Học java có ví dụ cụ thể

Bài 1 - Hello

Bài 2 – In ra chuỗi nhập vào

Bài đầu tiên của bạn, bạn đã học cách để Java in cái gì đó ra màn hình, trong bài này, bạn sẽ học cách nhập vào cái gì đó và Java in cái đó ra màn hình. Gõ cái này đi bạn (lưu ý, bạn phải gõ, không được copy và paste)

```
import java.io.*;
public class Hello {
                public static void main(String[] args) throws Exception {
                                BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
                                System.out.print("Your name is: ");
                                String str;
                                str = in.readLine();
                                System.out.println("Welcome " + str + " to
Java");
                }
}
```

***Lí thuyết: cấu trúc một chương trình Java**

```
public class Core {
                public static void main(String[] args) {
                                System.out.println("Hello,Everybody in the
World!");
                }
}
```

public class Core bạn bắt đầu một lớp Java
public static void main(String[] args) đây là một phương thức main trong Java, để cho chương trình chạy được. Tạm thời bạn phải gõ y như thế này
System.out.println("Hello,Everybody in the World!") đây là một câu lệnh trong Java, đơn giản nó chỉ in ra chuỗi nằm trong 2 dấu “” ra màn hình.
Mọi lớp và phương thức trong Java mở ra bằng { và đóng lại bằng }
Mọi câu lệnh trong java kết thúc bằng ;

Bài 3 – Biến trong Java

```
import java.io.*;
public class Hello {
                public static void main(String[] args) throws Exception {
                                BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
                                System.out.print("Nhap a: ");
```

```
                                int a = Integer.parseInt(in.readLine());
                                System.out.print("Nhap b: ");
                                int b = Integer.parseInt(in.readLine());
                                int ketqua;
                                ketqua = a+b;
                                System.out.println("Ket qua bai toan a+b la: “
+ ketqua);
                }
}
```

Nhập thử 2 số a và b vào đi bạn, kết quả bài toán a+b sẽ được in ra.
***Lí thuyết:**

```
import java.io.*;
public class Hello {
                public static void main(String[] args) throws Exception {
                                BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
                                System.out.print("Your name is: ");
                                String str;
                                str = in.readLine();
                                System.out.println("Welcome " + str + " to
Java");
                }
}
```

Tạm thời, trong chương trình này, bạn chỉ nên quan tâm đến dòng
String str khai báo biến str kiểu chuỗi, và

```
System.out.println("Welcome " + str + " to Java")
```

Đây cũng là dòng System.out.println như chương trình đầu, có khác là + str + tức là đưa một biến vào chuỗi in ra. Chỉ đến đó thôi nhé, sau đó, hãy quan tâm đến bài hôm nay

```
System.out.print("Nhap a: ");
int a = Integer.parseInt(in.readLine());
System.out.print("Nhap b: ");
int b = Integer.parseInt(in.readLine());
int ketqua;
ketqua = a+b;
System.out.println("Ket qua bai toan a+b la: “ + ketqua);
```

Giải thích
import bạn nhập class hay thư viện chuẩn, tạm thời đừng quan tâm nó là gì, chỉ cần nhớ là có nó để chương trình chạy
System.out.print in ra một chuỗi, nhưng không xuống dòng

System.out.println in ra một chuỗi, nhưng xuống dòng
int ketqua tức là khai báo biến ketqua kiểu int
ketqua = a+b tức là gán kết quả một biểu thức tính toán (ở đây là biến a + biến b) cho biến ketqua
System.out.println("Ket qua bai toan a+b la: " + ketqua) thì đơn giản rồi, in cái dòng đó ra, chỉ khác là nó đưa biến ketqua của bạn vào chuỗi đó.

Bài 4 – Chia hết, chia lấy dư

***Lí thuyết: một số kiểu biến trong Java**

Bạn đã biết 2 kiểu String (chuỗi) và int (nguyên) bây giờ bạn biết thêm kiểu float (thực)
Số nguyên và số thực bạn biết sự khác nhau rồi chứ. Bây giờ ta bắt đầu bài toán ví dụ

```
import java.io.*;
public class Hello {
                public static void main(String[] args) throws Exception {
                                BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
                                System.out.print("Nhap a: ");
                                float a = Float.parseFloat(in.readLine());
                                System.out.print("Nhap b: ");
                                float b = Float.parseFloat(in.readLine());
                                float ketqua = a/b;
                                System.out.println("Ket qua bai toan a+b la: "
+ ketqua);
                }
}
```

Bạn thử bài toán xem, nhớ đừng nhập số b=0 nhé, chuyện ấy sẽ xử lí sau.
Ví dụ nhập a=5, b=2, kết quả in ra sẽ là 2.5, thú vị phải không ?
Bây giờ cũng bài toán ấy, bạn thay đổi như sau

```
import java.io.*;
public class Hello {
                public static void main(String[] args) throws Exception {
                                BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
                                System.out.print("Nhap a: ");
                                int a = Integer.parseInt(in.readLine());
                                System.out.print("Nhap b: ");
                                int b = Integer.parseInt(in.readLine());
                                float ketqua = a/b;
                                System.out.println("Ket qua bai toan a+b la: "
+ ketqua);
                }
}
```

Cũng nhập a=5, b=2, lần này kết quả in ra là … 2

Phép chia sẽ là phép chia hết nếu cả 2 toán hạng đều kiểu nguyên, gọi là chia lấy nguyên (/) hay div
Bây giờ cũng chương trình ấy mà ta thay đổi lại chút xíu xem sao

```
import java.io.*;
public class Hello {
                public static void main(String[] args) throws Exception {
                                BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
                                System.out.print("Nhap a: ");
                                int a = Integer.parseInt(in.readLine());
                                System.out.print("Nhap b: ");
                                int b = Integer.parseInt(in.readLine());
                                float ketqua = a%b;
                                System.out.println("Ket qua bai toan a+b la: "
+ ketqua);
                }
}
```

Cũng nhập a=5, b=2, lần này kết quả in ra là … 1
Đây là kết quả phép chia lấy dư 5 chia cho 2, gọi là chia lấy dư (%) hay mod
*Thế nếu tôi muốn 2 số nguyên chia nhau mà ra kiểu thực chứ không phải phép chia lấy nguyên thì sao ?
Trong trường hợp đó, bạn dùng “ép kiểu”
int a=5,b=2;float ket qua;
ketqua=(float)a/b;
Bài 5 – Lập trình OOP

class
Đây là một class, class này có hai property (thuộc tính) là name và age

```
public class Person
{
                String name;
                int age;
}
```

Đây là một class, class này ngoài property còn có constructor (khởi tạo) của class đó

```
public class Person
{
                String name;
                int age;
                public Person(String name)
                {
                                this.name = name;
```

```
                }
}
```

Trong cái constructor này hãy lưu ý một điều, đó là biến this. Biến this có nghĩa là bản thân cái class đó (ở đây là class Person).
Trong class Person có một property là age, câu this.age = age có nghĩa là cái thuộc tính age của class Person sẽ nhận giá trị ở cái đối số age do constructor Person(int age) đưa vào.
Lưu ý là mọi class đều có sẵn ít nhất một constructor không có đối số.
Đây là một class, class này ngoài property, constructor còn có một behavior (hành vi)

```
public class Person
{
                String name;
                int age;
                public Person(int age)
                {
                                this.age = age;
                }
                public void Nhap()
                {
                                nameonsole.readLine("Nhap ho ten:");
                }
}
```

Khi ta viết câu lệnh sau
Person personOne = new Person(12);
Thì ta đã tạo ra một instance (thể hiện) là personOne của class Person
Gửi vào 31 December 2007 - 12:43 AM
Bài 5 – Lập trình OOP (tiếp)
Khai báo một class
public abstract class MyClass {}
Từ thứ 1 là khai báo quyền truy xuất và kế thừa,có 3 loại
-public:được phép truy xuất từ bất cứ nơi nào và bất cứ lớp nào cũng được quyền kế thừa
-protected:chỉ có phương thức cùng gói được phép truy xuất và kế thừa
-private:chỉ có phương thức cùng gói được phép truy xuất nhưng không lớp nào được phép kế thừa
-nếu không khai báo,mặc định là protected
Từ thứ 2 là khai báo một lớp trừu tượng hay là không trừu tượng
Nhiem vu: tao 1 lop Person, tao tiep 2 lop Students va Teachers ke thua lop Person, tao lop Execute chua ham chinh de chay chuong trinh.
--lop Person-

```
import corejava.*;
abstract class Person
{
                //cai nay goi la cac property hay state-thuoc tinh cua doi
```

```
tuong
                String hoten;
                int age;
                String diachi;
                int luong;
                //cac constructor
                public Person(int age)
                {
                                this.age = age;
                }
                //cac method hay behavior-hanh vi cua doi tuong
                public void Nhap()
                {
                                hoten = Console.readLine("Nhap ho ten:");
                                diachi = Console.readLine("Nhap dia chi:");
                }
                //vi la 1 class thuoc loai abstract nen Person duoc phep khai
bao cac method khong co noi dung, noi dung cua class In se duoc cac lop ke
thua no them vao noi dung cua rieng no
          public abstract void In();
                public abstract int Tinhluong();
}
```

--lop Students-

```
import corejava.*;
class Students extends Person
{
                int MaSV,Malop;
                public void Nhap()
                {
                                super.Nhap();
                                MaSV = Console.readInt("Nhap ma SV:");
                                Malop = Console.readInt("Nhap ma lop:");
                }
                public void In()
                {
                                System.out.println(hoten);
                                System.out.println(diachi);
                                System.out.println(MaSV);
                                System.out.println(Malop);
                }
                public int Tinhluong()
                {
                return 150000;
                }
}
```

tu khoa super se goi ham Nhap() tu lop Person la cha cua lop Students

--lop Teachers-

```
import corejava.*;
class Teachers extends Person
{
                int Makhoa;
                public void Nhap()
                {
                                super.Nhap();
                                Makhoa = Console.readInt("Nhap ma khoa::");
                }
                public void In()
                {
                                System.out.println(hoten);
                                System.out.println(diachi);
                                System.out.println(Makhoa);
                }
                public int Tinhluong()
                {
                                return 500000;
                }
}

--lop Execute-

import corejava.*;
class Execute
{
                public static void main(String args[])
                {
                                Students st = new Students();
                                st.Nhap();
                                st.In();
                                st.luong=st.Tinhluong();
                                Teachers tc = new Teachers();
                                tc.Nhap();
                                tc.In();
                                tc.luong=tc.Tinhluong();
                }
}
```

Khai báo một thuộc tính:
Khai báo 1 thuộc tính
public static void temp;
Từ thứ 1 là khai báo quyền truy xuất,có 3 loại
-public:được phép truy xuất từ bất cứ nơi nào
-protected:chỉ có lớp con mới được phép truy xuất
-private:chỉ có lớp đó xài(thuộc tính riêng của nó)
-nếu không khai báo,mặc định là protected

b.Từ thứ 2 là khai báo cách truy xuất(static)
static(tĩnh)
-nếu không khai báo,mặc định là không tĩnh
Tất cả các đối tượng thể hiện từ lớp cha đều được phép thay đổi giá trị của các thuộc tính không tĩnh,còn giá trị của thuộc tính tĩnh thì không được phép thay đổi

```
public class Car
{
                public string branch;
                public int cost;
                public static int tire=4;
}
```

Như ví dụ trên,tất cả các lớp con của lớp Car (như ToyotaCar,Peugeot,Mazda…) đều được phép thay đổi các thuộc tính branch hay cost để phù hợp cho riêng mình,nhưng thuộc tính tire (số bánh xe) không được phép thay đổi vì là thuộc tính tĩnh
Nói cách khác, chỉ có một và chỉ một thuộc tính có tên là tire trong class Car và tất cả các class con của nó, vì vậy gọi là tĩnh

Khai báo một hành vi
Một phương thức được khai báo như sau
public static double ketqua()
Có 3 chỉ định truy xuất là public, protected và private
-public:được phép truy xuất từ bất cứ nơi nào
-protected:chỉ có lớp kế thừa lớp chứa nó được truy xuất
-private:chỉ lớp chứa nó được truy xuất(dùng nội bộ)
-nếu không khai báo,mặc định là protected
Có 6 chỉ định thuộc tính là static, abstract, final, native, synchronized (đồng bộ) và volatile (linh hoạt)
static(tĩnh)
-nếu không khai báo,mặc định là không tĩnh

```
class TestObject
{
                static void StaticMethod() {…}
                void NonStaticMethod() {…}
}
```

Nếu là một phương thức không tĩnh, đầu tiên bạn phải khởi tạo một đối tượng,sau đó mới được phép gọi phương thức
TestObject test=new TestObject();
test.NonStaticMethod();
Nếu là một phương thức tĩnh,bạn được phép gọi trực tiếp từ lớp
TestObject.StaticMethod();
abstract(trừu tượng)
Một phương thức trừu tượng không có nội dung.Nội dung của nó sẽ được các lớp con tùy biến và phát triển

theo hướng của riêng nó.
- final: không thể được extends hay override (ghi đè)
- native: thân phương thức viết bằng C hay C++
- synchronized: chỉ cho phép 1 thread truy cập vào khối mã ở cùng một thời điểm
- volatile: sử dụng với biến để thông báo rằng giá trị của biến có thể được thay đổi vài lần vì vậy không ghi vào thanh ghi
.Từ thứ 3 là giá trị trả về.Nếu không có giá trị trả về thì là void

interface-template

Bây giờ ta có 1 khái niệm mới, là giao diện. Giao diện ra đời chính là để giải quyết đa kế thừa. Mỗi lớp trong Java chỉ có 1 lớp cha, nhưng có thể implements nhiều giao diện.
Giao diện được khai báo giống như 1 lớp, cũng có state và behavior. Nhưng state của giao diện là final còn behavior là abstract
Giả sử, ta sẽ khai báo một giao diện

```
public interface Product
{
                //hai state duoi day la final, tuc la lop implements khong
duoc phep doi gia tri
                static string maker = “My Corp”;
                static string phone = “555-7767”;
                //behavior duoi day la abstract, tuc la khong co noi dung
                public int getPrice(int id);
}
```

Bây giờ, ta sẽ viết một class có cài đặt (implements) giao diện này

```
public class Shoe implements Product
{
                public int getPrince(int id)
                {
                                return (id= =1)?5:10;
                }
                public String getMaker()
                {
                                return maker;
                }
}
```

Muốn implements nhiều giao diện, làm như sau, ví dụ class Toyota extends Car implements ActionCar, ActionMobilation

package-unit
Hãy tạo 1 thư mục có tên là Transport
Bên trong thư mục này hãy tạo 2 file là Car.java và Bicycle.java như sau

--Car.java-

```
package Transport;
public class Car
{
		public String manufacturer;
		public int year;
}
```

--Bicycle.java-

```
package Transport;
public class Bicycle
{
		public int cost;
		public Bicycle(int cost)
		{
				this.cost = cost;
		}
}
```

Như vậy là ta đã tạo ra 1 gói chứa 2 lớp là Car và Bicycle. Bây giờ ta có 1 chương trình muốn sử dụng gói này là TestProgram.java. Ta viết:
--ViDuTransport.java-

```
import Transport.*;
class TestProgram
{
		public static void main(String args[])
		{
				Car myCar = new Car();
				myCar.manufacturer = “Toyota”;
				Bicycle myBicycle = new Bicycle(1500);
		}
}
```

Lưu ý nếu trong file ViDuTransport bạn không khai báo import Transport.* thì bạn vẫn có thể khai báo tường minh như sau
Transport.Car myCar = new Transport.Car();
nạp chồng (overload) 1 phương thức

```
class Vidu
{
```

```
                public satic void main(String a[])
                {
                                private  float cost;
                                public float CalculateSalePrice()
                                {
                                                return cost*1.5;
                                }
                                public float CalculateSalePrice(double heso)
                                {
                                                return cost*(1+heso);
                                }
                }
}
```

Ở đây có 2 phương thức trùng tên CalculateSalePrice nhưng phương thức thứ 2 khác tham số, gọi là nạp chồng

*** nạp chồng (overload) và ghi đè (override)**

Những phương thức được nạp chồng là những phương thức trong cùng một lớp, có cùng một tên nhưng danh sách đối số khác nhau

Phương thức được ghi đè là phương thức có mặt ở lớp cha, được xác định là phương thức chung cho các lớp con, rồi xuất hiện ở các lớp con

Nạp chồng là một hình thức đa hình (polymorphism) trong quá trình biên dịch (compile) còn ghi đè là trong quá trình thực thi (runtime)

Bài 6 – Các kiểu dữ liệu nguyên thủy và phép toán

- Kiểu nguyên: gồm số nguyên(int,long)
- Kiểu dấu phẩy động (hay kiểu thực): gồm số thực(float,double)
- Kiểu kí tự (char)
- Kiểu chuỗi (String)

Hằng kí tự khai báo như sau, ví dụ 'H' (khác với "H" là một chuỗi kí tự)

Một số hằng kí tự đặc biệt, ví dụ '\\' để biểu diễn chính kí tự \, và \u biểu diễn Unicode, ví dụ:

'\u00B2' biểu diễn ² (bình phương)

'\u00BC' biểu diễn ¼ (một phần tư)

'\u0170' biểu diễn ª (mũ a)

- Kiểu boolean

Có 2 giá trị là 2 từ khóa true và false, và không thể chuyển kiểu sang int

*Khai báo biến

int i,j; //2 biến i và j có kiểu dữ liệu là int

char ch='A'; //biến ch kiểu char khởi tạo giá trị đầu 'A'

*Khai báo hằng

Hằng được khai báo với từ khóa final. Ví dụ:

final float PI = 3.14159;

*Phép toán

Phép toán của Java giống C. Trong class java.lang.Math có một số method để dùng trong toán học như sau

double y = Math.pow(x,a) = x^a

và random, sin, cos, tan, exp (mũ), log(logarit) ...

* Các phép toán số học

- Với cả kiểu nguyên và kiểu thực: + - * / (phép chia sẽ cho ra kết quả kiểu thực nếu một trong 2 toán tử là kiểu thực)
- Chia hết (/) chỉ áp dụng khi cả 2 toán tử là kiểu nguyên, ví dụ 10/3=3

- Chia lấy dư (%) chỉ áp dụng khi cả 2 toán tử là kiểu nguyên, ví dụ 10%3=1
* Các phép toán quan hệ (so sánh)
- Bao gồm ==,<,>,<=,>= trả về kiểu boolean
* Các phép toán với kiểu logic
- Bao gồm and(kí hiệu &&) or(kí hiệu ||) not(kí hiệu !)
* Phép ++ và --
- Phép này có 2 dạng, một là ++biến hay --biến, hai là biến++ hay biến-- Sự khác nhau chỉ là khi phép này thực hiện chung với một phép toán khác thì
- Với ++biến và --biến thì nó sẽ thực hiện phép toán này trước rồi mới thực hiện phép toán khác
- Với biến++ và biến-- thì nó sẽ thực hiện phép toán khác trước rồi mới thực hiện phép toán này
* Phép gán
- Phép này có dạng a=5
- Phép gán phức, ví dụ a+=5 nghĩa là a=a+5, hay a*=2 nghĩa là a=a*2
* Trình tự kết hợp
Hầu hết các phép toán điều có trình tự kết hợp từ trái sang phải, chỉ có các phép sau là từ phải sang trái
- Phép ++ và --
- Các phép gán như =,+=,-=,<<=,>>=

Bài 7 – Mệnh đề if

nếu em đẹp thì tôi sẽ cưa em không thì tôi cưa đứa khác
IF em đẹp THEN tôi sẽ cưa em ELSE tôi cưa đứa khác
IF(em đẹp) tôi sẽ cưa em;
ELSE tôi cưa đứa khác;
Cú pháp (syntax) của mệnh đề IF là
if(mệnh đề) lệnh 1;
else lệnh 2;
Nếu mệnh đề đúng thì thực hiện lệnh 1;
Không thì thực hiện lệnh 2;
Ví dụ

```
if(a>b) System.out.println(“So lon nhat la “+a);
else System.out.println(“So lon nhat la “+b);
```

***Ta xây dựng một bài toán làm tròn số**

Nhập vào một số bất kì. Nếu phần thập phân số này >=0.5, làm tròn tăng lên một đơn vị, ngược lại giảm đi một đơn vị.

```
import java.io.*;
public class Hello {
                public static void main(String[] args) throws Exception {
                                BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
                                System.out.print("Nhap a: ");
                                float a = Float.parseFloat(in.readLine());
                                float ketqua=a%1;
                                if(ketqua>=0.5) a=a-ketqua+1;
                                else a=a-ketqua;
                                System.out.println("Ket qua bai toan la: " +
a);
```

```
                }
}
```

*** Phép điều kiện ? và phép chọn :**
- Giả sử có mệnh đề if

if(a> a=2;
else a=0;
Phép điều kiện biểu diễn như sau a=a>b?2:0 nghĩa là nếu chân trị của a>b là đúng thì a=2 nếu là sai thì a=0
* Sau khi học xong if, bạn có rất nhiều bài tập để mà … làm, cổ điển nhất vẫn là giải phương trình bậc một và hai, ngoài ra còn nhiều bài tập khác nữa. Ở đây chỉ có giải phương trình bậc một. Bạn nên tìm nhiều bài tập để tự làm trước khi tiếp tục phần kế.
Ví dụ: phương trình bậc 1

```
import java.io.*;
public class Hello {
                public static void main(String[] args) throws Exception {
                                BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
                                System.out.println("Giai phuong trinh bac nhat
dang ax+b=0");
                                System.out.print("Nhap he so a: ");
                                float a = Float.parseFloat(in.readLine());
                                System.out.print("Nhap he so b: ");
                                float b = Float.parseFloat(in.readLine());
                                if(a==0) {
                                                if(b==0)
System.out.println("Phuong trinh vo so nghiem");
                                                if(b!=0)
System.out.println("Phuong trinh vo dinh");
                                }
                                else System.out.println("Phuong trinh mot
nghiem x=" + -b/a);
                }
}
```

Bài 8 – switch
Bạn đã học xong if. Bạn muốn dùng vòng lặp if để đánh giá điểm số nhập vào. Bạn sẽ viết chương trình sau đây

```
import java.io.*;
public class Hello {
                public static void main(String[] args) throws Exception {
                                BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
                                System.out.print("Nhap diem so: ");
                                int diem = Integer.parseInt(in.readLine());
```

```
                                if(diem<=2) System.out.println("Yeu");
                                if((diem>2) && (diem<=3))
System.out.println("Trung binh");
                                if((diem>3) && (diem<=4))
System.out.println("Kha");
                                if((diem>4) && (diem<5))
System.out.println("Gioi");
                                if(diem==5) System.out.println("Xuat sac");
                }
}
```

Thay vì lặp lại những câu if ấy, bạn nên dùng switch

```
import java.io.*;
public class Hello {
                public static void main(String[] args) throws Exception {
                                BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
                                System.out.print("Nhap diem so: ");
                                int diem = Integer.parseInt(in.readLine());
                                switch(diem)
                                {
                                                case 0:
                                                case 1:
                                                case
2:System.out.println("Yeu");break;
                                                case
3:System.out.println("Trung binh");break;
                                                case 4:
                                                case
5:System.out.println("Gioi");break;

default:System.out.println("Vao sai");
                                }
                }
}
```

*break với switch: break sẽ thoát ngay ra khỏi khối lệnh trong thân của switch

Bài 9 – String

Khác với C, String là một lớp của Java. String được khai báo như sau

String a = "Hello";

Cộng 2 String bằng dấu +

System.out.println("Gia tri la " + n);

Java có khả năng tự chuyển kiểu bất cứ dữ liệu kiểu số nào khi cộng vào String. Dù n là int, float, double đều có thể chuyển thành String nhờ mẹo vặt ("" + n)

Các method trong class String

*** substring**

String s1 = "Hello";
String s2 = s1.substring(0,4); //bắt đầu từ kí tự thứ 0 (tức là 'H') lấy đi 4 kí tự (tức là "Hell")
*** length**
int n = s1.length(); //tức là bằng 5
*** charAt**
char ch = s1.charAt(4); //tức là bằng 'o'
Đây là method tìm kí tự thứ i trong String, các kí tự trong String được đánh số từ 0
*** equals**
Kiểu tra xem chuỗi nguồn s có giống chuỗi đích d hay không, ta dùng method equals trả về boolean
boolean b = s.equals(t);
String không giống dữ liệu kiểu số, tuyệt đối không dùng giống như if(s==t)
*** compareTo**
int a = s2.compareTo(s1);
a>0 s2>s1
a<0 s2<s1
a=0 s2=s1
So sánh giữa s2 và s1 là so sánh thứ tự giữa kí tự đầu của hai chuỗi so đi, ví dụ "kc" > "kazbe"
*** toCharArray (đổi chuỗi ra mảng kí tự)**
char[] chuoi = s1.toCharArray();
*** indexOf**
String s1 = "Hello Everybody";
String s2 = "lo";
int n = s1.indexOf(s2); //n sẽ bằng 4
Đây là method trả về vị trí của chuỗi s2 trong chuỗi s1, nếu không tìm thấy sẽ trả về -1
*** Chuyển kiểu từ String ra dữ liệu kiểu số**
Chuyển từ dữ liệu kiểu số ra String khá dễ dàng, dùng "" + n, nhưng ngược lại thì phải dùng các method tương ứng.
Các method này nằm trong góijava.lang, trong các class Byte, Short, Integer, Long, Float, Double
String input = "230";
int n = Integer.parseInt(input); //n sẽ bằng 230
Tương tự với các method sau Byte.parseByte, Short.parseShort, Float.parseFloat, ...
Bài 10 – vòng lặp for

```
for(int i=0;i<n;i++)
                s+=i;
```

3 thành phần trong câu for ta có thể bỏ hết nhưng phải giữ lại các dấu ; khi đó nến muốn ta có thể đặt phép toán điều khiển vòng lặp trong thân lệnh như sau

```
for(int i=0;i<n;)
{
                s+=i;
                i++;
}
```

***break với for: break sẽ thoát ngay ra khỏi vòng for**

```
for(int i=0;i<n;i++)
{
                System.out.println(i);break;
                System.out.println("Tiep tuc");
}
```

Kết quả in ra không có câu "Tiep tuc" vì break nhảy ngay ra khỏi vòng for sau khi in 1
*continue với for: continue sẽ khiến vòng for bắt đầu 1 chu trình mới và bỏ qua tất cả các lệnh bên dưới nó
VD: in tất cả các số từ 0 đến 10, bỏ qua 3,4,5

```
for(int i=0;i<10;i++)
{
                if((i= =3)||(i= =4)||(i= =5)) continue;
                System.out.println(i);
}
```

Bài 11 – while

```
while(biểu thức)
                lệnh;
```

Nếu biểu thức đúng thì thực hiện lệnh
***break với while: break sẽ thoát ngay ra khỏi vòng while**

```
int i=0;
while(i<10)
{
                System.out.println(i);break;
                i++;
}
```

Sẽ chỉ in ra 0
*** continue với while: nó sẽ xác định giá trị biểu thức viết ngay sau while**

```
int i=0;
while(i<10)
{
                System.out.println(i);continue;
```

```
                i++;
}
```

Bài 12 – vòng lặp do..while

```
do
                lệnh;
while(biểu thức);
```

Nếu biểu thức đúng thì tiếp tục thực hiện lệnh
***break với do..while: break sẽ thoát ngay ra khỏi vòng while**

```
int i=0;
do
{
                System.out.println(i);break;
                i++;
}

while(i<10);
```

Sẽ chỉ in ra 0
*** continue với while: nó sẽ xác định giá trị biểu thức viết ngay sau**

```
while
int i=0;
do
{
                System.out.println(i);continue;
                i++;
}
while(i<10);
```

Ta sẽ được một loạt in 0 vô tận
Bài 13 – array
Ta khai báo 1 mảng với câu lệnh sau, và không cung cấp số phần tử
int[] a;
Tuy vậy, với Java, để dùng được một array, ta cần phải khởi tạo array đó, và lúc này phải cung cấp số phần tử
int[] a;
a = new int[100];

Hai câu có thể viết lại thành một câu
int[] a = new int[100];
Java sẽ khởi tạo một mảng 100 phần tử đều là int có đánh thứ tự từ 0 đến 99
Mảng có giá trị đầu: Mảng loại này không cần new mà cũng chẳng cần số phần tử
int[] a = {1,45,6,8,21};
Các method với mảng
*** length**
method này sẽ cung cấp số phần tử của mảng, ví dụ ta muốn gán giá trị số cho các phần tử của mảng a
for(int i=0;i<a.length;i++) a|i|=i; lưu ý là length, không phải length()
***System.arraycopy**
Giả sử, ban đầu ta có 2 mảng
int[] s = {1,3,5,7,9,11,13,15};
int[] d = {2,4,6,8,10,12,14};
method arraycopy trong gói System
System.methodcopy(s,3,d,2,4);
sẽ cho ra một mảng d mới là {2,4,7,9,11,13,14}
method này sẽ thay thế 4 phần tử, tính từ phần tử thứ 2 trong mảng d, bằng ngần ấy phần tử tính từ phần tử thứ 3 trong mảng s
Các method nằm trong class java.util.Arrays
*** void sort**
Nó sẽ sắp xếp một mảng số tăng dần
int[] s = {28,7,14,11};
Arrays.sort(s);
*** int binarySearch**
Nó sẽ tìm vị trí của một phần tử trong một mảng, trả về -1 nếu không tìm thấy
int[] s = {28,7,14,11};
int n = Arrays.binarySearch(s,14); n sẽ bằng 2
Mảng nhiều chiều
int[][] = new int[100][50];
Hoặc khai báo 1 mảng có giá trị đầu. Đây là mảng 2 chiều gồm 4 phần tử là 4 mảng 1 chiều, mỗi mảng 1 chiều chứa 3 phần tử

```
int[][] a =
{
{16, 3, 2},
{5, 10, 11},
{9, 6, 7},
{4, 15, 14}
};
```

Bài 14 - ngoại lệ

```
[code]int x,y;
x=10;y=x-10;
x=x/y;
```

Khi chạy đoạn mã này bạn sẽ thấy xuất hiện thông báo
java.lang.ArithmeticException: divide by zero
Và chương trình sẽ thoát ra ngay lúc đó. Muốn chương trình chạy tiếp và không thoát ra, ta đón "bắt" ngoại

lệ này, đưa ra biến e, cuối cùng in e (để xem là ngoại lệ gì)

```
int x,y;
try
{
                x=10;y=x-10;
                x=x/y;
}
catch(Exception e)
{
                System.out.println(e.getMessage());
}
```

Xử lí ngoại lệ (Exception)
Để "ném" ngoại lệ do bất cứ dòng mã nào trong một phương thức sinh ra, bạn có thể khai báo để ném bỏ ngoại lệ đó

```
public void divide() throws Exception
{
                int a=5/0;
}
```

hoặc nếu muốn "bắt" ngoại lệ đó lại để xem đó là ngoại lệ gì để xử lí, bạn "bắt" nó rồi in ra

```
try
{
                int a=5/0;
}
catch(Exception e)
{
                System.out.println(e.getMessage());
}
```

Nếu muốn chương trình thành công thì sinh thông báo thành công, thất bại thì sinh thông báo ngoại lệ, bạn có thể dùng

```
boolean done=false;
try
{
                int a=5/b;
                done=true;
}
```

```
catch(Exception e)
{
                System.out.println(e.getMessage());
}
if(done==true) System.out.println("Successful");
```

Bài 15 - Vector (mảng không giới hạn số phần tử)

Các method trong bài này nằm ở 2 class java.util.Vector và java.util.Enumeration

Khai báo

Vector vt = new Vector();

Nhập dữ liệu cho một Vector (class Console nằm trong gói corejava)

Lưu ý là mỗi phần tử của Vector đều phải là một đối tượng, nên ta phải có new Integer(n) khi muốn đưa vào một biến kiểu int. Tương tự với Byte, Long, Float, ...

```
do
{
int n = Console.readInt("");
if(n!=0) vt.addElement(new Integer(n));
}
while(n!=0);
```

In ra các phần tử của một Vector

```
for(int i=0;i<vt.size();i++)
System.out.println(vt.elementAt(i));
```

Để đưa Vector về kiểu mảng cho dễ thao tác, ta đưa về kiểu Enumeration (một kiểu mảng)

Enumeration e = vt.elements();

Như vậy ta có mảng e kiểu Enumeration sao chép y khuôn Vector vt để dễ xử lí, không đụng đến Vector vt

In ra các phần tử của một Enumeration

```
while(e.hasMoreElements())
System.out.println(e.nextElement());
```

Bài 16 - Lớp nội (lớp nằm trong lớp khác)

```
public class TestProgram
{
                static int currentCount;
                static class Apple
                {
                                int weight;
                                public Apple(int weight)
                                {
                                                this.weight=weight;
                                                currentCount++;
                                }
                                public int Weight()
```

```
                                {
                                        return weight;
                                }
                }
                public static void main(String args[])
                {
                                Apple a=new Apple(12);//khoi tao 1 quả tao
nang 12kg
                                System.out.print(a.Weight());
                }
}
```

Ở đây ta thấy lớp nội Apple trong lớp TestProgram, khi biên dịch Java sẽ làm xuất hiện 2 file là TestProgram.class và TestProgram$Apple.class. Ưu điểm khi sử dụng lớp nội là:
- thể hiện tính đóng gói cao
- các lớp nội có thể truy xuất trực tiếp các biến của lớp cha
Lưu ý là lớp nội khác với các lớp mà nằm chung một file, ví dụ như tập tin MainClass.java dưới đây

```
public class MainClass
{
}
class Subclass
{
}
```

Khi biên dịch nó sẽ tạo ra 2 file là MainClass.class và Subclass.class

Bài 17 - Tạo tập tin jar tự chạy
Giả sử chương trình của bạn có vài file .class trong đó file chương trình chính là MainPro.class chẳng hạn. Bạn hãy tạo một file lấy tên là mymf.mf có nội dung như sau
Main-Class: MainPro

Bắt buộc phải chính xác như thế (tức là phải có cả xuống dòng), không thì trình chạy jar không hiểu được. Sau đó bạn vào %JAVA_HOME%\bin\ chép tất cả các tập tin .class của ứng dụng và cả mymf.mf vào đó, rồi chạy jar.exe với tham số dòng lệnh như sau
jar cmfv mymf.mf MyProgram.jar *.class
Tương tự nếu bạn muốn đưa thêm 2 thư mục dir1 và dir2 vô file JAR thì bạn cũng gõ
jar cmfv mymf.mf MyProgram.jar *.class dir1 dir2
Trình jar sẽ tạo file MyProgram.jar (tên khác tùy bạn) có thể chạy được, không phải dùng lệnh java hay giả sử không có IDE quen thuộc của bạn