**I.Sự khác nhau giữa int và Interger**

- Cả int và Interger đều được sử dụng để biểu diễn số nguyên nhưng:

+ int là kiểu dữ liệu nguyên thủy ( primitive type), không phải là một obj. Khi khởi tạo thì giá trị số nguyên sẽ được lưu trực tiếp vào bộ nhớ stack.Tuy nhiên vì không phải obj nên nó sẽ không có các method kèm theo và không thể gán bằng null.  
+ Interger là một class nên nó bao gồm các method hỗ trợ hữu ích, đồng thời object thuộc kiểu Interger được lưu vào bộ nhớ heap và có thể được gán bằng null. Do vậy nên Interger sẽ tốn bộ nhớ hơn int và có hiệu suất chậm hơn int do Heap Memory chậm hơn Stack Memory vì nó cần nhiều thời gian hơn để cấp phát và thu hồi bộ nhớ (garbage collection).

**II.Kiểu dữ liệu java dành cho tiền**- Trong java, khi làm việc liên quan đến tiền, thì ta thường hay sử dụng các kiểu dữ liệu sau để tránh làm tròn số:  
+ BigDecimal : có độ chính xác cao, không bị làm tròn, dải giá trị rất lớn và hỗ trợ nhiều phép toán liên quan đến số thập phân. Tuy nhiên vì là obj nên nó có hiệu suất không cao lắm.  
+ long: là một kiểu dữ liệu nguyên thủy nên có hiệu suất cao và tiết kiệm bộ nhớ hơn, không bị lỗi làm tròn. Tuy nhiên long chỉ hỗ trợ lưu số nguyên, không lưu số thập phân và các phép toán cũng không hỗ trợ số thập phân nên ta chỉ nên dùng long khi dữ liệu là số nguyên.



**III. Tại sao nên tránh cộng chuỗi String trong vòng for, tìm cách tối ưu**- Vì String trong java là bất biến (Immutable), khi cộng hai String a và b, java sẽ tạo một Sring mới thay vì thay đổi String cũ. Tức là String a và b vẫn còn chiếm giữ bộ nhớ mà không mất đi nên khi lặp lại nhiều lần có thể gây tràn bộ nhớ heap.  
- Để tối ưu ta có thể sử dụng StringBuilder:  
 + StringBuilder là một class cho phép thay đổi object trực tiếp mà không cần tạo object mới.  
   
**IV. So Sánh giữa cộng chuỗi bằng concat và bằng toán tử +**- concat() chỉ có thể nối hai đối tượng String, còn + có thể nối String với kiểu dữ liệu khác( int, double,..) bằng cách tự động chuyển đổi kiểu dữ liệu ( autoboxing).  
- concat() chỉ có thể nối hai chuỗi cùng một lúc còn + có thể nối nhiều chuỗi cùng một lúc.  
- + có thể chậm hơn concat() do java ngầm dùng StringBuilder để nối chuỗi.  
**V. Cách truyền tham số từ dòng lệnh trong java**

- giả sử ta đã có một file Hello.java, để truyền tham số từ dòng lệnh ta làm như sau:  
+ mở terminal, chạy câu lệnh javac Hello.java ( Câu lệnh này dùng để compile)  
+ chạy câu lệnh java Hello tham\_so1 tham\_so2  
+ lúc này các tham số được truyền vào sẽ được lưu trong mảng args[] trong phương thức main. Tức là args[0] = tham\_so1, args[1] = tham\_so2.  
+ mặc định tham số truyền vào luôn là String.  
**VI. Tham chiếu ( pass by reference) và tham trị ( pass by value) trong java  
-** Vì không có con trỏ nên khác với C/C++ java không hỗ trợ pass by reference mà chỉ có pass by value. Tuy nhiên java đã giả lập pass by reference bằng cách “pass references by value”. Ví dụ:  
+ Với các kiểu dữ liệu nguyên thủy, khi truyền vào biến kiểu nguyên thủy, java sẽ tạo một bản sao của giá trị đó và truyền vào phương thức. Bất kỳ thay đổi nào trong phương thức đều không ảnh hưởng đến biến gốc. Ví dụ:  
****  
+ Với object, khi truyền một object, thì giá trị được truyền vào sẽ là địa chỉ của object đó nên nếu thay đổi thuộc tính của object bên trong phương thức thì giá trị của object cũng sẽ thay đổi vì cả tham số trong phương thức và biến gốc đều trỏ đến cùng một object.  
Ví dụ :   


Ở ví dụ trên, person là một biến tham chiếu đến object person(“John”) tức là person không trực tiếp chứa object mà nó chứa địa chỉ bộ nhớ của object person(“John”). Khi truyền person vào changename(person), java truyền giá trị của tham chiếu (địa chỉ object) vào p, lúc này p và person đểu trỏ đến object person(“John”) nên khi thay đổi p.name thì person.name cũng bị thay đổi theo.  
  
+ Tuy nhiên nếu viết như sau thì object sẽ không bị thay đổi:  
  
Khi gọi changeReference(person), thì p sẽ trỏ đến person(“John”). Tuy nhiên sau khi viết p = new Person(“Bob”) thì p không còn trỏ đến John nữa mà trỏ đến một object person(“Bob”) còn person vẫn trỏ đến person(“John”). Do đó thay đổi p không làm thay đổi person nên sau khi phương thức kết thúc, person.name vẫn nhận giá trị là John.