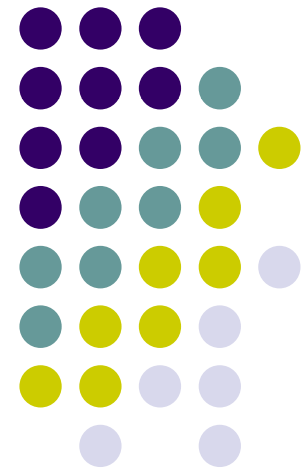


Giới thiệu sơ lược về ngôn ngữ Java

- Sơ lược về ngôn ngữ Java
- Các khái niệm cơ bản
- Biên dịch và thực thi dùng JDK
- Một số kỹ thuật
- Ngoại lệ
- Nhập xuất
- Thread



Sơ lược về ngôn ngữ Java



- **Lịch sử phát triển:**

- ❖ 1990: Ngôn ngữ Oak được tạo ra bởi James Gosling trong dự án Green của Sun Microsystems nhằm phát triển phần mềm cho các thiết bị dân dụng.
- ❖ 1995: Oak đổi tên thành Java.
- ❖ 1996: trở thành chuẩn công nghiệp cho Internet.

- **Đặc điểm:**

- ❖ Ngôn ngữ hoàn toàn **hướng đối tượng** (Pure OOP).
- ❖ Ngôn ngữ **đa nền**: "**Viết một lần , Chạy trên nhiều nền**".
- ❖ Ngôn ngữ **đa luồng** (multi-threading): xử lý và tính toán song song.
- ❖ Ngôn ngữ **phân tán** (distributed): cho phép các đối tượng của một ứng dụng được phân bố và thực thi trên các máy tính khác nhau.
- ❖ Ngôn ngữ **động**: cho phép mã lệnh của một chương trình được tải từ một máy tính về máy của người yêu cầu thực thi chương trình.
- ❖ Ngôn ngữ **an toàn**: hạn chế các thao tác nguy hiểm cho máy tính thật.
- ❖ Ngôn ngữ **đơn giản**, dễ học, kiến trúc chương trình đơn giản, trong sáng.

Sơ lược về ngôn ngữ Java



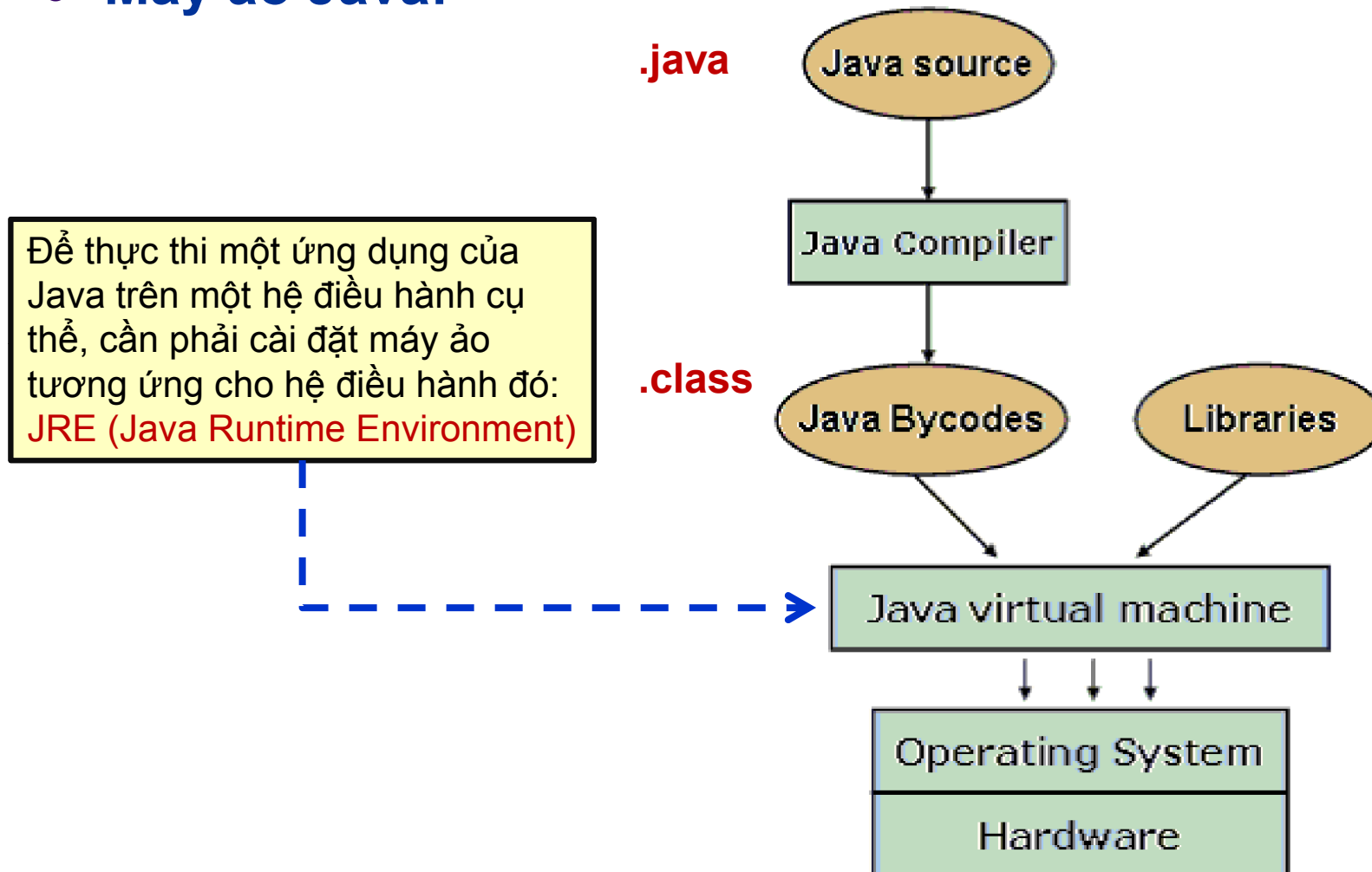
- **Khả năng:**

- ❖ Ngôn ngữ **bậc cao**.
- ❖ Có thể được dùng để tạo ra các loại ứng dụng để giải quyết các vấn đề về số, xử lý văn bản, tạo ra trò chơi, và nhiều thứ khác.
- ❖ Có các thư viện hàm hỗ trợ **xây dựng giao diện** (GUI) như AWT, Swing, ...
- ❖ Có các **môi trường lập trình đồ họa** như JBuilder, NetBeans, Eclipse, ...
- ❖ Có khả năng **truy cập dữ liệu** từ xa thông qua cầu nối JDBC
- ❖ Hỗ trợ các lớp hữu ích, tiện lợi trong **lập trình các ứng dụng mạng** (Socket) cũng như truy xuất Web hay nhúng vào trong trang Web (Applet).
- ❖ Hỗ trợ **lập trình phân tán** (Remote Method Invocation) cho phép một ứng dụng có thể được xử lý phân tán trên các máy tính khác nhau.
- ❖ **Lập trình trên thiết bị cầm tay** (J2ME).
- ❖ Xây dựng các ứng dụng trong **môi trường xí nghiệp** (J2EE).
- ❖ ...

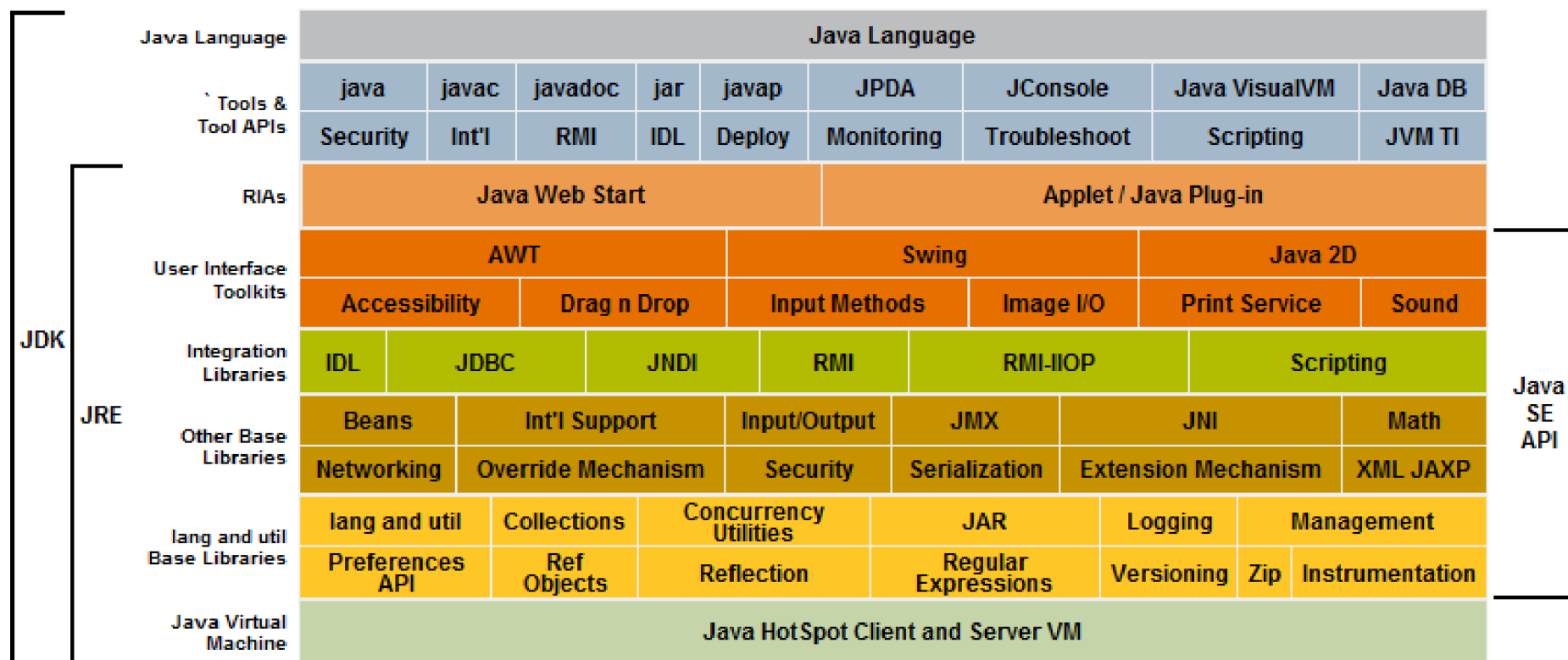
Sơ lược về ngôn ngữ Java



- **Máy ảo Java:**



Sơ lược về ngôn ngữ Java



Java Platform, Standard Edition 7 (Java SE)

Sơ lược về ngôn ngữ Java



- Các kiểu ứng dụng dưới Java



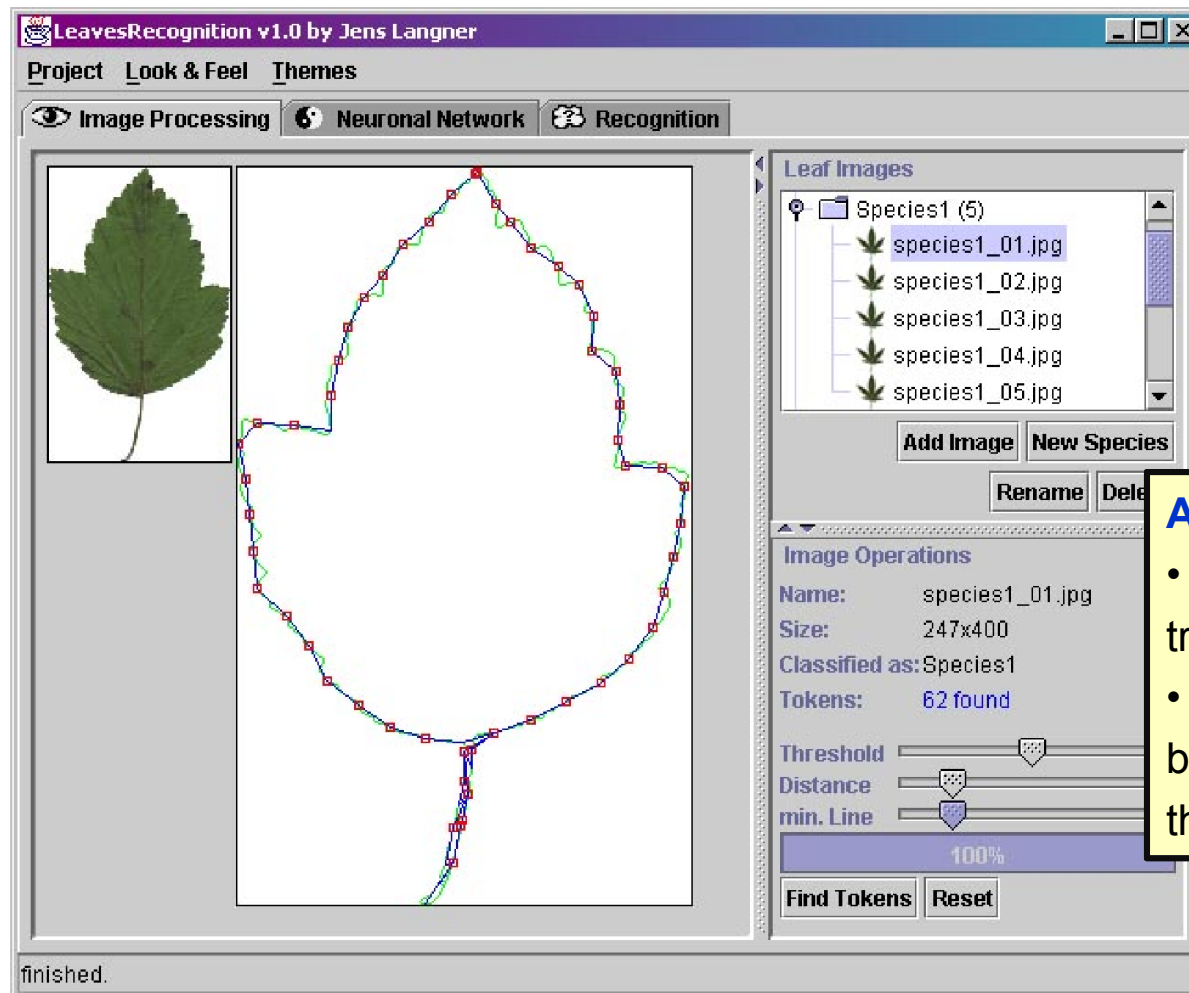
Applet:

- Ứng dụng được **nhúng vào các trang web**.
- Mã ứng dụng được lấy từ web server

Sơ lược về ngôn ngữ Java



- Các kiểu ứng dụng dưới Java



Application:

- Ứng dụng được thực thi trên các máy ảo Java.
- Bộ thông dịch dịch mã bytecode của ứng dụng thành mã máy đích.

Sơ lược về ngôn ngữ Java



- Các kiểu ứng dụng dưới Java



Mobile:

- Ứng dụng được thực thi trên các máy ảo Java trên điện thoại di động và các thiết bị cầm tay.

Java Development Kit (JDK)



- **Bộ phát triển ứng dụng java gồm:**

- ❖ **javac**: Bộ biên dịch chương trình viết bằng ngôn ngữ java thành mã thực thi(**byte code**) trên máy ảo Java
- ❖ **java** (máy ảo java – Java Virtual Machine): Thông dịch mã bytecode của các chương trình kiểu application thành mã thực thi được trên hệ điều hành của máy đích.
- ❖ **appletviewer**: Bộ thông dịch thực thi applet.
- ❖ **javadoc**: Tạo tài liệu tự động.
- ❖ **jdb**: Gỡ rối.
- ❖ **rmic**: Tạo stub cho ứng dụng kiểu RMI.
- ❖ **rmiregistry**: Phục vụ tên (Name Server) trong hệ thống RMI
- ❖ ...

Các kiểu dữ liệu



- **Kiểu số**

Tên kiểu	Kích thước
byte	1 byte
short	2 bytes
int	4 bytes
long	8 bytes
float	4 bytes
double	8 bytes

- **Kiểu ký tự char:**

- ❖ 2 bytes theo mã UNICODE
- ❖ 127 ký tự đầu trùng với mã ASCII

- **Kiểu chuỗi String:**

- ❖ Là một lớp trong ngôn ngữ java.
- ❖ Có nhiều phương thức thao tác trên chuỗi.

- **Kiểu Boolean:**

- ❖ nhận 2 giá trị true / false

Mảng



- **Kiểu mảng:**

- ❖ **Khai báo**

hoặc:

```
int[] a ; float[] yt; String[] names;  
int a[]; float yt[]; String names[];  
int maTran[][]; float bangDiem[][];
```

- ❖ **Khởi tạo**

```
a = new int[3];    yt = new float[10];  
names = new String[50];  
maTran = new int[10][10];
```

- ❖ **Sử dụng**

```
int i = a[0];      float f = yt[9];  
String str = names[20];    int x = matran[0][0];
```

Các phép toán



- **Phép toán số học:**

$+, -, *, /, \%, =$

$++, --, +=, -=, *=, /=, \% =$

- **Phép toán logic:**

$=, !=, \&\&, ||, !$

$>, <, >=, <=$

- **Phép toán trên bit:**

$\&, |, ^, <<, >>, \sim$

- **Phép toán điều kiện:**

? :

- **Phép toán chuyển kiểu:**

(kiểu mới)

Quy cách đặt tên



- Tên phân biệt giữa chữ hoa và chữ thường.
- Dùng các chữ cái, ký tự số, ký tự _ và \$.
- Không bắt đầu bằng ký tự số.
- Không có khoảng trắng trong tên.
- Quy ước:
 - ❖ Tên lớp:
 - Các ký tự đầu tiên của một từ được viết Hoa,
 - Các ký tự còn lại viết thường.
 - Ví dụ: *lớp Ngươi, SinhVien, MonHoc, String, InputStream, . . .*
 - ❖ Tên biến, tên hằng, tên phương thức:
 - Từ đầu tiên viết thường.
 - Ký tự đầu tiên của từ thứ hai trở đi được viết Hoa.
 - Ví dụ: *ten, ngaySinh, diaChi, inDiaChi(), getInputStream(), . . .*

Java - Ứng dụng kiểu application



- Java ngôn ngữ thuần đối tượng (pure object):
=> *Tất cả đều được định nghĩa trong các lớp (class)*
- Trong một ứng dụng có một **lớp thực thi được**.
- Lớp thực thi được:
 - ❖ Có tên lớp **trùng với tên tập tin** chứa nó.
 - ❖ Phải khai báo **public**
 - ❖ Có chứa phương thức được thực thi đầu tiên:

```
public static void main (String argv[]){  
    ...  
}
```
 - ❖ Nếu nhiều lớp được định nghĩa trong một tập tin:
=> chỉ có một lớp được khai báo **public**.

Ứng dụng kiểu application



- Ví dụ ứng dụng kiểu application
 - ❖ Định nghĩa trong lớp HelloWorld
 - ❖ Chứa trong tập tin HelloWorld.java

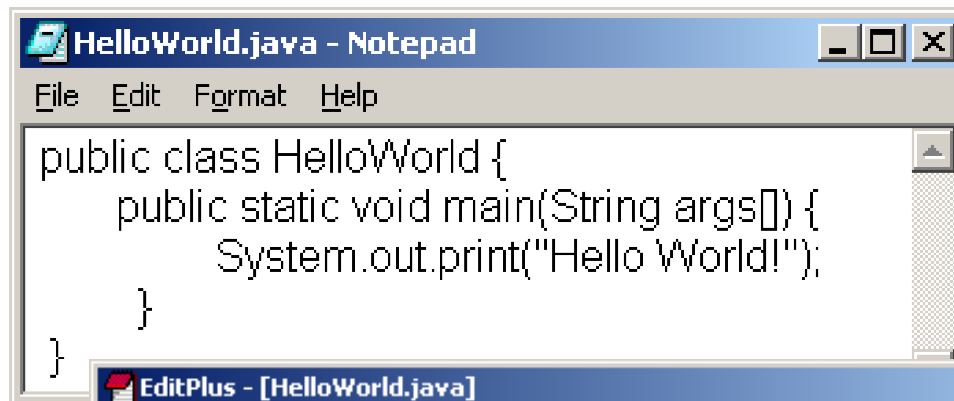
```
public class HelloWorld {  
    public static void main(String args[]) {  
        System.out.print("Hello World!");  
    }  
}
```

Phương thức **System.out.print()** sẽ in tất cả các tham số trong dấu ngoặc ra màn hình..

Ứng dụng kiểu application

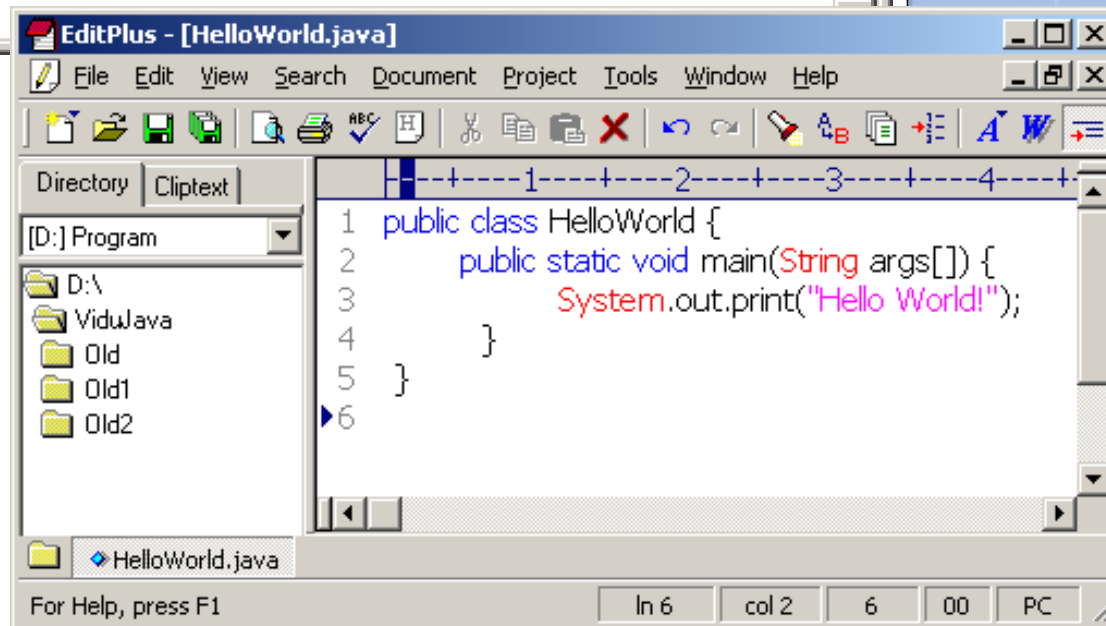


- Biên soạn chương trình Java



```
public class HelloWorld {  
    public static void main(String args[]) {  
        System.out.print("Hello World!");  
    }  
}
```

Có thể dùng bất kỳ chương trình soạn thảo văn bản nào.



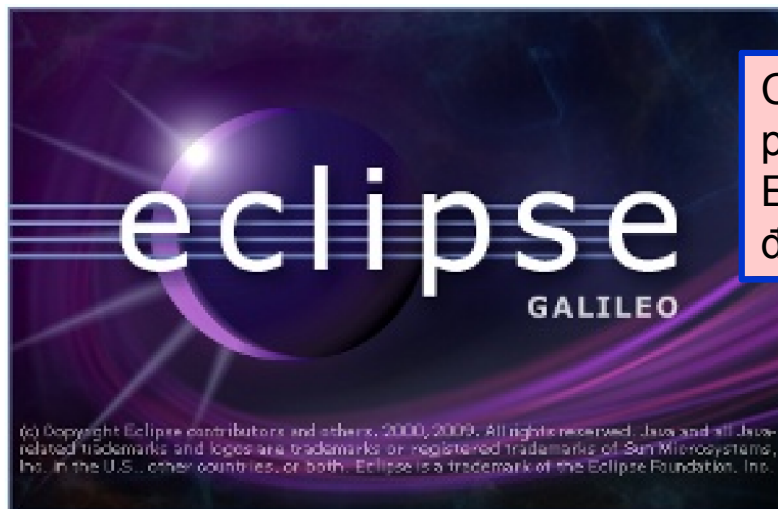
```
1 public class HelloWorld {  
2     public static void main(String args[]) {  
3         System.out.print("Hello World!");  
4     }  
5 }  
6
```



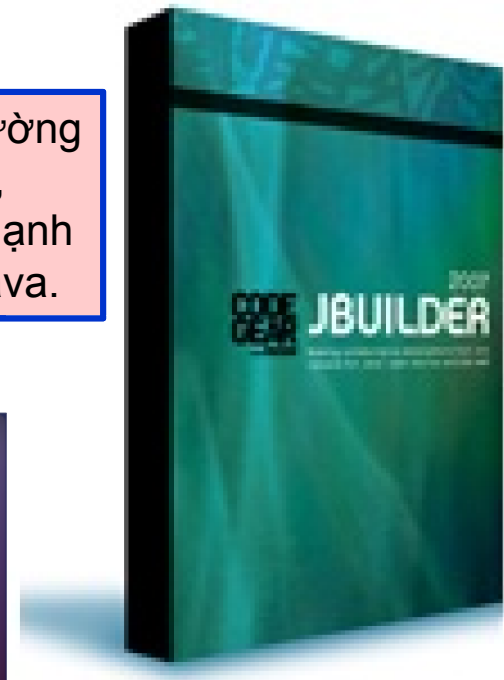
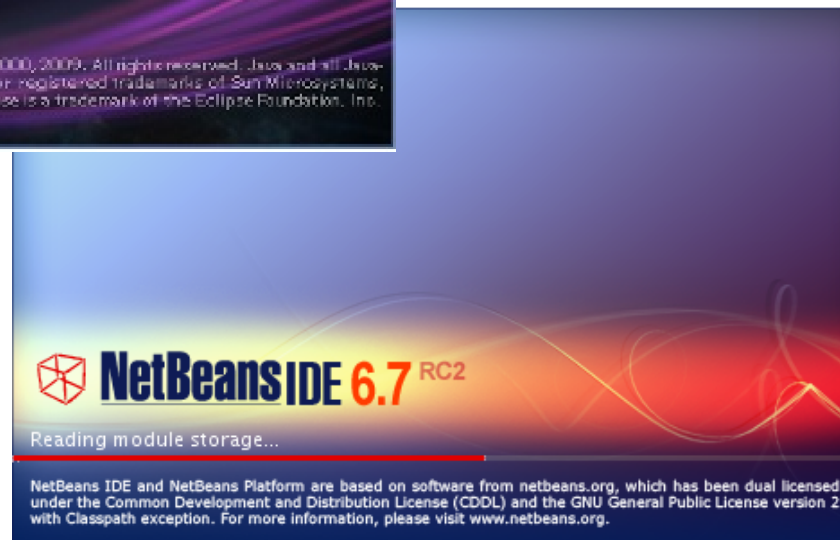
Ứng dụng kiểu application



- Biên soạn chương trình Java



Có nhiều các IDE (môi trường phát triển) như NetBeans, Eclipse, JBuilder, ... rất mạnh để xây dựng ứng dụng Java.



Cài đặt và cấu hình JDK

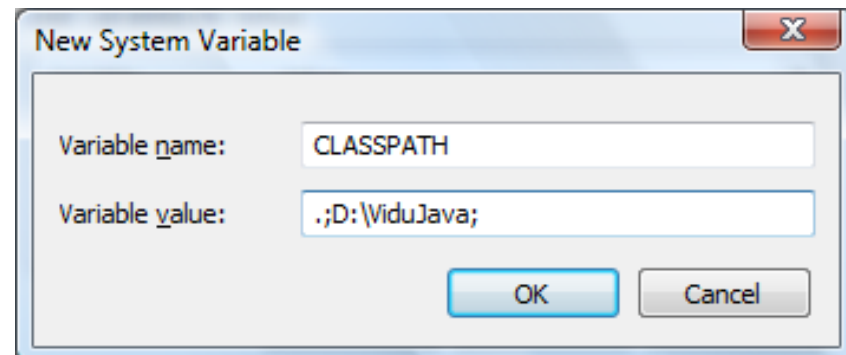
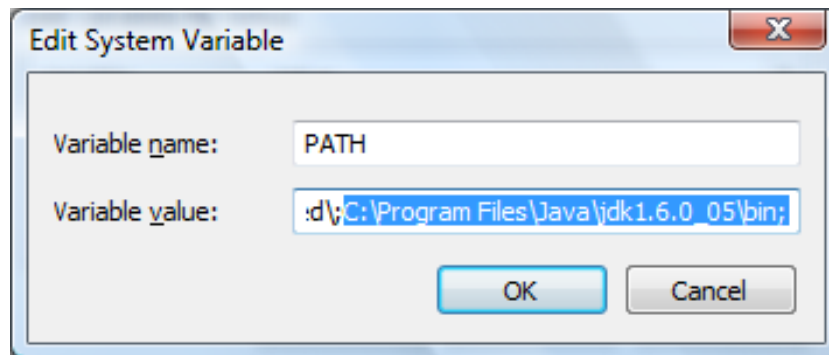


- Cài đặt

- ❖ Phiên bản: jdk-6u5-windows-i586-p.exe
- ❖ Thư mục cài đặt: C:\Program Files\Java\jdk1.6.0_05

- Cấu hình

- ❖ Đặt biến môi trường PATH và CLASSPATH
 - Vào System Properties / Environment Variables: trong phần System Variables, **thêm tiếp tục** vào đường dẫn **PATH**
C:\Program Files\Java\jdk1.6.0_05\bin;
 - Click chọn **NEW**, thêm vào
CLASSPATH = .; [Các thư mục hoạt động khác, nếu cần]



Biên dịch và thực thi chương trình



- Biên dịch và thực thi trong Command Prompt
 - ❖ Giả sử thư mục sử dụng là **D:\vidujava**
 - ❖ Giả sử file nguồn là **HelloWorld.java**

```
C:\Windows\system32\cmd.exe

D:\ViduJava>javac HelloWorld.java

D:\ViduJava>java HelloWorld
Hello World!

D:\ViduJava>_
```

Cấu trúc điều khiển



- Cấu trúc điều khiển trong Java giống hệt như cấu trúc điều khiển của C++, bao gồm:
 - ❖ Lệnh if-else
 - ❖ Phép toán ? :
 - ❖ Lệnh switch
 - ❖ Lệnh while
 - ❖ Lệnh do – while
 - ❖ Lệnh for
 - ❖ Lệnh break
 - ❖ Lệnh continue

Một số kỹ thuật



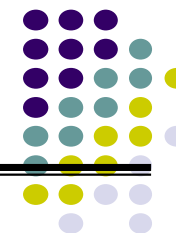
- **Hiển thị ra màn hình**

- ❖ `System.out.print(arg1+ arg2+ .. + argn)`
- ❖ `System.out.println(arg1+ arg2+ .. + argn)`: xuống hàng.

```
public class Display {  
    public static void main(String args[]) {  
        int i = 10;  
        String str = " nam yeu ";  
        char ch = 'm';  
        System.out.print("\n" + "Bai hat " + i + str + ch);  
    }  
}
```

```
Command Prompt  
D:\UduJava>javac Display.java  
D:\UduJava>java Display  
Bai hat 10 nam yeu m  
D:\UduJava>
```

Một số kỹ thuật



- Nhập **1 ký tự** từ bàn phím
 - ❖ **System.in.read()** trả về một số nguyên là thứ tự trong bảng mã ASCII của ký tự vừa nhập từ bàn phím.

```
import java.io.*;
public class KeyRead {
    public static void main(String args[]) {
        try {
            System.out.print("Nhập 1 ký tự: ");
            int ch = System.in.read();
            System.out.print("Ký tự " + (char)ch + " có mã ascii = " + ch);
        } catch (IOException ie) {
            System.out.print("Error " + ie);
        }
    }
}
```

```
C:\ Command Prompt
D:\ViduJava>javac KeyRead.java
D:\ViduJava>java KeyRead
Nhập 1 ký tự: A
Ký tự A có mã ascii = 65
D:\ViduJava>java KeyRead
Nhập 1 ký tự: a
Ký tự a có mã ascii = 97
D:\ViduJava>
```

Một số kỹ thuật



- Đọc đối số của chương trình
 - ❖ Thực thi **java ClassName arg1 arg2 arg3 argn**
 - Các đối số cách nhau khoảng trắng.
 - main() phải khai báo một tham số kiểu mảng các chuỗi.
 - Các đối số lần lượt được đặt vào các phần tử của mảng.
 - Số lượng đối số: truy xuất thuộc tính **length** của mảng.

```
public class PrintArgs {  
    public static void main (String args[]) {  
        for (int i = 0; i < args.length; i++) {  
            System.out.println(args[i]);  
        }  
    }  
}
```

Giá trị đối số của chương trình
được nhận vào theo **dạng chuỗi**

```
C:\WINNT\system32\cmd.exe  
D:\UiduJava>java PrintArgs chuoi1 1234 56.78 e  
chuoi1  
1234  
56.78  
e  
D:\UiduJava>
```

Một số kỹ thuật



- Đổi chuỗi thành số và số thành chuỗi

```
public class TimSoMax {  
    public static void main (String args[]) {  
        int max=Integer.valueOf(args[0]).intValue();  
        int vitri=0;  
        for(int i=1; i<args.length; i++) {  
            int x= Integer.valueOf(args[i]).intValue();  
            if(max<x)  
                { max= x; vitri = i; }  
        }  
        System.out.print("Số lớn nhất trong dãy là " + max);  
        System.out.println(" tại vị trí thứ " + (vitri+1) );  
    }  
}
```

Đổi chuỗi thành số

```
int i = Integer.valueOf(str).intValue();  
long l = Long.valueOf(str).longValue();  
float f = Float.valueOf(str).floatValue();
```

Đổi số thành chuỗi

cộng 1 chuỗi rỗng ("") cho số đó.
Ví dụ:

```
int x = 15; float y = 3.14;  
String str1 = "" + x;  
String str2 = "" + y;
```

```
C:\WINNT\system32\cmd.exe  
D:\UiduJava>java TimSoMax 5 -3 9 17 0 -14 8  
Số lớn nhất trong dãy là 17 tại vị trí thứ 4  
D:\UiduJava>
```


Một số kỹ thuật



- Đổi chuỗi thành mảng các byte (byte[])
 - ❖ Dùng hàm **getBytes()** trên chuỗi.

```
public class ChuoiVaMang {  
    public static void main (String args[]) {  
        String str1 = "Day la chuoi can doi";  
        byte b[] = str1.getBytes();  
        System.out.print("Cac ky tu gom: ");  
        for(int i=0; i<b.length; i++)  
            System.out.print((char)b[i] + " ");  
    }  
}
```

```
C:\WINNT\system32\cmd.exe  
  
D:\UiduJava>java ChuoiVaMang  
Cac ky tu gom: D a y   l a   c h u o i   c a n   d o i  
D:\UiduJava>
```

Một số kỹ thuật



- Đổi mảng các byte (byte[]) thành chuỗi
 - ❖ **String(byte[] b)**: tạo chuỗi từ tất cả các phần tử trong mảng b[]
 - ❖ **String(byte[] b, int offset, int length)**: Tạo chuỗi bắt đầu tại vị trí offset và chiều dài length từ các phần tử trong mảng b[].

```
public class ChuoiVaMang1 {  
    public static void main (String args[]) {  
        byte b[] = new byte[50];  
        int i=0;  
        for(byte x='A'; x<='a'; x++)  
            b[i++]=x;  
        String str = new String(b, 0, i);  
        System.out.println(str + " - Chiều dài = " + str.length());  
    }  
}
```

```
C:\WINNT\system32\cmd.exe  
D:\UiduJava>java ChuoiVaMang1  
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`a - Chiều dài = 33  
D:\UiduJava>_
```

Cài đặt và sử dụng 1 lớp



```
public class Person {  
    String name;        //Thuộc tính  
    String address;     //Thuộc tính  
    Person(String n, String address) {    // Hàm xây dựng  
        name = n;  
        this.address = address;  
    }  
    void display() {    // Hiện thị tên và địa chỉ ra màn hình  
        System.out.print(name + " is at " + address);  
    }  
    public static void main(String args[]){  
        Person tom = new Person("Tom","Disney Land");  
        tom.display();  
    }  
}
```

Cú pháp cài đặt 1 lớp trong Java gần giống cú pháp cài đặt 1 lớp trong C++

Không khai báo thuộc tính truy cập thì mặc nhiên là **public** trong cùng package (xem như trong cùng thư mục chứa nó).

Cú pháp tạo đối tượng:
ClassName obj = **new** **ClassName**([các tham số]);

```
C:\WINNT\system32\cmd.exe  
D:\UduJava>java Person  
Tom is at Disney Land  
D:\UduJava>
```

Cài đặt và sử dụng 1 lớp



- Cài đặt nhiều lớp trong 1 tập tin

```
public class MultiClass {  
    public static void main(String args[]){  
        Person tom = new Person("Tom", "Disney Land");  
        tom.display();  
    }  
}  
class Person{  
    String name;  
    String address;  
  
    Person(String n, String address) {  
        name = n;  
        this.address = address;  
    }  
    void display(){  
        System.out.println(name + " is at " + address);  
    }  
}
```

- Chỉ có một lớp được khai báo là **public**
- Lớp public sẽ được nhìn thấy (truy xuất được) bởi các lớp khác ở bất kỳ ở đâu.
- Lớp public sẽ chứa hàm main().
- Các lớp còn lại mặc nhiên là **package-private**, nghĩa là có thể truy xuất được từ các lớp trong cùng file hay cùng package (thư mục).

Thừa kế trong Java



- Một lớp chỉ có thể có một lớp cha (thừa kế đơn).
- Dùng từ khóa **extends** để khai báo thừa kế.
- Lớp cha được tham khảo từ lớp con bởi từ khóa **super**.

```
public class Customer extends Person{
    int telephone;
    long buy;
    public Customer(String n, String a, int t, long b) {
        super(n,a);
        telephone=t;
        buy=b;
    }
    public void display() {
        super.display();
        System.out.println( " , telephone:" + telephone + " , buy: " + buy );
    }
    public static void main(String args[]){
        Customer tom = new Customer("Tom","Disney Land",123456,1000);
        tom.display();
    }
}
```

Ngoại lệ (Exception) trong Java



- Ngoại lệ là các lỗi sinh ra từ các "thao tác không chắc chắn" như:
 - ❖ Mở 1 file không tồn tại
 - ❖ Nối kết mạng không thực hiện được
 - ❖ Nhập, xuất dữ liệu không đúng định dạng
- Java bắt buộc các lệnh có thể dẫn đến các lỗi truy xuất (ngoại lệ) phải có các đoạn mã xử lý phòng hờ:

```
try {  
    [ Các thao tác vào ra có thể sinh ra các ngoại lệ ]  
}  
catch (KiểuNgoạiLệ01 biến) {  
    [ Ứng xử khi ngoại lệ KiểuNgoạiLệ01 sinh ra ]  
}  
catch (KiểuNgoạiLệ02 biến) {  
    [ Ứng xử khi ngoại lệ KiểuNgoạiLệ02 sinh ra ]  
}  
finally {  
    [ Công việc luôn luôn được thực hiện ]  
}
```

Ngoại lệ (Exception) trong Java



- Ví dụ

```
public class TimSoMax {  
    public static void main (String args[]) {  
        int max=Integer.valueOf(args[0]).intValue();  
        int vitri=0;  
        for(int i=1; i<args.length; i++) {  
            int x= Integer.valueOf(args[i]).intValue();  
            if(max<x)  
                { max= x; vitri = i; }  
        }  
        System.out.print("Số lớn nhất trong dãy là " + max);  
        System.out.println(" tại vị trí thứ " + (vitri+1) );  
    }  
}
```

Chương trình
không có phần xử
lý ngoại lệ

```
C:\WINNT\system32\cmd.exe  
D:\UiduJava>java TimSoMax  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0  
    at TimSoMax.main(TimSoMax.java:3)  
D:\UiduJava>_
```

nếu ta không nhập
vào đối số dòng
lệnh giá trị nào

```
C:\WINNT\system32\cmd.exe  
D:\UiduJava>java TimSoMax 12 -8 23 4 6 -6 abc 3 22  
Exception in thread "main" java.lang.NumberFormatException: For input string: "abc"  
    at java.lang.NumberFormatException.forInputString(Unknown Source)  
    at java.lang.Integer.parseInt(Unknown Source)  
    at java.lang.Integer.valueOf(Unknown Source)  
    at TimSoMax.main(TimSoMax.java:6)
```

nếu ta nhập vào
giá trị không phải
là số nguyên

Ngoại lệ (Exception) trong Java



```
public class TimSoMax {
    public static void main (String args[]) {
        int vitri=-1, max=Integer.MIN_VALUE;
        try {
            max=Integer.valueOf(args[0]).intValue();
            vitri=0;
        }
        catch(ArrayIndexOutOfBoundsException e)
        { System.out.println("Chua nhap tham so"); return; }
        catch(NumberFormatException e)
        { System.out.println("Sai dinh dang tai vi tri so thu 1"); }
        for(int i=1; i<args.length; i++) {
            try {
                int x= Integer.valueOf(args[i]).intValue();
                if(vitri==-1)    // chua gan duoc so max ban dau
                    { max = x; vitri = i; }
                else
                    if(max<x)    // neu van co 1 so > max
                        { max= x; vitri = i; }
            }
            catch(NumberFormatException e)
            { System.out.println("Sai dinh dang tai vi tri so thu " + (i+1) ); }
        } // for
        if(vitri!=-1) {
            System.out.print("So lon nhat trong day la " + max);
            System.out.println(" tai vi tri thu " + (vitri+1) );
        }
        else
            System.out.println("Trong day doi so khong co so nguyen");
    }
}
```

Chương trình **có**
phần xử lý ngoại lệ

```
C:\WINNT\system32\cmd.exe

D:\UiduJava>java TimSoMax
Chua nhap tham so

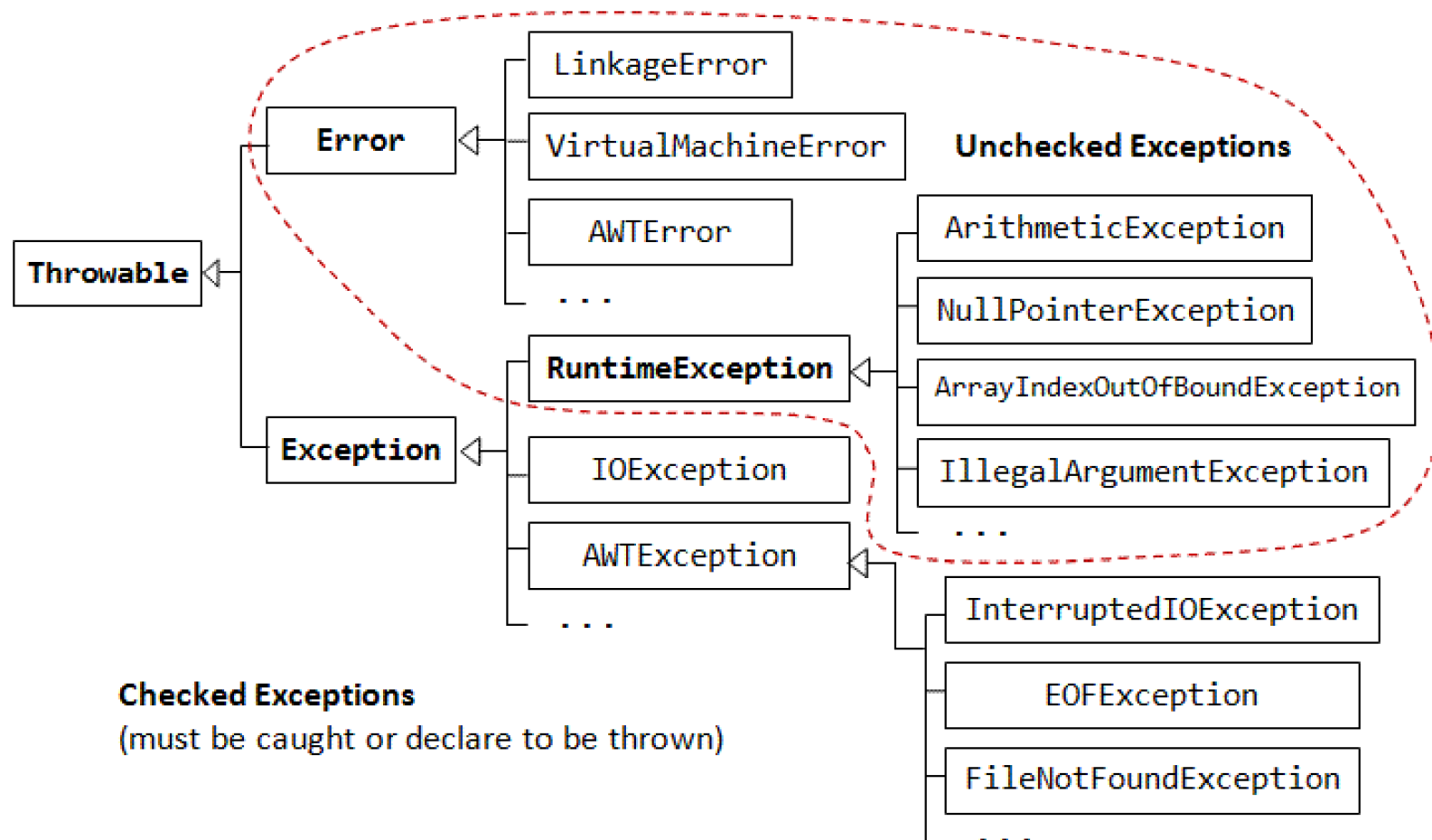
D:\UiduJava>java TimSoMax abc def gg
Sai dinh dang tai vi tri so thu 1
Sai dinh dang tai vi tri so thu 2
Sai dinh dang tai vi tri so thu 3
Trong day doi so khong co so nguyen

D:\UiduJava>java TimSoMax 12 -8 23 4 6 -6 abc 98 3 22
Sai dinh dang tai vi tri so thu 7
So lon nhat trong day la 98 tai vi tri thu 8
```


Ngoại lệ (Exception) trong Java



- Sơ đồ thừa kế của Exception

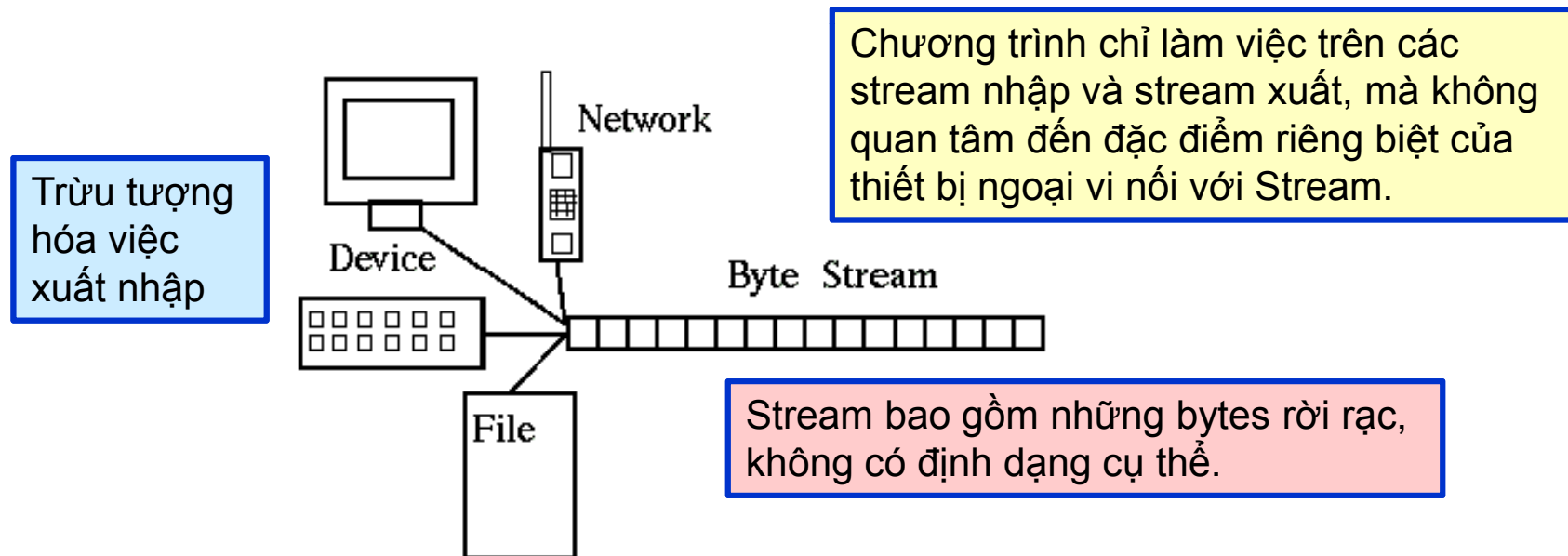


Java – Nhập xuất với Stream



- Stream

- ❖ Stream là một dòng liên tục, có thứ tự các bytes dữ liệu “chảy” giữa chương trình và các thiết bị ngoại vi:
 - Stream nhập (in): nhận (read) dữ liệu từ thiết bị ngoại vi.
 - Stream xuất (out): xuất (write) dữ liệu ra thiết bị ngoại vi.



Java – Nhập xuất với Stream

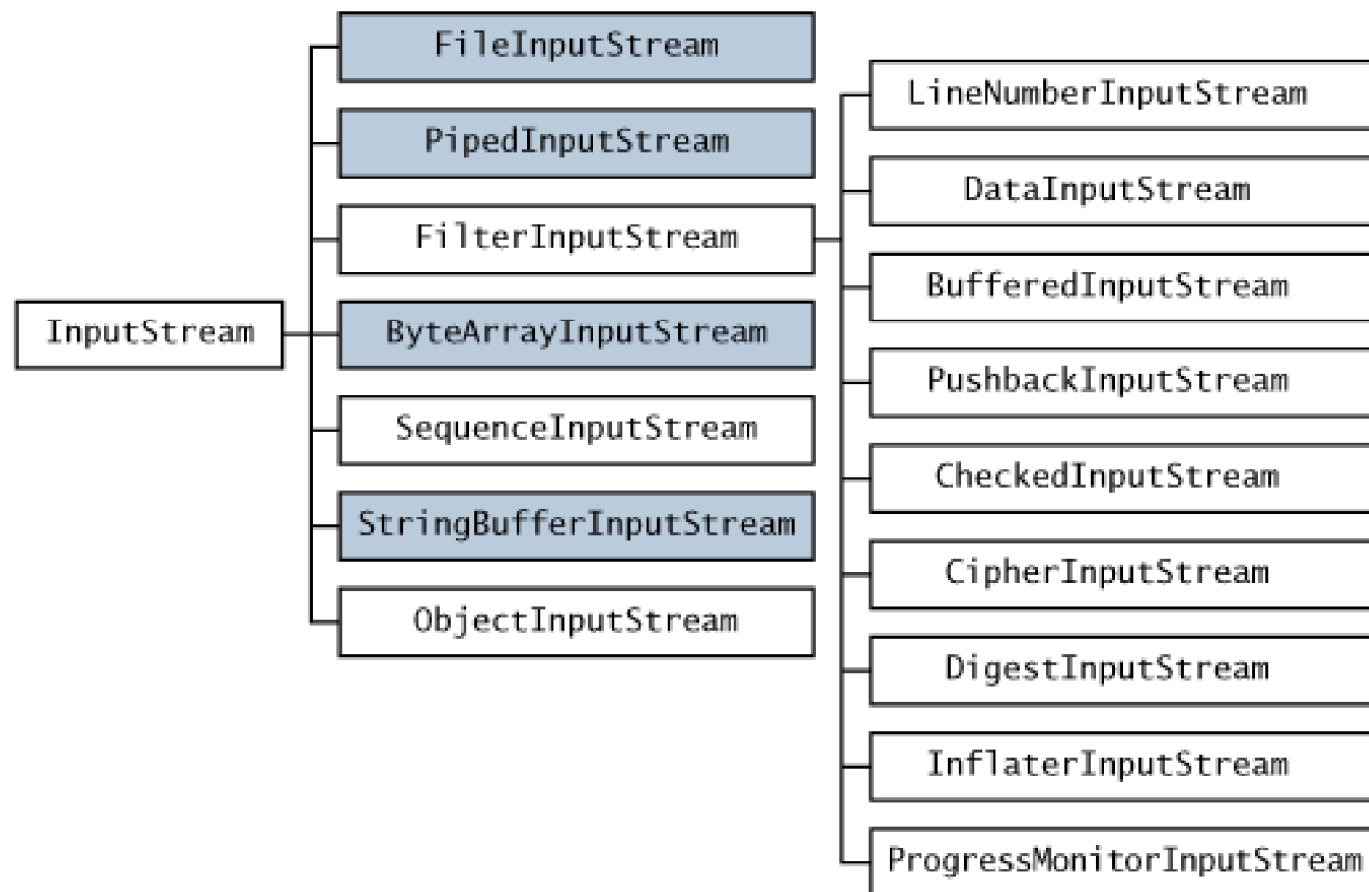


- Stream trong Java
 - ❖ 2 lớp cơ bản trong gói java.io
 - **java.io.InputStream**: Stream nhập
 - **java.io.OutputStream**: Stream xuất
 - Dữ liệu không định dạng:
 - Nhập xuất theo từng ký tự hoặc
 - Nhập xuất theo nhiều byte (byte[]).
 - ❖ Các lớp thừa kế từ InputStream và OutputStream:
 - Cung cấp các tiện ích nhập xuất theo nhiều định dạng.
 - FileInputStream, PipedInputStream, ObjectInputStream, DataInputStream, BufferedReader, ...
 - FileOutputStream, PipedOutputStream, ObjectOutputStream, DataOutputStream, PrintWriter, ...

Java – Nhập xuất với Stream



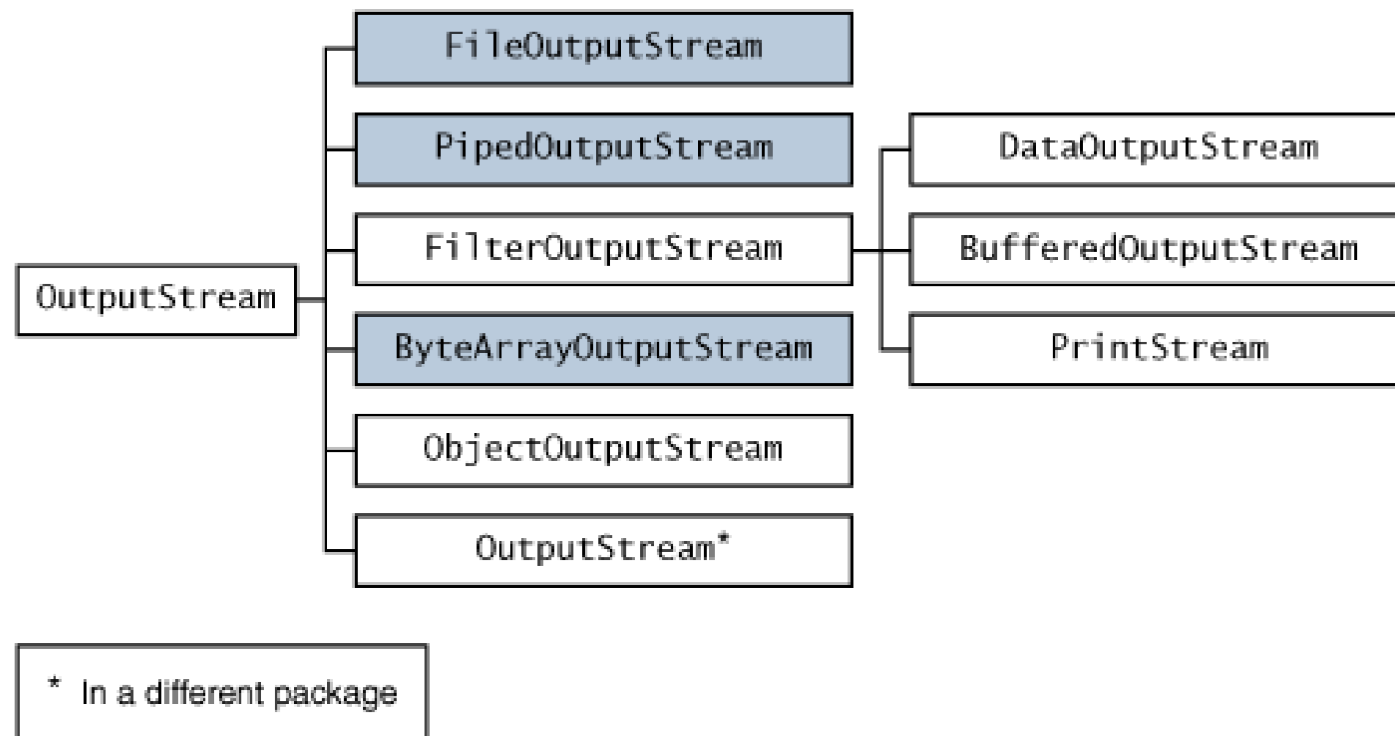
- Sơ đồ thừa kế của InputStream



Java – Nhập xuất với Stream



- Sơ đồ thừa kế của OutputStream

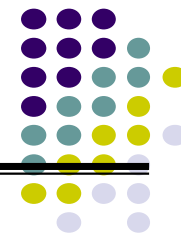


Java – Nhập xuất với Stream



- Lớp `java.io.InputStream`
 - ❖ Các phương thức chính:
 - ❑ `public abstract int read()` throws `IOException`: nhận 1 ký tự
 - ❑ `public int read(byte b[])` throws `IOException`: nhận nhiều ký tự
 - ❑ `public int read(byte b[], int offset, int len)` throws `IOException`
 - ❑ `public int available()` throws `IOException`
 - ❑ `public long skip(long n)` throws `IOException`
 - ❑ `public void close()` throws `IOException`
 - ❖ **System.in** là một `InputStream` nối kết với bàn phím.
 - ❑ Nhận các ký tự nhập từ bàn phím.
 - ❑ Thêm 2 ký tự kết thúc là `'\r'` và `'\n'` khi nhập xong.
- => Phải loại bỏ 2 ký tự đó khi sử dụng dữ liệu nhận về.

Java – Nhập xuất với Stream



- Ví dụ 1 về java.io.InputStream

```
import java.io.*;
public class Nhap1 {
    public static void main(String[] args) {
        InputStream is = System.in; // Ban phim
        while(true) {
            try {
                System.out.print("Nhap 1 ky tu : ");
                int ch = is.read();
                if (ch == -1 || ch == 'q') break;
                System.out.println("Ky tu nhap duoc: " + (char)ch);
            }
            catch(IOException e)
            { System.out.println("Co loi ve nhap xuat");
            }
        }
    }
}
```

- Nhập từng ký tự từ bàn phím.
- Hiển thị ra màn hình ký tự đó.
- Kết thúc khi nhập vào ký tự q

```
C:\Windows\system32\cmd.exe
D:\ViduJava>java Nhap1
Nhap 1 ky tu : a
Ky tu nhap duoc: a
Nhap 1 ky tu : K
Ky tu nhap duoc: K
Nhap 1 ky tu : q
D:\ViduJava>
```

Java – Nhập xuất với Stream



• Ví dụ 2 về java.io.InputStream

```
import java.io.*;
public class Nhap2 {
    public static void main(String args[]) {
        while (true) {
            System.out.print("Nhập chuỗi: ");
            byte b[] = new byte[100];    // Tạo vùng đệm để nhập chuỗi
            try {
                int n = System.in.read(b);    // Nhập n ký tự
                String str = new String(b, 0, n-2);    // Chuyển byte[] -> String
                if (str.equals("EXIT")) break;    // Kiểm tra điều kiện thoát
                System.out.println("Chuỗi nhận được là: " + str);
            }
            catch (IOException ie) {
                System.out.print("Error: " + ie);
            }
        }
    }
}
```

- Nhập 1 chuỗi ký tự từ bàn phím.
- Hiện thị ra màn hình chuỗi ký tự đó.
- Kết thúc khi nhập vào chuỗi "EXIT"

```
C:\WINNT\system32\cmd.exe
D:\UiduJava>java Nhap2
Nhập chuỗi: Ngo Ba Hung & Nguyen Cong Huy
Chuỗi nhận được là: Ngo Ba Hung & Nguyen Cong Huy
Nhập chuỗi: EXIT
D:\UiduJava>
```


Java – Nhập xuất với Stream



- Ví dụ 3 về java.io.InputStream

```
import java.io.*;
public class Nhap3 {
    public static void main(String args[]) {
        try {
            FileInputStream f1 = new FileInputStream("C:/Test.txt");
            int len = f1.available();
            System.out.println("Chieu dai file: " + len);
            System.out.println("Noi dung file:");
            for(int i=0; i<len; i++)
                System.out.print((char)f1.read());
            f1.close();
        }
        catch (IOException ie)
        { System.out.println("Loi khi truy xuất file"); }
    }
}
```

- Đọc nội dung 1 file vào vùng đệm.
- Hiển thị ra màn hình nội dung đó.

```
C:\WINNT\system32\cmd.exe
D:\UiduJava>java Nhap3
Chieu dai file: 66
Noi dung file:
Day la dong thu nhat
abcd efgh 0123456789
Ket thuc file tai day!
D:\UiduJava>
```

Java – Nhập xuất với Stream



- Lớp `java.io.OutputStream`
 - ❖ Các phương thức chính:
 - ❑ `public abstract void write(int b) throws IOException`: xuất ra 1 ký tự có giá trị là `b`
 - ❑ `public void write(byte b[]) throws IOException`: xuất tất cả byte dữ liệu được lưu trong mảng `b[]`.
 - ❑ `public void write(byte b[], int offset, int len) throws IOException`
 - ❑ `public void flush() throws IOException`
 - ❑ `public void close() throws IOException`
 - ❖ **System.out** là một đối tượng thuộc lớp `PrintStream` (thừa kế từ `OutputStream`), nối kết với màn hình.
 - ❑ Cho phép xuất ra màn hình nhiều dạng dữ liệu khác nhau.
 - ❑ Có thể sử dụng các phương thức của `OutputStream`.

Java – Nhập xuất với Stream



- Ví dụ về java.io.OutputStream

```
import java.io.*;
public class Ghi {
    public static void main(String args[]) {
        try {
            FileOutputStream f1 = new FileOutputStream("C:/Test1.txt");
            int ch='@'; f1.write(ch);
            byte b1[] = new byte[10];
            int m=0;
            for(int i= '0'; i<='9'; i++)    b1[m++]=(byte)i;
            f1.write(b1); m=0; f1.write("\r"); f1.write("\n");
            byte b2[] = new byte[50];
            for(int j= 'A'; j<='Z'; j++)    b2[m++]=(byte)j;
            f1.write(b2, 0, m);
            f1.close();
        }
        catch (IOException ie)
        {    System.out.println("Loi khi truy xuất file"); }
    }
}
```

Ghi 1 file với các dạng dữ liệu khác nhau

Nội dung file kết quả



Java – Nhập xuất với Stream



- Nhập từ bàn phím 1 chuỗi

- ❖ Cách 1: Sử dụng lớp InputStream

- ❑ Nhập vào 1 mảng byte[].
- ❑ Chuyển đổi từ mảng byte[] thành chuỗi.

```
byte b[] = new byte[100]; // Tạo vùng đệm cho mảng, dự kiến là không quá 100 ký tự
int n = System.in.read(b); // Nhập dữ liệu từ bàn phím và lưu vào mảng b[]
String str = new String(b, 0, n-2); // Chuyển đổi từ mảng b[] sang chuỗi
```

- ❖ Cách 2: Sử dụng lớp thừa kế BufferedReader

- ❑ Tạo ra 1 đối tượng BufferedReader từ System.in
- ❑ Sử dụng hàm readLine() để nhập 1 chuỗi (có xuống hàng).

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String str = br.readLine(); // Kết quả nhập từ bàn phím sẽ lưu trong chuỗi str.
```

Java – Nhập xuất với Stream

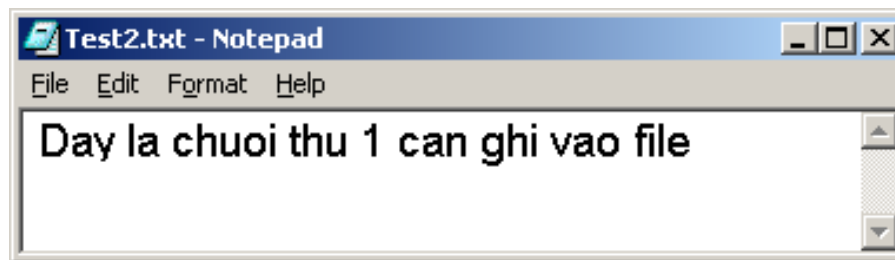


- Xuất ra 1 chuỗi

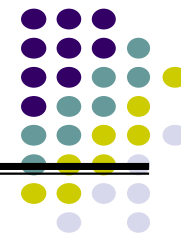
- ❖ Cách 1: Sử dụng lớp OutputStream

- ❑ Chuyển đổi từ chuỗi cần ghi vào mảng byte[].
- ❑ Ghi mảng byte[] ra stream xuất.

```
FileOutputStream f2 = new FileOutputStream("C:/Test2.txt");  
String str = "Day la chuoai thu 1 can ghi vao file \r\n";  
byte b[] = str.getBytes(); // Doi chuoai thanh mang byte[]  
f2.write(b);  
f2.close();
```

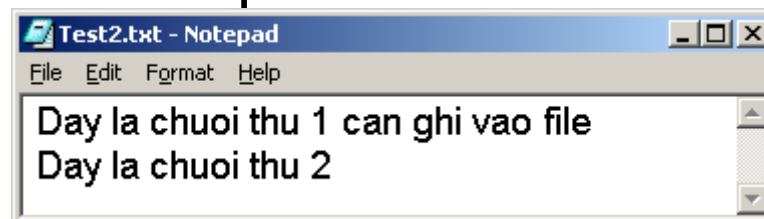


Java – Nhập xuất với Stream



- Xuất ra 1 chuỗi
 - ❖ Cách 2: Sử dụng lớp `PrintWriter`
 - Dùng các hàm `print`, `println`, ... để xuất nhiều dạng dữ liệu.
 - Gọi thêm lệnh `flush()` để đẩy dữ liệu từ vùng đệm ra ngoài.

```
import java.io.*;
public class Ghi1 {
    public static void main(String args[]) {
        try {
            FileOutputStream f2 = new FileOutputStream("C:/Test2.txt");
            PrintWriter pw = new PrintWriter(f2);
            String str1 = "Day la chuoai thu 1 can ghi vao file";
            String str2 = "Day la chuoai thu 2";
            pw.println(str1);    // Ghi chuoai str1, sau do xuong dong
            pw.print(str2);     // Ghi chuoai str2 khong xuong dong
            pw.flush();         // Day du lieu tu Stream vao ngoai vi
            f2.close();
        }
        catch (IOException ie)
        { System.out.println("Loi khi truy xuat file"); }
    }
}
```

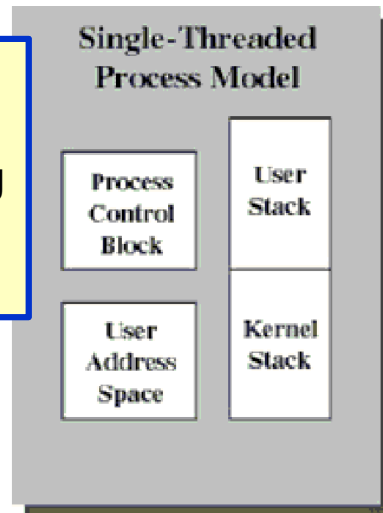


Luồng (Thread)



- Khái niệm:

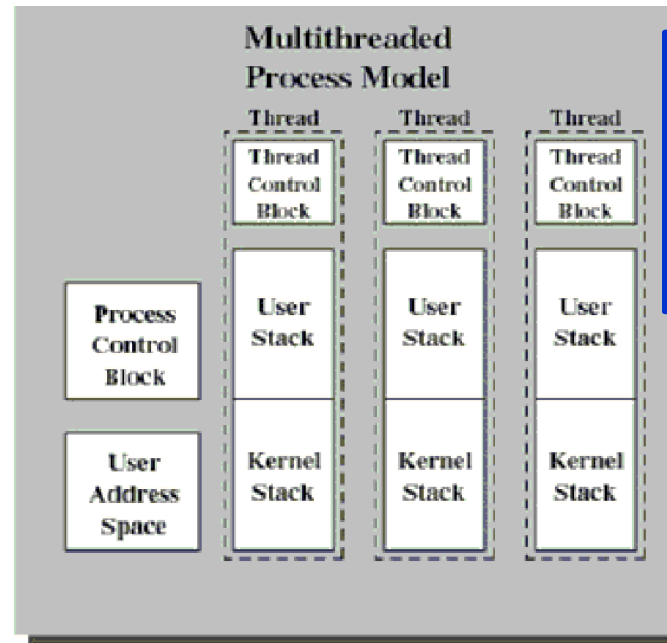
Trong hệ điều hành cổ điển
Đơn vị sử dụng CPU là **quá trình (process)**



Mỗi quá trình có:

- Thanh ghi bộ đếm chương trình
- Thanh ghi trạng thái
- Ngăn xếp
- Không gian địa chỉ riêng

Multithreaded Process Model



Trong hệ điều hành hiện đại
Đơn vị sử dụng CPU là **luồng (thread)**

Luồng cung cấp cơ chế **tính toán và xử lý song song** trong các ứng dụng.

Mỗi luồng:

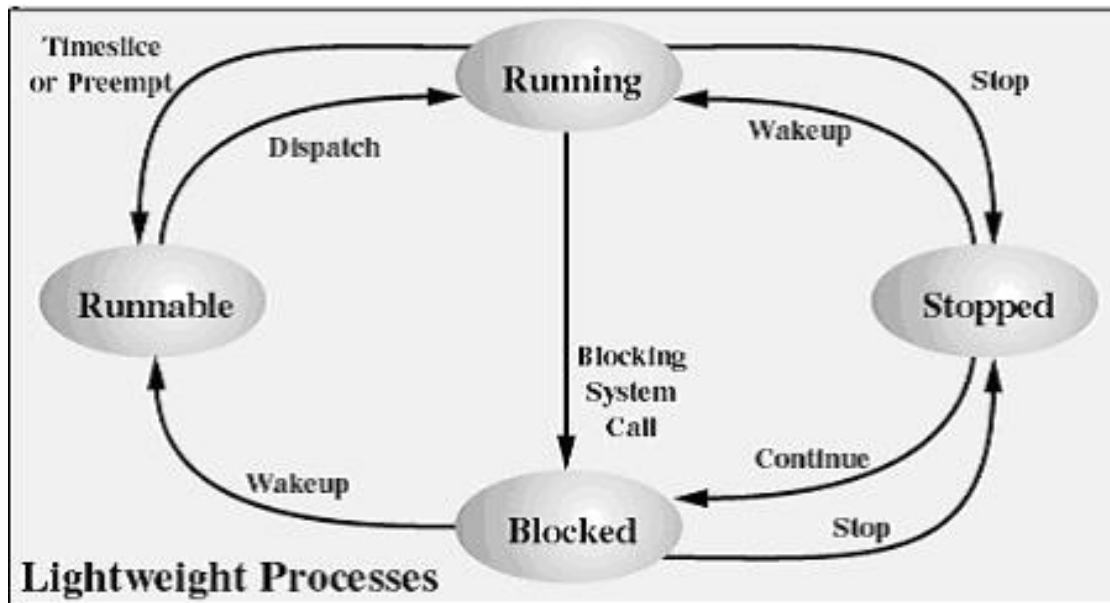
- Có riêng bộ đếm chương trình, trạng thái thanh ghi và ngăn xếp.
- **Chia sẻ không gian địa chỉ**
=> Dễ dàng chia sẻ file, hiệu báo

Luồng (Thread)



- Các trạng thái

- ❖ Tương tự trạng thái của 1 quá trình: đang chạy (**running**), sẵn sàng (**ready**), nghẽn (**blocked**) và kết thúc (**Dead**).
- ❖ Mỗi hệ điều hành sẽ có cách cài đặt luồng khác nhau => sẽ có sơ đồ chuyển trạng thái cũng khác nhau.

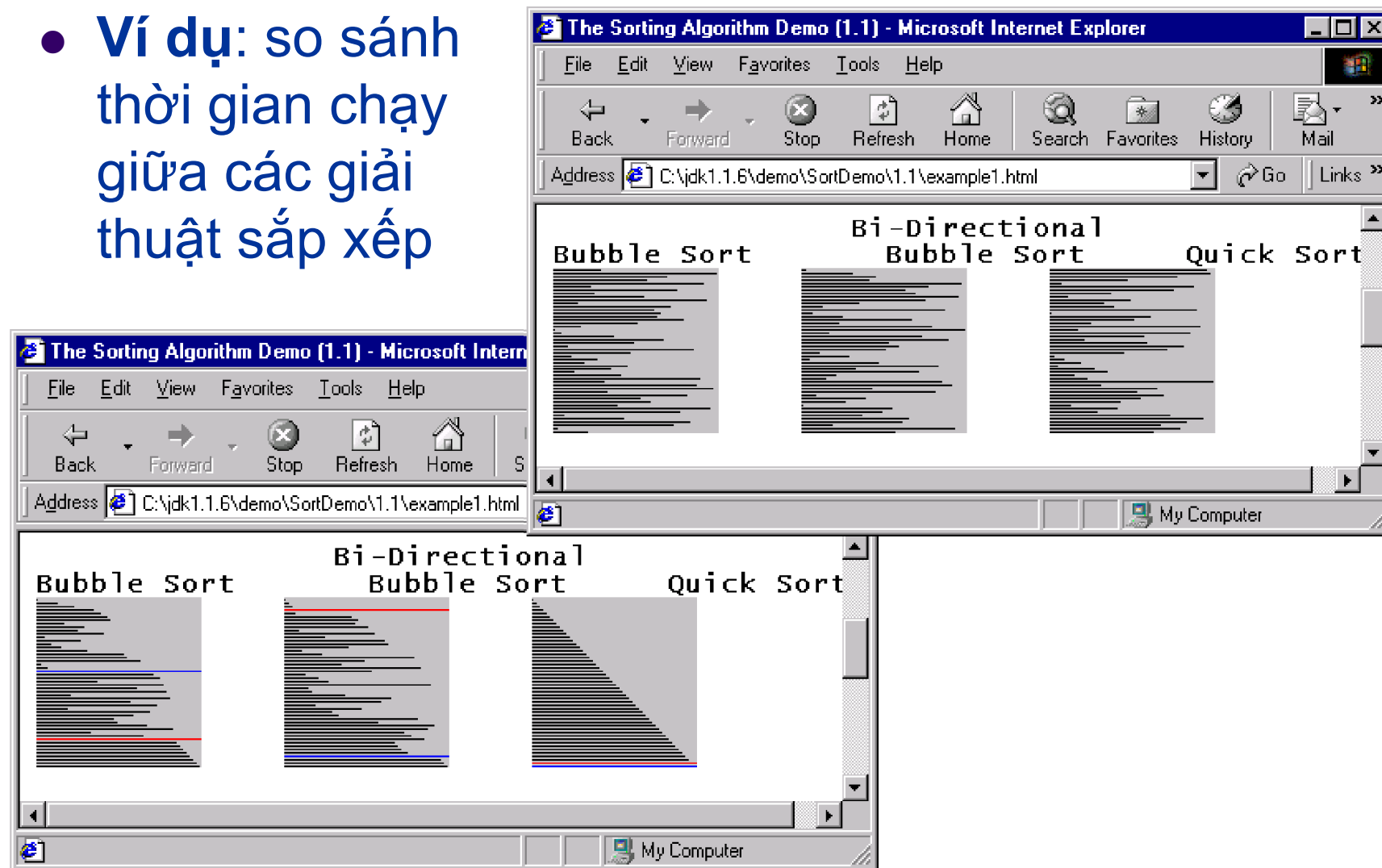


sơ đồ chuyển trạng thái của Thread sử dụng trong hệ điều hành Sun Solaris

Luồng (Thread)



- Ví dụ: so sánh thời gian chạy giữa các giải thuật sắp xếp



Luồng (Thread)



- Luồng trong Java

- ❖ Luồng là 1 đối tượng thuộc lớp **java.lang.Thread**.
- ❖ Một chương trình trong java cài đặt luồng bằng cách tạo ra một lớp con (extends) của lớp **java.lang.Thread**.
- ❖ **Thread** có 3 phương thức cơ bản để điều khiển:
 - ❑ `public native synchronized void start()`
 - ❑ `public void run()` : các công việc thực sự của luồng.
 - ❑ `public final void stop()`
- ❖ Luồng kết thúc khi:
 - ❑ Tất cả các công việc trong phương thức `run()` được thực hiện.
 - ❑ Hoặc phương thức `stop()` được kích hoạt.

Luồng (Thread)



- Ví dụ

Tạo ra 3 luồng thực thi song song việc đếm số từ 1 đến 50

```
public class MyThread extends Thread {
    String name;
    public MyThread(String ten) {
        name = ten;
        System.out.println("Thread "+name+" duoc khoi tao ... !");
    }
    public void run() {
        for(int i=1; i<=50; i++)
            System.out.print(name + "-" + i + "\t");
    }
    public static void main(String args[]){
        MyThread t1 = new MyThread("Cang");
        MyThread t2 = new MyThread("Phi");
        MyThread t3 = new MyThread("Hung");
        t1.start(); t2.start(); t3.start();
    }
}
```

```
C:\WINNT\system32\cmd.exe
D:\UduJava>java MyThread
Thread Cang duoc khoi tao ... !
Thread Phi duoc khoi tao ... !
Thread Hung duoc khoi tao ... !
Cang-1 Cang-2 Cang-3 Cang-4 Cang-5 Cang-6 Cang-7 Cang-8 Cang-9 Cang-10
Cang-11 Cang-12 Cang-13 Cang-14 Cang-15 Cang-16 Phi-1 Hung-1 Phi-2 Hung-2
Phi-3 Hung-3 Phi-4 Hung-4 Phi-5 Hung-5 Phi-6 Hung-6 Phi-7 Hung-7
Phi-8 Hung-8 Phi-9 Phi-10 Phi-11 Phi-12 Phi-13 Phi-14 Phi-15 Phi-16
Phi-17 Phi-18 Phi-19 Phi-20 Phi-21 Phi-22 Phi-23 Phi-24 Phi-25 Hung-9
Hung-10 Hung-11 Hung-12 Hung-13 Hung-14 Hung-15 Hung-16 Hung-17 Hung-18 Hung-19
Hung-20 Hung-21 Hung-22 Hung-23 Hung-24 Hung-25 Phi-26 Phi-27 Phi-28 Phi-29
Phi-30 Phi-31 Phi-32 Phi-33 Phi-34 Phi-35 Phi-36 Phi-37 Phi-38 Phi-39
Phi-40 Phi-41 Phi-42 Hung-26 Hung-27 Hung-28 Hung-29 Hung-30 Hung-31 Hung-32
Hung-33 Hung-34 Hung-35 Hung-36 Hung-37 Hung-38 Hung-39 Hung-40 Hung-41 Hung-42
Phi-43 Phi-44 Phi-45 Phi-46 Phi-47 Phi-48 Phi-49 Phi-50 Hung-43 Hung-44
Hung-45 Hung-46 Hung-47 Hung-48 Hung-49 Hung-50 Cang-17 Cang-18 Cang-19
Cang-21 Cang-22 Cang-23 Cang-24 Cang-25 Cang-26 Cang-27 Cang-28 Cang-29 Cang-30
Cang-31 Cang-32 Cang-33 Cang-34 Cang-35 Cang-36 Cang-37 Cang-38 Cang-39 Cang-40
Cang-41 Cang-42 Cang-43 Cang-44 Cang-45 Cang-46 Cang-47 Cang-48 Cang-49 Cang-50
D:\UduJava>
```

Luồng (Thread)



- Luồng trong Java:
 - ❖ Các phương thức khác của lớp **java.lang.Thread**
 - `public static void sleep(long m) throws InterruptedException:`
Blocked trong khoảng m ms
 - `public final void suspend():` chuyển từ Ready -> Blocked
 - `public final void resume():` chuyển từ Blocked -> Ready
 - `public final void yield():` chuyển từ Running -> Ready
 - ❖ Các phương thức thừa kế từ **java.lang.Object**
 - `public void wait(long m) throws InterruptedException:`
dừng Thread trong m ms xác định
 - `public final void notify():` đánh thức 1 Thread trong hàng đợi
 - `public final void notifyAll():` đánh thức tất cả các Thread trong hàng đợi.

Luồng (Thread)



- Độ ưu tiên của luồng

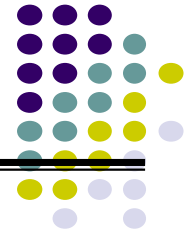
- ❖ `void setPriority(int x)`: Đặt độ ưu tiên của luồng là x
- ❖ `int getPriority()`: Trả về giá trị ưu tiên của luồng.

```
Thread.MAX_PRIORITY = 10
Thread.NORM_PRIORITY = 5
Thread.MIN_PRIORITY = 1
```

```
public static void main(String args[]){
    MyThread t1 = new MyThread("Cang");
    MyThread t2 = new MyThread("Phi");
    MyThread t3 = new MyThread("Hung");
    t1.setPriority(Thread.MIN_PRIORITY);
    t2.setPriority(Thread.NORM_PRIORITY);
    t3.setPriority(Thread.MAX_PRIORITY);
    t1.start(); t2.start(); t3.start();
}
```

```
C:\WINNT\system32\cmd.exe
D:\UduJava>java MyThread
Thread Cang duoc khoi tao ... ?
Thread Phi duoc khoi tao ... ?
Thread Hung duoc khoi tao ... ?
Hung-1 Hung-2 Hung-3 Hung-4 Hung-5 Hung-6 Hung-7 Hung-8 Hung-9 Hung-10
Hung-11 Hung-12 Hung-13 Hung-14 Hung-15 Hung-16 Hung-17 Hung-18 Hung-19 Hung-20
Hung-21 Hung-22 Hung-23 Hung-24 Hung-25 Hung-26 Hung-27 Hung-28 Hung-29 Hung-30
Hung-31 Hung-32 Hung-33 Hung-34 Hung-35 Hung-36 Hung-37 Hung-38 Hung-39 Hung-40
Hung-41 Hung-42 Hung-43 Hung-44 Hung-45 Hung-46 Hung-47 Hung-48 Hung-49 Hung-50
Phi-1 Phi-2 Phi-3 Phi-4 Phi-5 Phi-6 Phi-7 Phi-8 Phi-9 Phi-10
Phi-11 Phi-12 Phi-13 Phi-14 Phi-15 Phi-16 Phi-17 Phi-18 Phi-19 Phi-20
Phi-21 Phi-22 Phi-23 Phi-24 Phi-25 Phi-26 Phi-27 Phi-28 Phi-29 Phi-30
Phi-31 Phi-32 Phi-33 Phi-34 Phi-35 Phi-36 Phi-37 Phi-38 Phi-39 Phi-40
Phi-41 Phi-42 Phi-43 Phi-44 Phi-45 Phi-46 Phi-47 Phi-48 Phi-49 Phi-50
Cang-1 Cang-2 Cang-3 Cang-4 Cang-5 Cang-6 Cang-7 Cang-8 Cang-9 Cang-10
Cang-11 Cang-12 Cang-13 Cang-14 Cang-15 Cang-16 Cang-17 Cang-18 Cang-19 Cang-20
Cang-21 Cang-22 Cang-23 Cang-24 Cang-25 Cang-26 Cang-27 Cang-28 Cang-29 Cang-30
Cang-31 Cang-32 Cang-33 Cang-34 Cang-35 Cang-36 Cang-37 Cang-38 Cang-39 Cang-40
Cang-41 Cang-42 Cang-43 Cang-44 Cang-45 Cang-46 Cang-47 Cang-48 Cang-49 Cang-50
D:\UduJava>
```

Luồng (Thread)



- Đồng bộ hóa các luồng

- ❖ Trong 1 thời điểm, chỉ có 1 Thread thực thi được phương thức có khai báo thuộc tính **synchronized**

```
class TaiKhoan {  
    private String sotaikhoan;  
    private double sodu; // so du trong tai khoan  
    public TaiKhoan(String sotk)  
    { sotaikhoan = sotk; }  
    public synchronized void GanSoDu(double s) {  
        sodu = s;  
    }  
    public synchronized double LaySoDu() {  
        return sodu;  
    }  
    synchronized boolean CapNhatSoDu(double sotien) {  
        if(LaySoDu() + sotien >= 0)  
        { GanSoDu(LaySoDu() + sotien); return true; }  
        else  
        return false;  
    }  
}
```