



Java™

LẬP TRÌNH JAVA 2

BÀI 4: NGOẠI LỆ

PHẦN 1

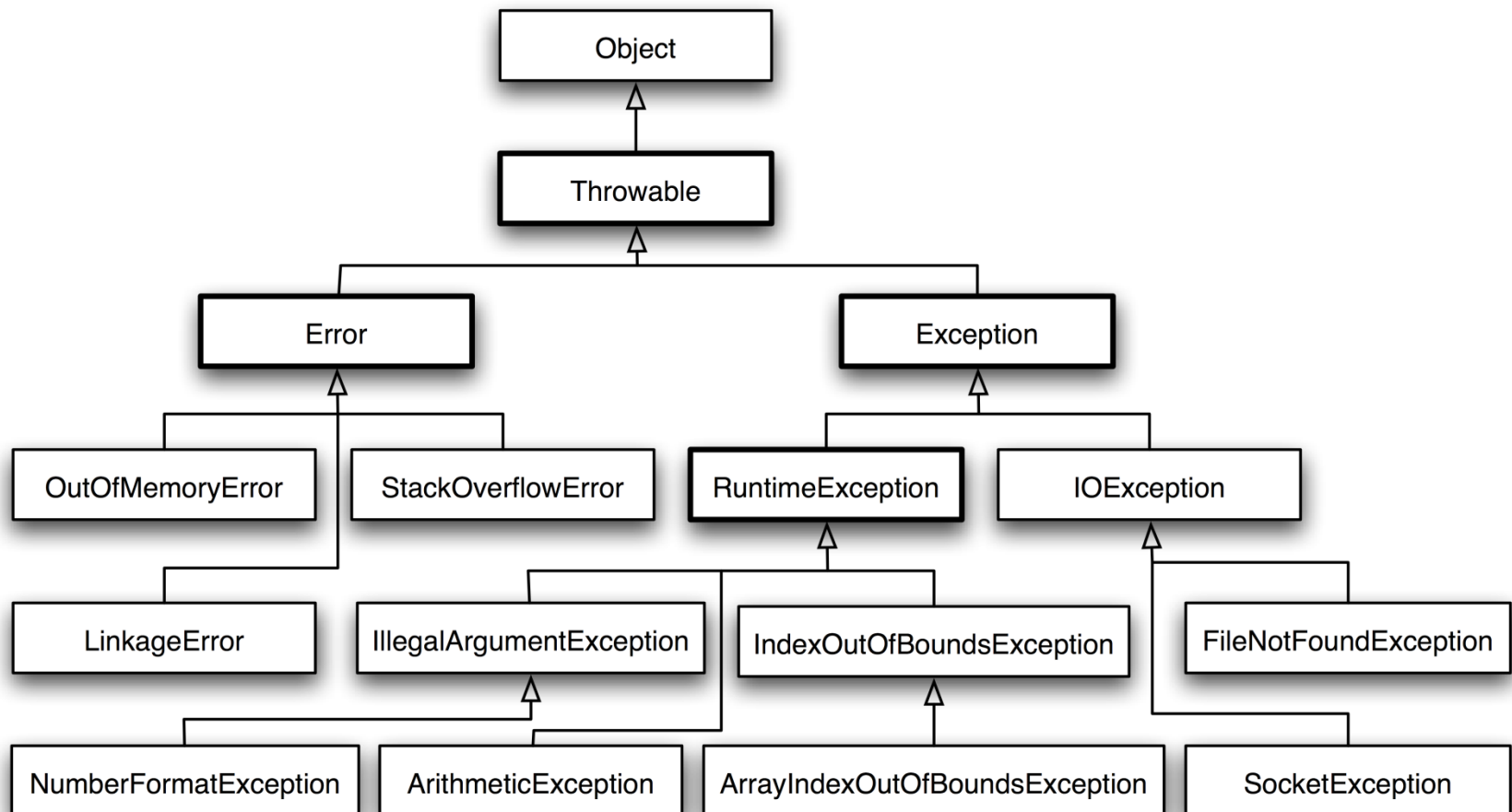
- ⊙ Giải thích được ngoại lệ
- ⊙ Phân loại được ngoại lệ
- ⊙ Sử dụng khối try...catch để xử lý ngoại lệ
- ⊙ Sử dụng finally để xử lý sau try...catch
- ⊙ Sử dụng throws để cho phép quảng ngoại lệ ra ngoài phương thức
- ⊙ Sử dụng throw để phát sinh ngoại lệ
- ⊙ Tạo lớp ngoại lệ mới



- ❑ Có những lỗi chỉ khi chạy chương trình mới xuất hiện và chương trình đang chạy lập tức ngừng lại và xuất hiện thông báo lỗi – đó chính là ngoại lệ (exception).
- ❑ Ví dụ: Xét chương trình chia 2 số. Nếu ta cho mẫu số = 0 thì phát sinh lỗi và đó được coi là 1 ngoại lệ.

```
int a = 5, b = 0;  
a/b  $\Rightarrow$  error !
```

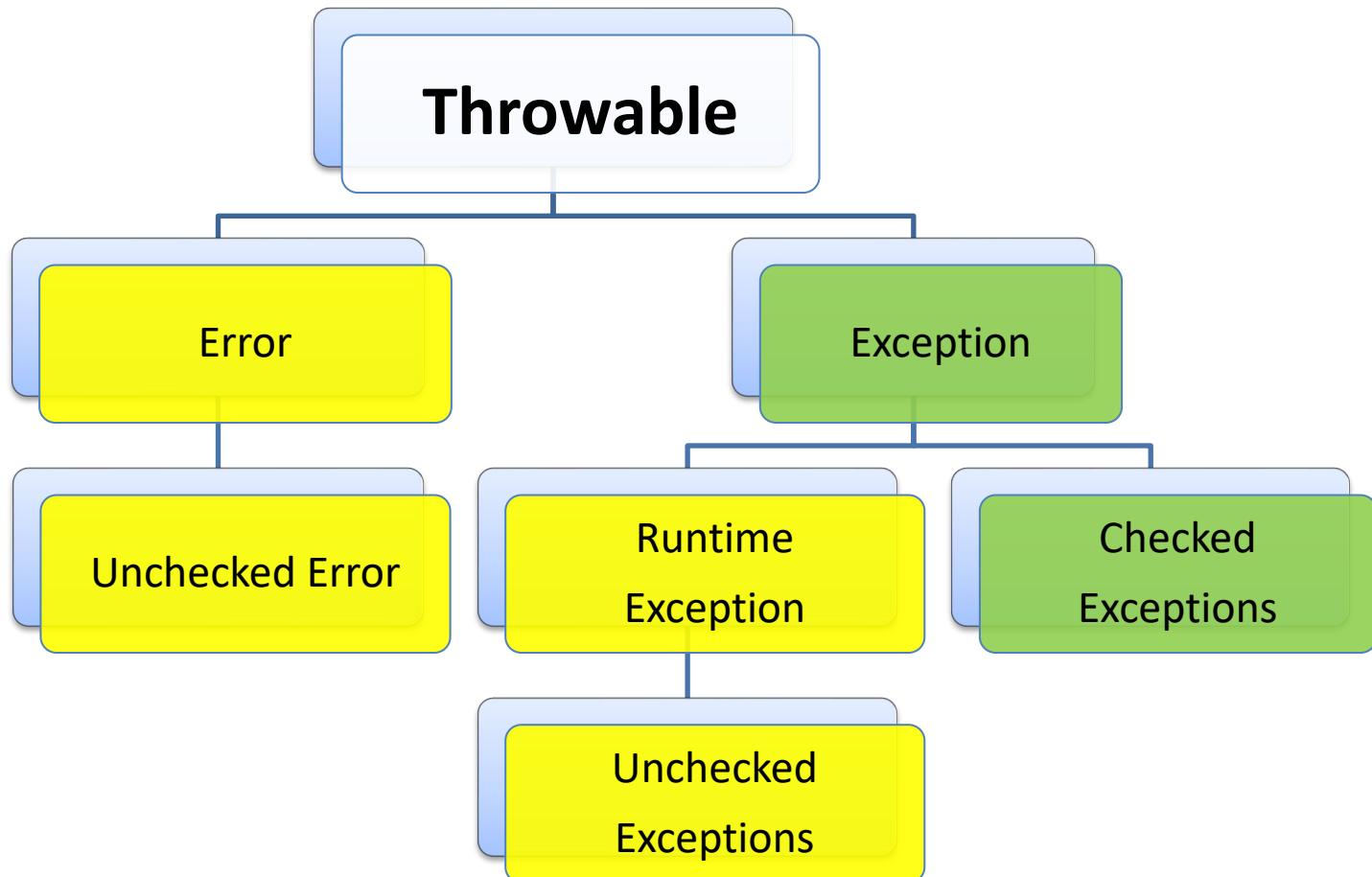
- ❑ Class Throwable xử lý lỗi và ngoại lệ (Error, Exception).



❑ Chuyển đổi chuỗi sang số

```
try{  
    int a = Integer.parseInt(string);  
    System.out.println("Thành công");  
}  
  
catch (Exception ex) {  
    System.out.println("Lỗi");  
}
```

- ❑ Exception chia làm 2 loại là checked (xanh) và unchecked (vàng)



❑ Ngoại lệ 'unchecked':

- ❖ Là các ngoại lệ được kiểm tra lúc chạy
- ❖ Bao gồm các class Error, RuntimeException và các lớp con của chúng
- ❖ Ví dụ: **Integer.parseInt("abc")** vẫn dịch được nhưng chạy lỗi.

❑ Ngoại lệ 'checked':

- ❖ Là các ngoại lệ được kiểm tra lúc dịch
- ❖ Bao gồm các class exception còn lại
- ❖ Ví dụ: **new FileWriter("c:/data.txt")** dịch lỗi dù file đã tồn tại

❑ Một số ngoại lệ 'checked':

- ❖ ClassNotFoundException
- ❖ IOException
 - FileNotFoundException
 - EOFException

❑ Một số ngoại lệ 'unchecked'

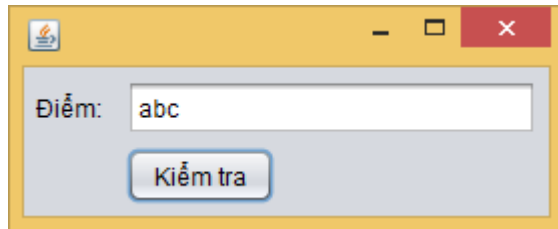
- ❖ ArithmeticException
- ❖ IllegalArgumentException
- ❖ IndexOutOfBoundsException
- ❖ NullPointerException
- ❖ InputMismatchException

❑ Sử dụng lệnh try...catch để xử lý các ngoại lệ

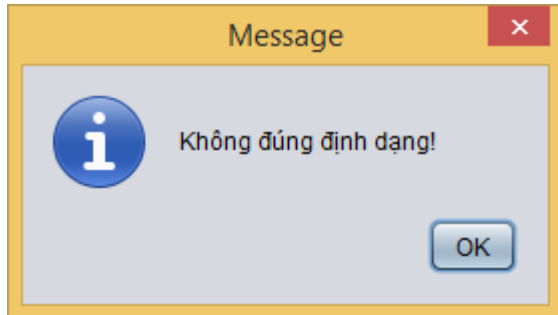
```
try{  
    //Khởi lệnh  
}  
  
catch (...) {  
    //Khởi lệnh xử lý ngoại lệ  
}
```

❑ Ví dụ sau xử lý lỗi chuyển chuỗi sang số nguyên


```
try{  
    int a = Integer.parseInt(s)  
}  
  
catch (Exception ex) {  
    System.out.println("Lỗi");  
}
```

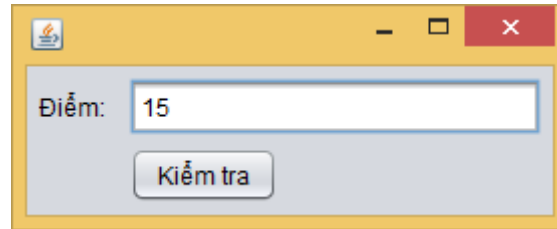


Điểm:

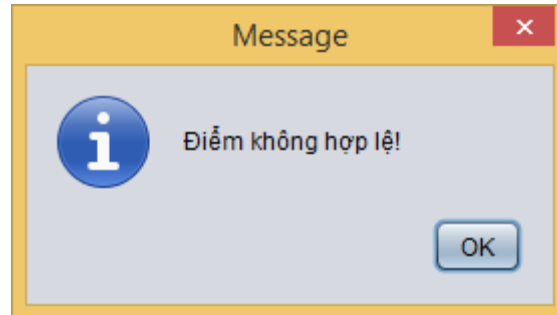


Message


 Không đúng định dạng!

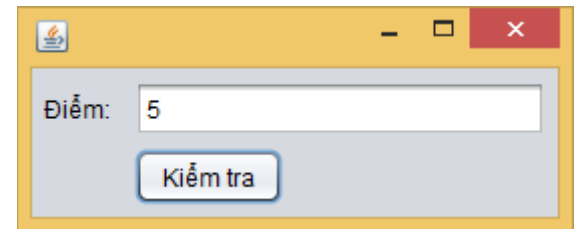


Điểm:

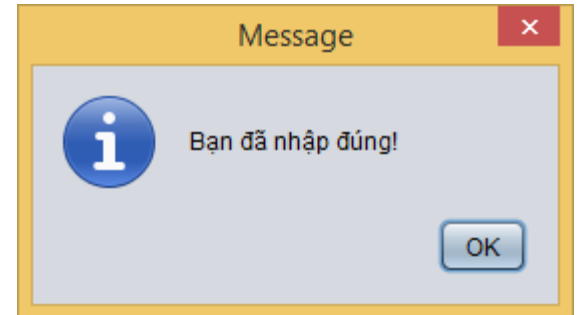


Message


 Điểm không hợp lệ!



Điểm:



Message

 Bạn đã nhập đúng!

```
String text = txtDiem.getText();
try {
    double diem = Double.parseDouble(text);
    if(diem < 0 || diem > 10){
        JOptionPane.showMessageDialog(this, "Điểm không hợp lệ!");
    }
    else{
        JOptionPane.showMessageDialog(this, "Bạn đã nhập đúng!");
    }
}
catch (Exception e) {
    JOptionPane.showMessageDialog(this, "Không đúng định dạng!");
}
```



DEMO



Hiện thực hóa kiểm lỗi

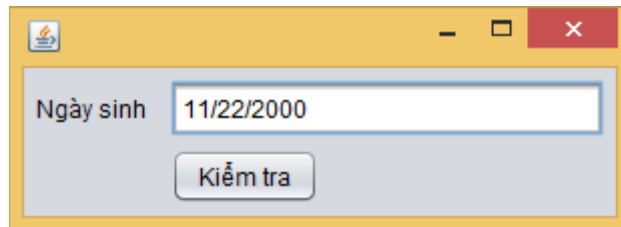
- ❑ Giả sử có mảng chuỗi ss. Lệnh sau đây có thể xảy ra những lỗi nào?
 - ❖ `Integer.parseInt(ss[5]);`
- ❑ Rõ ràng có khả năng xảy ra 3 lỗi
 - ❖ Mảng ss null (chưa được khởi tạo)
 - ❖ Mảng ss ít hơn 6 phần tử
 - ❖ Phần tử thứ 6 (ss[5]) không thể chuyển sang số
- ❑ Xử lý các lỗi này thế nào?

❑ Khối mã try có thể có nhiều ngoại lệ xảy ra. Sử dụng **nhiều khối catch** để bắt và xử lý chi tiết các ngoại lệ đó.

```
String[] ss = {"1", "a", "2"};
try {
    int a = Integer.parseInt(ss[1]);
}
catch (NumberFormatException e1) {
    System.out.println("Không đúng định dạng số !");
}
catch (ArrayIndexOutOfBoundsException e2) {
    System.out.println("Ngoài phạm vi mảng !");
}
catch (NullPointerException e3) {
    System.out.println("Mảng chưa được khởi tạo !");
}
```

- ❑ Catch thứ 2 của đoạn mã sau bắt chung cả 2 ngoại lệ `NumberFormatException` và `NullPointerException` do cả 2 ngoại lệ này đều là con của `Exception`
- ❑ Catch bắt ngoại lệ chung phải đặt sau cùng

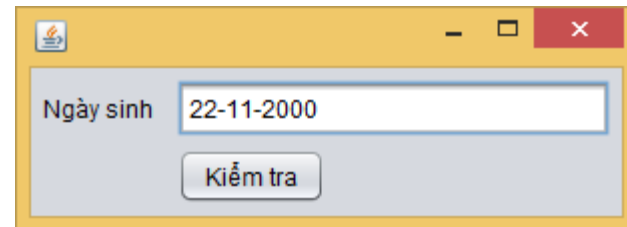
```
String[] ss = {"1", "a", "2"};
try {
    int a = Integer.parseInt(ss[1]);
}
catch (NumberFormatException e1) {
    System.out.println("Không đúng định dạng số!");
}
catch (Exception e2) {
    System.out.println("Lỗi chuyển đổi số!");
}
```

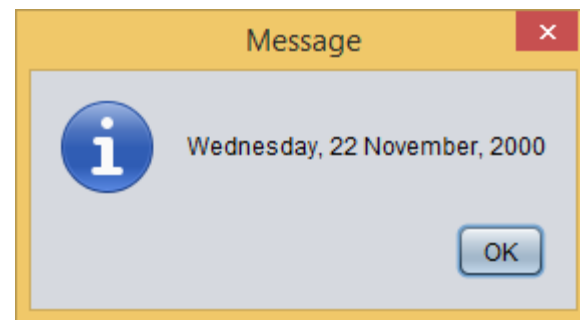
Form input field labeled "Ngày sinh" (Date of birth) containing the text "11/22/2000". Below the field is a button labeled "Kiểm tra" (Check).



Message box titled "Message" with a blue information icon. The text inside says "Không đúng định dạng!" (Incorrect format!). There is an "OK" button at the bottom right.



Form input field labeled "Ngày sinh" (Date of birth) containing the text "22-11-2000". Below the field is a button labeled "Kiểm tra" (Check).



Message box titled "Message" with a blue information icon. The text inside says "Wednesday, 22 November, 2000". There is an "OK" button at the bottom right.

```
SimpleDateFormat formater = new SimpleDateFormat();

String text = txtNgaySinh.getText();
try {
    formater.applyPattern("dd-MM-yyyy");
    Date date = formater.parse(text);

    formater.applyPattern("EEEE, dd MMMM, yyyy");
    String text2 = formater.format(date);

    JOptionPane.showMessageDialog(this, text2);
}
catch (Exception e) {
    JOptionPane.showMessageDialog(this, "Không đúng định dạng!");
}
```



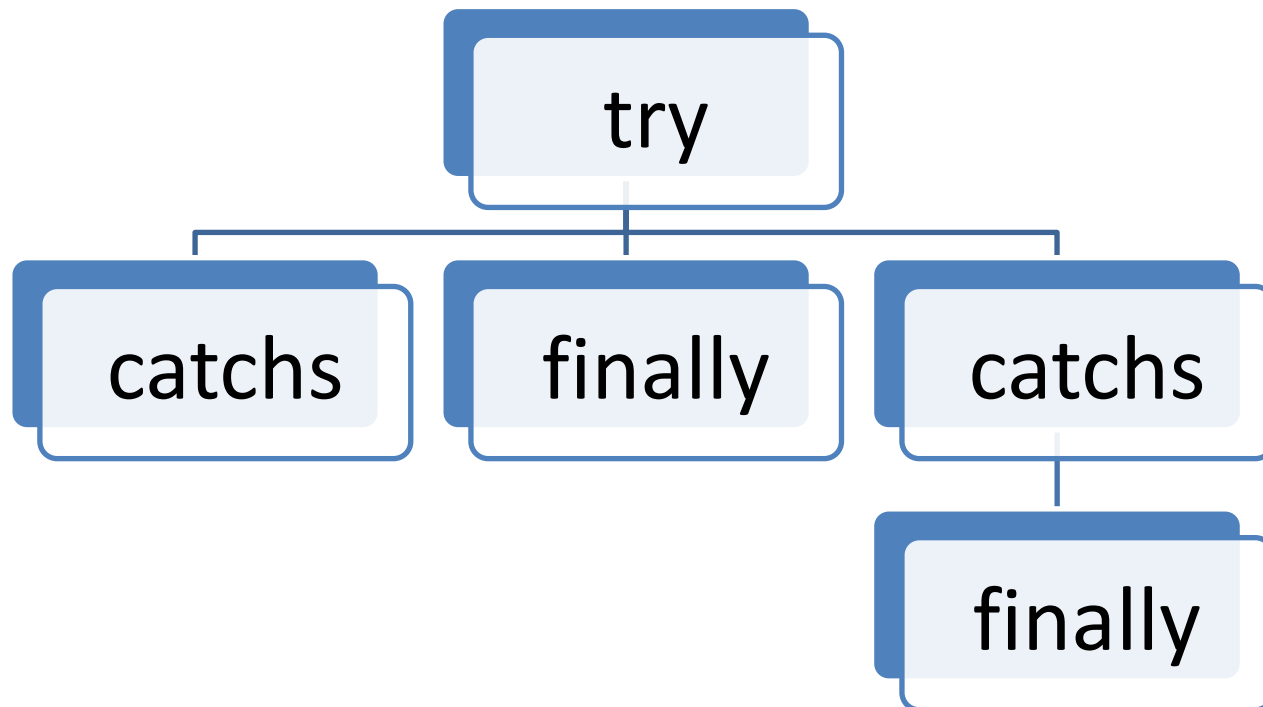
Java™

LẬP TRÌNH JAVA 2

BÀI 4: NGOẠI LỆ

PHẦN 2

- ❑ Mỗi khối **try** yêu cầu có ít nhất một khối **catch** hoặc/và duy nhất một khối **finally**.
- ❑ Khối **finally** sẽ được thực hiện dù ngoại lệ có xuất hiện hay không.



```
try {  
    int a = Integer.parseInt(ss[1]);  
}  
catch (Exception e) {  
    System.out.println("Lỗi!");  
}
```

```
try {  
    int a = Integer.parseInt(ss[1]);  
}  
finally {  
    ss = null;  
}
```

Ngoại lệ đã được xử lý

```
try {  
    int a = Integer.parseInt(ss[1]);  
}  
catch (Exception e) {  
    System.out.println("Lỗi!");  
}  
finally {  
    ss = null;  
}
```

Ngoại lệ chưa được xử lý

- ❑ throw được sử dụng để phát sinh một ngoại lệ
 - ❖ **throw new RuntimeException("Lỗi");**
- ❑ throws được sử dụng để quăng ngoại lệ ra ngoài phương thức. Ngoại lệ sẽ được xử lý khi gọi phương thức
 - ❖ void method() **throws FileNotFoundException**{...}

```
public void ghifile() throws IOException{
    FileWriter file = new FileWriter("data.txt");
    file.write("Xu ly ngoai le trong java");
    file.write(100);
    System.out.println("Da ghi xong !");
    file.close();
}

try {
    throwsexampel obj = new throwsexampel();
    obj.ghifile();
    System.out.println("Su dung tu khoa throws");
} catch (IOException ex) {
    System.out.println("Co loi: "+ex);
}
```

Phải bắt ngoại lệ khi gọi phương thức ghifile()

- ❑ Nếu khi gọi hàm có ngoại lệ mà chưa muốn bắt thì có thể tiếp tục quăng ra ngoài

```
public void ghifile() throws IOException{
    FileWriter file = new FileWriter("data.txt");
    file.write("Xu ly ngoai le trong java");
    file.write(100);
    System.out.println("Da ghi xong !");
    file.close();
}

public static void main(String[] args) throws IOException {
    throwsexampel obj = new throwsexampel();
    obj.ghifile();
    System.out.println("Su dung tu khoa throws");
}
}
```


- ❑ Thông thường các exception sẽ được 'ném' ra bởi hệ thống Java runtime. Tuy vậy ta vẫn có thể lập trình để 'ném' ra các ngoại lệ khi gặp một tình huống nào đó trong khi lập trình.
- ❑ Trong một phương thức có thể throw nhiều ngoại lệ.
- ❑ Có 2 cách để 'ném' (throw) ra các ngoại lệ:
 - ❖ Dùng toán tử new
 - ❖ Đưa 1 tham số vào mệnh đề catch.
- ❑ Ví dụ:
if (check == 0)
 throw new NullPointerException();

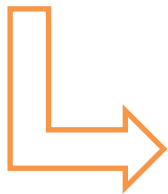
```
static void demoProc() {  
    try {  
        throw new NullPointerException("demo");  
    } catch (NullPointerException e) {  
        System.out.println("Ben trong xu ly ngoai le demoPro");  
        throw e;  
    }  
}  
  
public static void main(String args[]) {  
    try {  
        demoProc();  
    } catch (NullPointerException e) {  
        System.out.println("Trong main, tiep tuc xu ly ngoai le");  
    }  
}
```

- ❑ Chúng ta có thể tự viết class xử lý ngoại lệ của riêng mình bằng cách kế thừa một class Exception nào đó (checked hoặc unchecked)

```
public class myexception extends Exception {  
    private int message;  
    myexception(int a) {  
        message = a;  
    }  
    @Override  
    public String toString() {  
        return "My exception " + message;  
    }  
}
```

- ❑ Sau khi đã tạo Exception, chúng ta có thể sử dụng như các Exception được định nghĩa sẵn

```
static void tinhtoan(int a) throws myexception {  
    if (a > 10) {  
        throw new myexception(a);  
    }  
    System.out.println("Normal exit");  
}
```



```
try {  
    tinhtoan(1);           //không tạo ra exception  
    tinhtoan(20);          //tạo ra exception  
} catch (myexception e) {  
    System.out.println("Caught " + e);  
}
```

- ☑ Giải thích được ngoại lệ
- ☑ Phân loại được ngoại lệ
- ☑ Sử dụng khối try...catch để xử lý ngoại lệ
- ☑ Sử dụng final để xử lý sau try...catch
- ☑ Sử dụng throws để cho phép quảng ngoại lệ ra ngoài phương thức
- ☑ Sử dụng throw để phát sinh ngoại lệ
- ☑ Tạo lớp ngoại lệ mới

