 1; i > 0; i++) System.out.println(“Duy đẹp trai !!”);

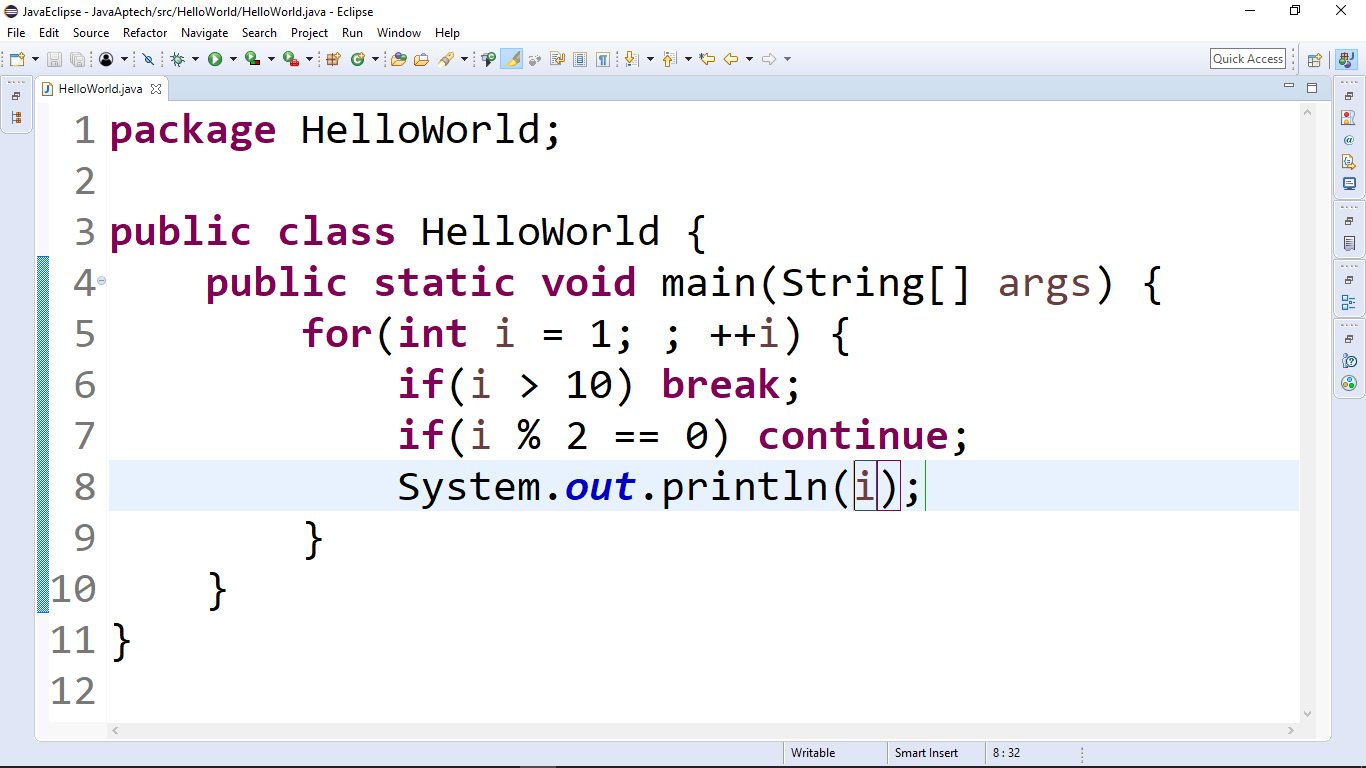
TH2:

for (; ;) System.out.println(“Duy đẹp trai !!”);

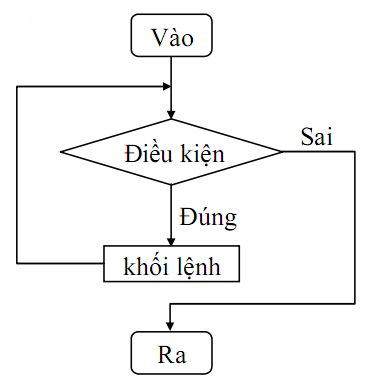
* Hoặc đôi lúc người lập trình viên lợi dụng vào **vòng lặp vô tận** để giải quyết các bài toán khi mà **điều kiện để được lặp xét phức tạp** trong khi **điều kiện dừng thì xét đơn giản hơn**, họ sẽ cho vòng lặp vô tận chạy và kèm theo bên trong là **điều kiện để dừng lại.**
* Từ khóa break và continue

Khi thao tác với câu lệnh lặp thì 2 câu lệnh mà ta thường xuyên sẽ sử dụng đó là **break** và **continue**.  
- Từ khóa **break**: Có chức năng thoát ra khỏi vòng lặp đang chứa nó. Lưu ý: **Chỉ có thể thoát ra khỏi duy nhất 1 vòng lặp chứa nó.**  
- Từ khóa **continue**: Có chức năng bỏ qua lần lặp hiện tại, xét sang lần lặp mới.

VD: Viết chương trình in ra các số lẻ có trong khoảng từ 1 đến 10.

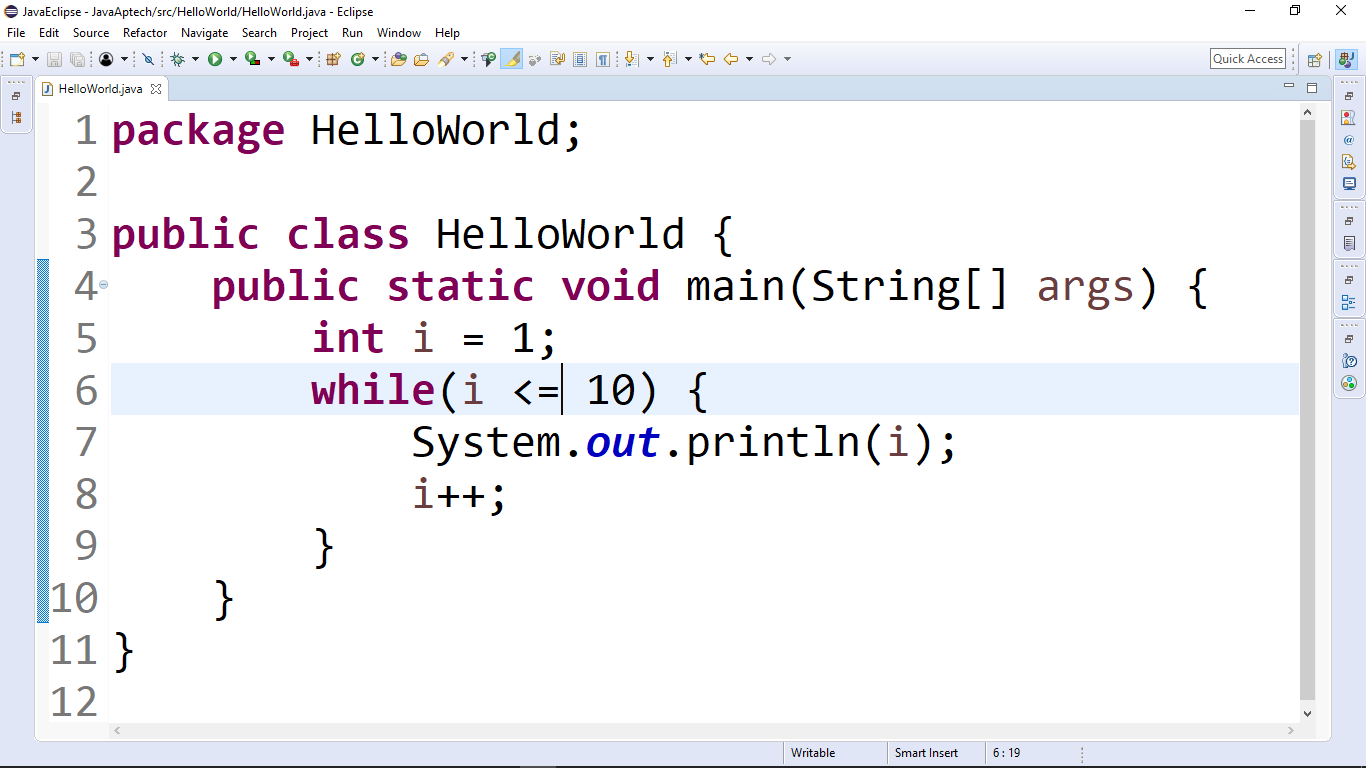


* 1. Câu lệnh lặp while

<Khởi tạo>  
**while**(<Điều Kiện Lặp>)  
<Hành Động>

Trong đó <Hành Động> ở đây bao gồm là hành động nó thực hiện và bước lặp của nó. Có thể là “câu lệnh đơn” (chỉ có duy nhất 1 câu lệnh) hoặc là “câu lệnh phức” (có từ 2 câu lệnh trở lên).  
Nếu là “câu lệnh phức” thì phải có dấu ngoặc nhọn {} kèm theo bọc câu lệnh đó bên trong.  
Nếu là “câu lệnh đơn” thì không cần cũng được.

VD: In ra màn hình từ 1 đến 10 bằng vòng lặp while:

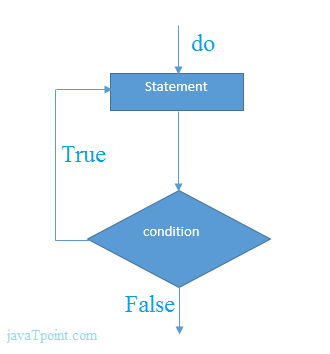


* **Câu lệnh lặp while cũng giống như là câu lệnh lặp for bỏ đi phần <Khởi Tạo>, <Bước  
  Lặp> chỉ còn lại mỗi <Điều Kiện Lặp>**
* **Còn những tính chất khác như vòng lặp lồng nhau, từ khóa break, continue … thì  
  cũng y như vòng lặp for.**
* Vòng lặp vô tận
* Muốn tạo ra vòng lặp vô tận thì điều kiện xét trong while phải cho ra kết quả là **true** (khác 0).

VD: Các vòng lặp vô hạn:

while(true) {}  
while(x) {} với x là số khác 0 (thường chọn số 1)  
while(1 < 2) {}

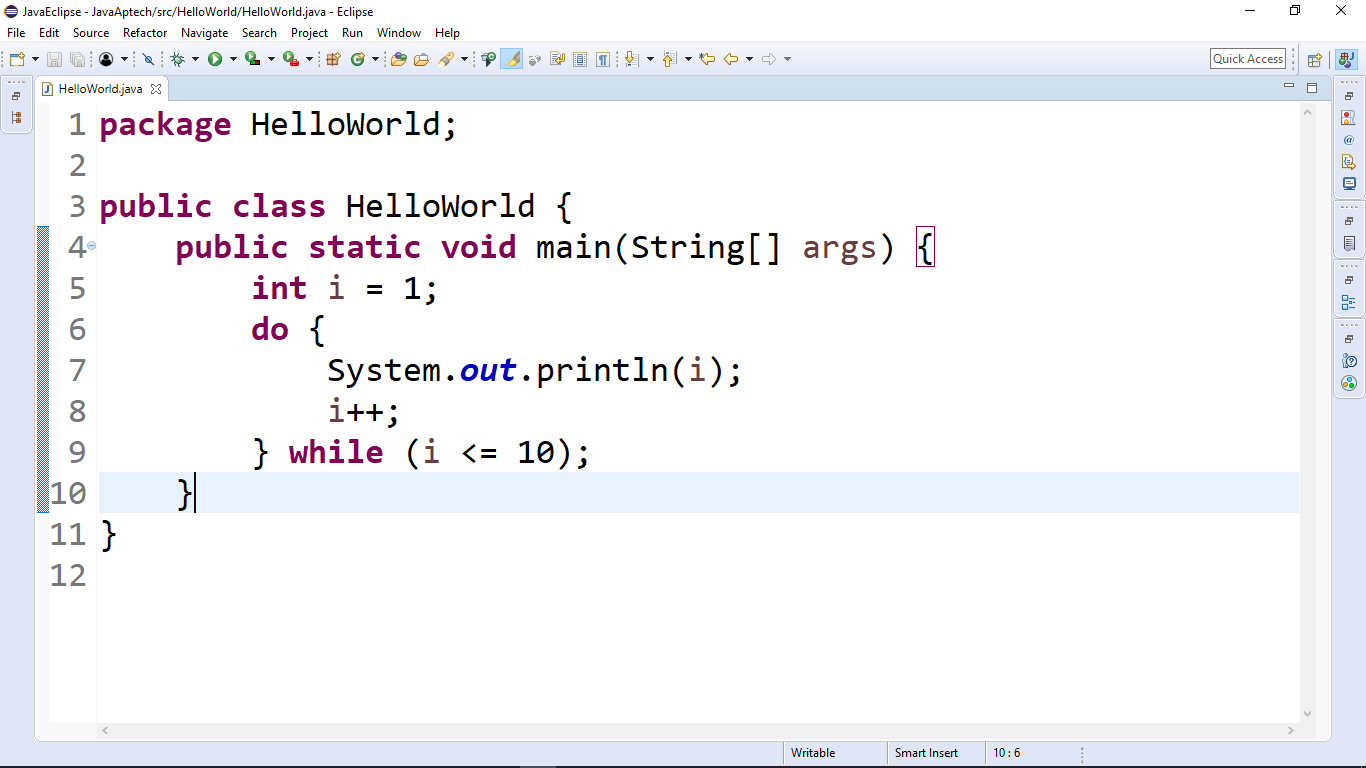
* 1. Câu lệnh lặp do – while

**do**<Hành Động>  
**while**(Điều Kiện Lặp);

Trong đó <Hành Động> ở đây bao gồm là hành động nó thực hiện và bước lặp của nó. Có thể là “câu lệnh đơn” (chỉ có duy nhất 1 câu lệnh) hoặc là “câu lệnh  
phức” (có từ 2 câu lệnh trở lên).

Nếu là “câu lệnh phức” thì phải có dấu ngoặc nhọn {} kèm theo bọc câu lệnh đó bên trong. Nếu là “câu lệnh đơn” thì không cần cũng được.

VD: In ra số từ 1 đến 10:

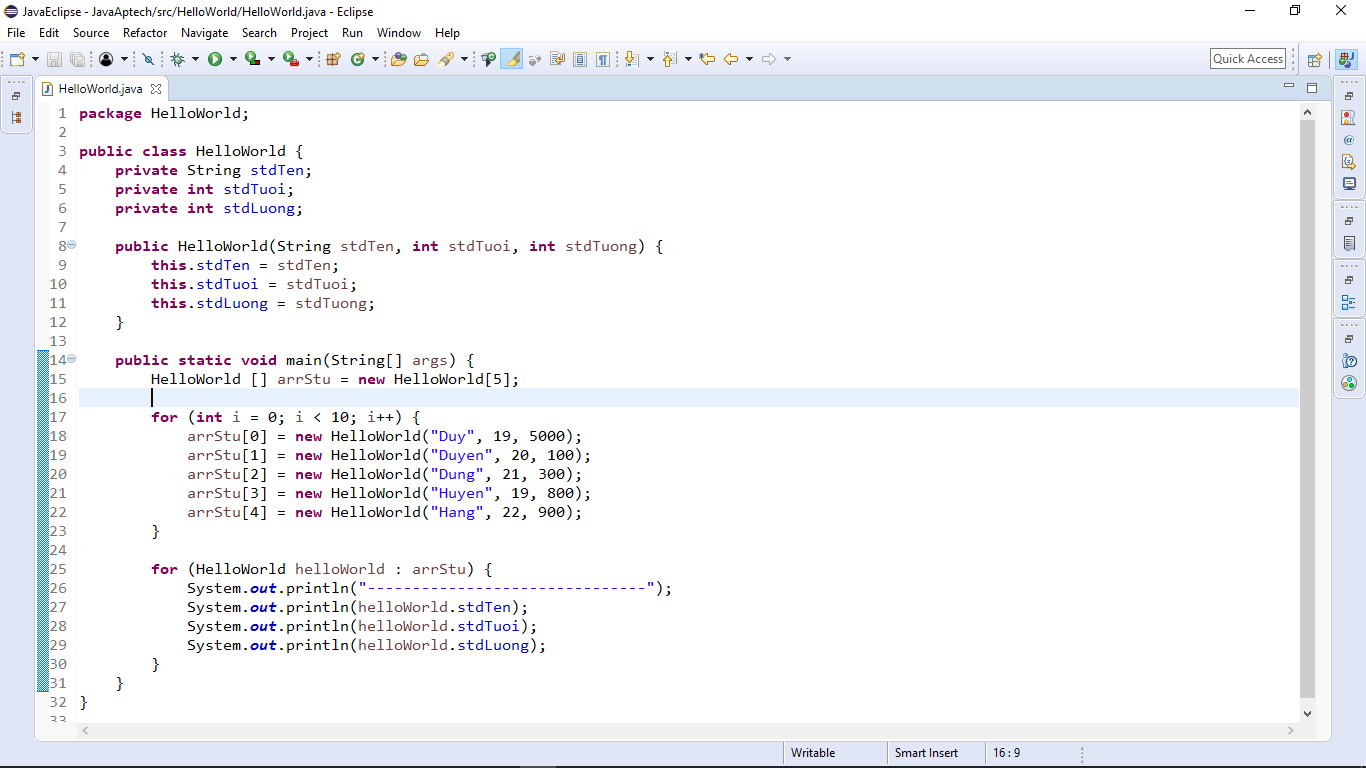


* **Câu lệnh lặp while cũng giống như là câu lệnh lặp for bỏ đi phần <Khởi Tạo>, <Bước  
  Lặp> chỉ còn lại mỗi <Điều Kiện Lặp>**
* **Còn những tính chất khác như vòng lặp lồng nhau, từ khóa break, continue … thì  
  cũng y như vòng lặp for.**
* Vòng lặp vô tận

Tương tự như vòng lặp while.

* 1. Giới thiệu về câu lệnh for – each
* Giống vòng lặp for, thường được dùng để duyệt màng, lấy đối tượng (object), đọc dữ liệu lưu vào mảng.

VD:



1. Chia sẽ kinh nghiệm khi lập trình với câu lệnh lặp

- Điểm khác biệt giữa vòng lặp for, while so với do while đó là vòng lặp for, while đi kiểm tra điều kiện trước tiên, rồi sau đó thỏa rồi thì mới đi thực hiện câu lệnh còn vòng lặp do while thì ngược lại, nó lại đi thực hiện câu lệnh trước xong rồi sau đó mới đi kiểm tra điều kiện thỏa thì lại tiếp tục thực hiện, cho nên ta phải linh động tùy trường hợp cụ thể mà nên biết áp dụng vòng lặp nào.  
**=> for, while có thể sẽ không thực hiện được lần nào nhưng do while luôn thực hiện được ít nhất là 1 lần.**- Khi đề yêu cầu ta nhập một số dữ liệu cho chương trình và kiểm tra nếu dữ liệu không hợp lệ thì sẽ bắt người dùng nhập lại, ta nhận thấy là yêu cầu dữ liệu phải được nhập trước thì sau đó mới có dữ liệu để mà đi kiểm tra  
=> Hành động làm trước sau đó mới kiểm tra => áp dụng do while  
- Vd: Nhập số nguyên n, yêu cầu n phải là số dương, nếu sai thì bắt nhập lại:

