



# NỘI DUNG

## LẬP TRÌNH HĐT

*Chương 1: Tổng quan về lập trình HĐT*

*Chương 2: Giới thiệu về lập trình Java*

*Chương 3: Lập trình Java cơ sở*

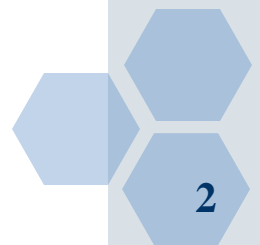
*Chương 4: Lập trình hướng đối tượng với Java*

*Chương 5: Xử lý ngoại lệ*



# XỬ LÝ NGOẠI LỆ

- ❖ Giới thiệu
- ❖ Mô hình xử lý ngoại lệ
- ❖ Sử dụng các khối “try”, “catch” và “finally”
- ❖ Sử dụng các từ khoá “throw” và “throws”.
- ❖ Định nghĩa mới một ngoại lệ





# Giới thiệu

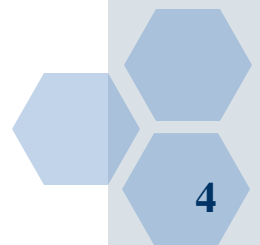
- ❖ **Một chương trình được viết ra thường gặp các lỗi sau:**
  - **Lỗi cú pháp:** Do lập trình viên không tuân thủ theo đúng cú pháp của ngôn ngữ lập trình và được trình biên dịch hỗ trợ sửa lỗi. VD: `int a=(5+2;`
  - **Lỗi ngữ nghĩa:** sử dụng câu lệnh không đúng cách.  
VD: `char c="A";`
  - **Lỗi logic:** Chương trình cho ra kết quả sai hoặc không mong muốn. VD: `int cong(int a, int b){return a*b;}`
- ❖ **Để giúp cho lập trình viên có thể kiểm soát và quản lý được các lỗi compiler/runtime. Java cung cấp cơ chế “Ngoại lệ và xử lý ngoại lệ”.**





## ❖ Giới thiệu về ngoại lệ (Exception)

- Ngoại lệ là một loại lỗi đặc biệt
- Lỗi này xuất hiện vào lúc thực thi chương trình
- Các trạng thái không bình thường xảy ra trong khi thực thi chương trình tạo ra các ngoại lệ.





# Khái niệm ngoại lệ (Exception)

- ❖ Exception là gì?
- ❖ Exception là một sự kiện xảy ra trong quá trình thực thi chương trình, nó phá vỡ luồng bình thường của chương trình.
- ❖ Ví dụ: việc nhập dữ liệu không hợp lệ, một tệp tin cần được mở ra nhưng không tìm thấy tệp tin đó hay phép chia 1 số cho 0 ...





# Nguyên nhân phát sinh ngoại lệ

- ❖ Lỗi chương trình
- ❖ Lỗi phía trình khách
- ❖ Lỗi do điều khiển của chương trình
- ❖ ...

❖ VD:

```
int a=10;
```

```
int b=0;
```

```
System.out.println(a/b);
```



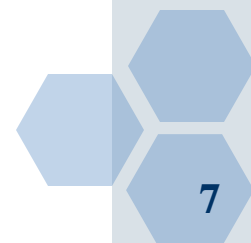
**Kết quả ?**





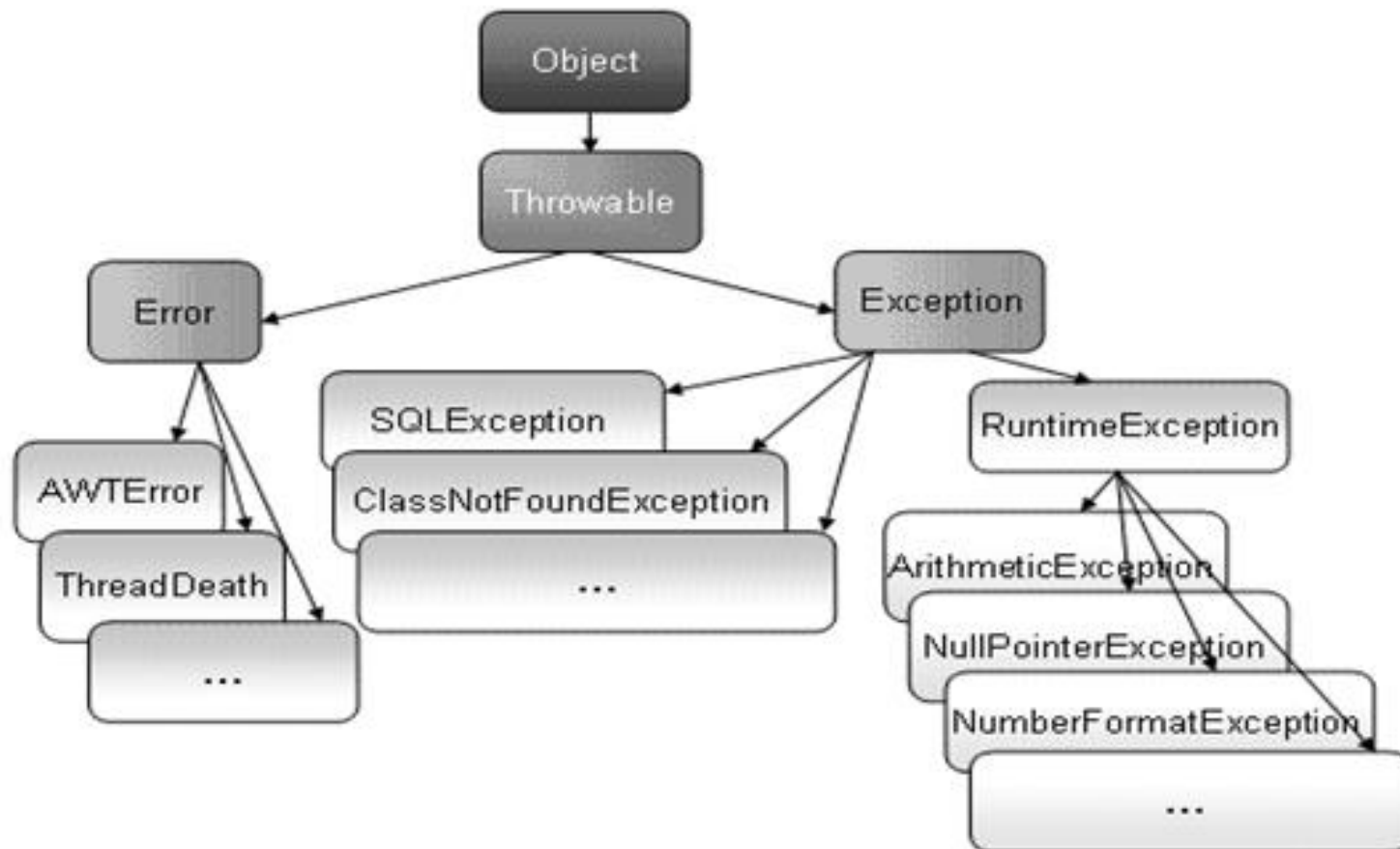
# Mục đích của việc xử lý ngoại lệ

- ❖ Ngăn việc chương trình bị ngắt đột ngột khi ngoại lệ xảy ra.
- ❖ Giải phóng tài nguyên hệ thống cấp phát khi ngoại lệ xảy ra.
- ❖  $\Rightarrow$  Cần có một cơ chế xử lý ngoại lệ thích hợp
- ❖ VD: Xét thao tác vào ra (I/O) trong 1 tệp tin.
- ❖ Java cung cấp cho ta cơ chế xử lý ngoại lệ  $\Rightarrow$  là ngôn ngữ mạnh.





# Hệ thống phân cấp các lớp Exception

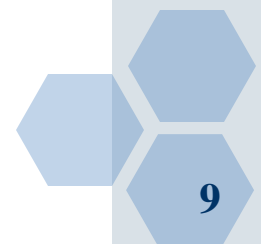






# Lớp Throwable

- ❖ Một biến kiểu String để lưu thông tin chi tiết về ngoại lệ đã xảy ra.
- ❖ Một số phương thức cơ bản:
  - new Throwable(String s): tạo 1 ngoại lệ với thông tin về ngoại lệ là s.
  - String getMessage(): Lấy thông tin về ngoại lệ
  - String toString(): Mô tả ngắn gọn về ngoại lệ
  - void printStackTrace(): In ra tất cả các thông tin liên quan đến ngoại lệ (tên, loại, vị trí...)





# Ngoại lệ được kiểm tra Checked Exceptions

- ❖ Gồm tất cả các lỗi cú pháp và một số lỗi ngữ nghĩa, chúng sẽ được phát hiện bởi trình biên dịch Java
- ❖ Class Exception và các lớp con của nó là Checked Exceptions. Ngoại trừ class RuntimeException- các lớp con của RuntimeException và class Error – các lớp con của Error.
- ❖ Việc kiểm tra và xác định Exceptions được thực hiện ngay tại thời điểm compile time.
- ❖ IDE(Netbean, Eclipse) giúp ta hiển thị lỗi cú pháp nếu gọi 1 phương thức throw.
- ❖ Một số checked exception tiêu biểu như: IOException, InterruptedException, XMLParseException...



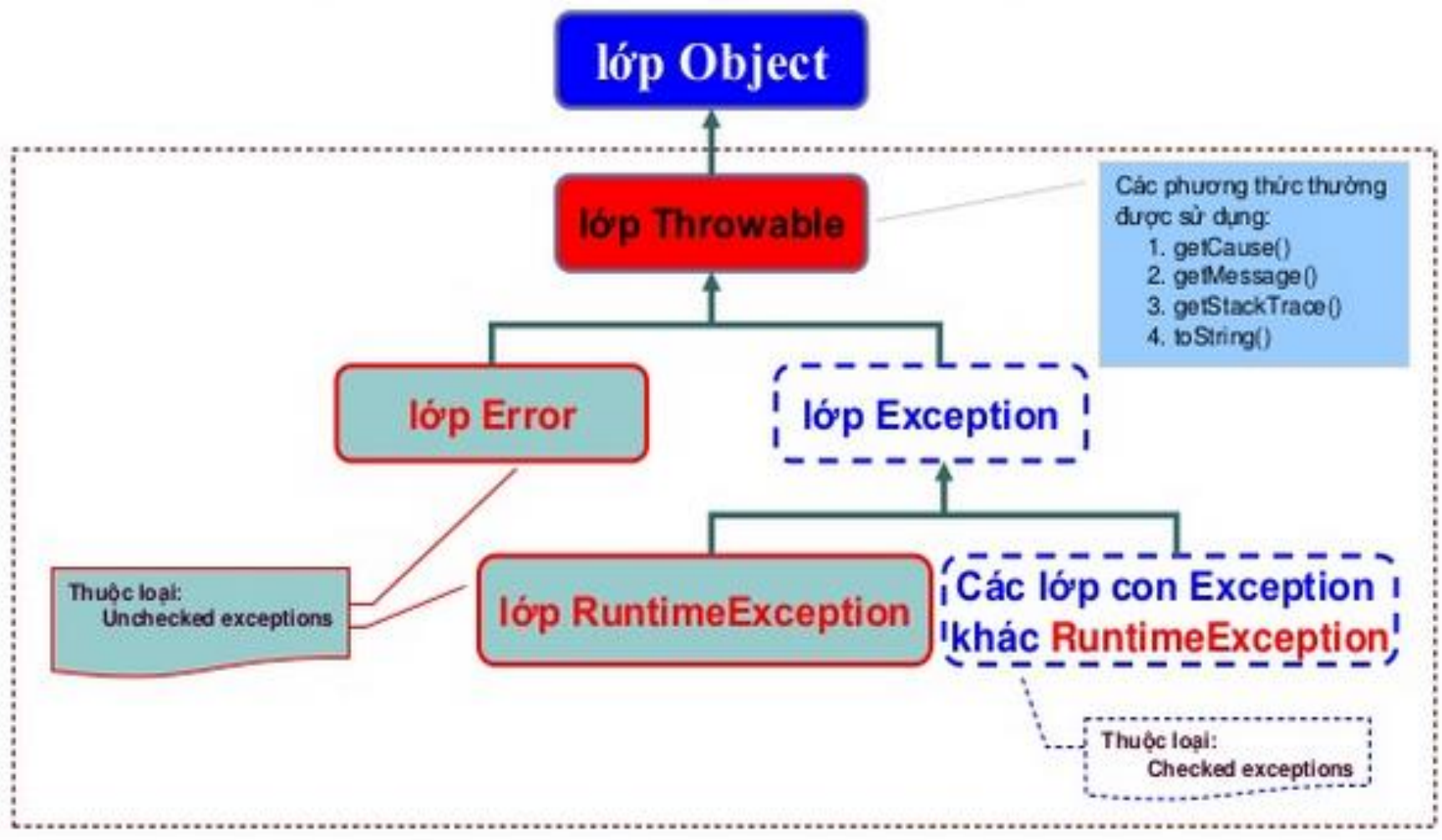
# Ngoại lệ không được kiểm tra Unchecked Exception

- ❖ Là những exceptions được sinh ra do lỗi logic và các lỗi ngữ nghĩa và chỉ được phát hiện khi chạy chương trình (runtime).
- ❖ Lớp RuntimeException và tất cả các lớp con của lớp RuntimeExceptions đều là Unchecked Exception.
- ❖ Việc xác định có exception hay không chỉ có thể thực hiện ở thời điểm runtime.
- ❖ Các IDE không hỗ trợ chúng ta việc xác định các exception.
- ❖ Một số unchecked tiêu biểu: NullPointerException, IndexOutOfBoundsException, ClassCastException...





# Các loại ngoại lệ được ném lên thông qua lớp Throwable



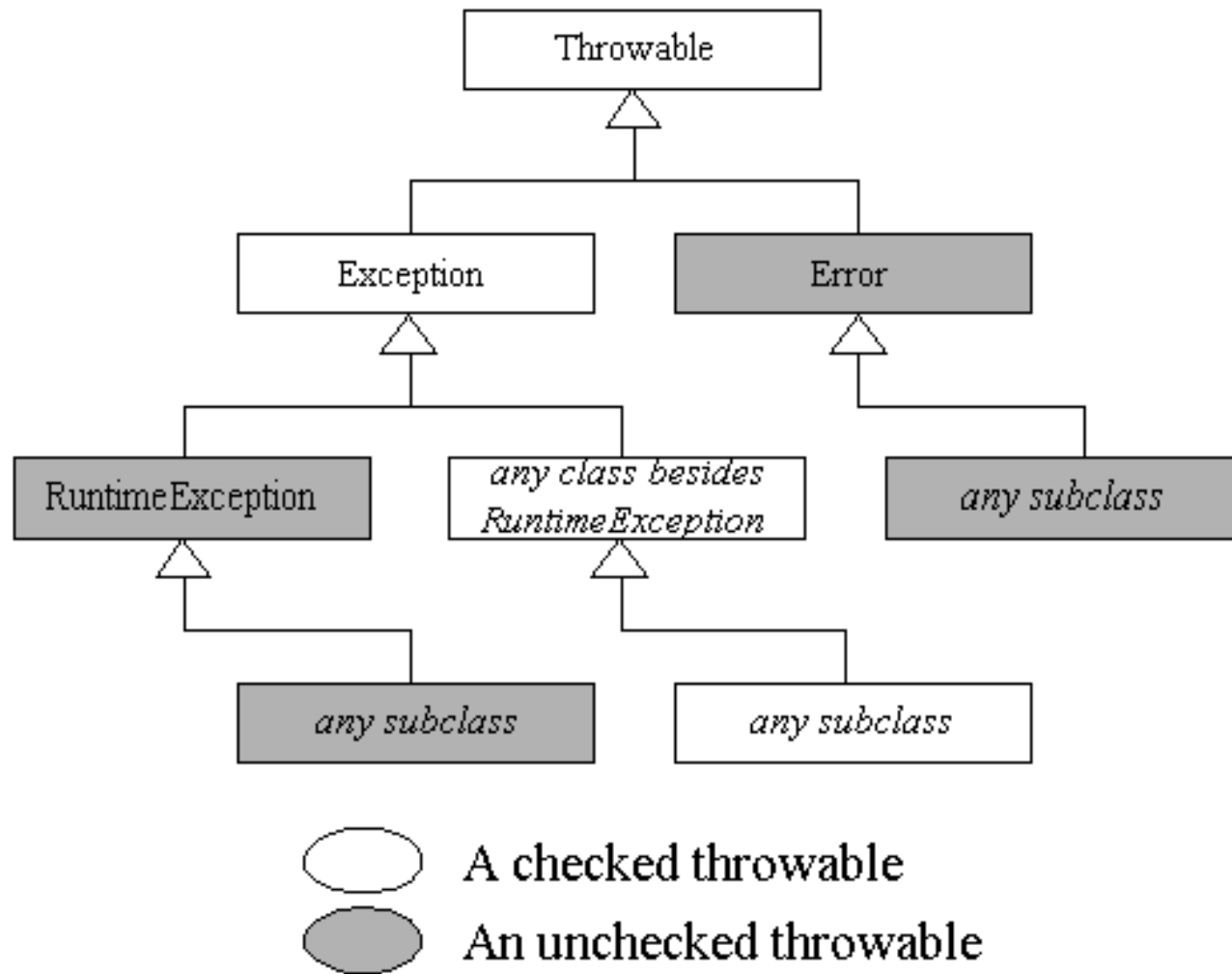


# Error (lỗi)

- ❖ Là vấn đề phát sinh ngoài sự kiểm soát của người sử dụng hoặc các lập trình viên.
- ❖ VD: việc ngăn xếp bị tràn sẽ phát sinh error, hoặc việc mở 2 tệp tin thành công nhưng không đọc được nội dung do ổ cứng bị hỏng 1 vài sector...
- ❖ Không thể phục hồi lại chương trình khi có lỗi xảy ra. Chương trình nên kết thúc.
- ❖ VD: `OutOfMemoryError`, `StackOverflowError`...
- ❖ Lớp `Error` và các lớp con của nó không được kiểm tra tại thời điểm biên dịch nên cũng được coi là `Unchecked Exceptions`.

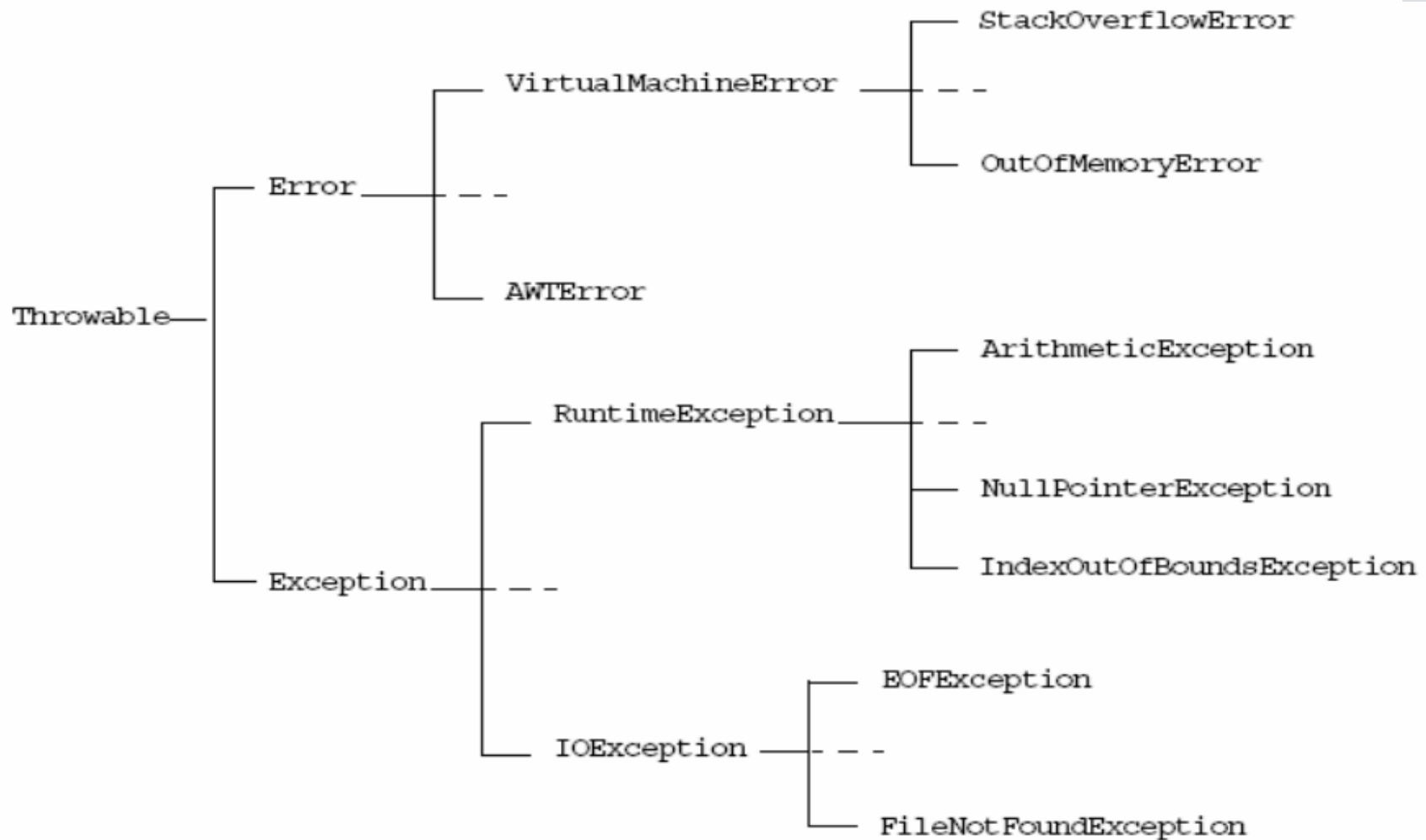


# Khái niệm ngoại lệ (Exception)





# Khái niệm ngoại lệ (Exception)

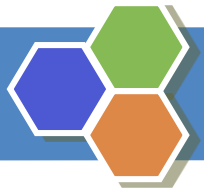




# Một số lớp ngoại lệ

Tên lớp ngoại lệ	Ý nghĩa
Throwable	Đây là lớp cha của mọi lớp ngoại lệ trong Java
Exception	Đây là lớp con trực tiếp của lớp Throwable, nó mô tả một ngoại lệ tổng quát có thể xảy ra trong ứng dụng
RuntimeException	Lớp cơ sở cho nhiều ngoại lệ java.lang
ArithmeticException	Lỗi về số học, ví dụ như 'chia cho 0'.
IllegalAccessException	Lớp không thể truy cập.
IllegalArgumentException	Đối số không hợp lệ.
ArrayIndexOutOfBoundsException	Lỗi truy cập ra ngoài mảng.
NullPointerException	Khi truy cập đối tượng null.
SecurityException	Cơ chế bảo mật không cho phép thực hiện.
ClassNotFoundException	Không thể nạp lớp yêu cầu.
NumberFormatException	Việc chuyển đổi từ chuỗi sang số không thành công.
AWTException	Ngoại lệ về AWT
IOException	Lớp cha của các lớp ngoại lệ I/O
FileNotFoundException	Không thể định vị tập tin
EOFException	Kết thúc một tập tin.
NoSuchMethodException	Phương thức yêu cầu không tồn tại.
InterruptedException	Khi một luồng bị ngắt.





# XỬ LÝ NGOẠI LỆ

- ❖ Giới thiệu
- ❖ Mô hình xử lý ngoại lệ
- ❖ Sử dụng các khối “try”, “catch” và “finally”
- ❖ Sử dụng các từ khoá “throw” và “throws”.
- ❖ Định nghĩa mới một ngoại lệ



# Mô hình xử lý ngoại lệ

- ❖ **Khi xảy ra ngoại lệ, một đối tượng thuộc lớp ngoại lệ cụ thể liên quan đến lỗi đó được tạo ra. Nhờ đó lập trình viên có thể bắt được đối tượng này để xử lý lỗi, giúp chương trình không bị ngắt đột ngột.**
  - Tách biệt luồng điều khiển bất thường với luồng bình thường.
  - Java cung cấp cơ chế hiệu quả trong việc bắt và xử lý lỗi ngoại lệ.
  - VD: Lớp `ArithmeticException` để quản lý lỗi ngoại lệ của phép chia cho 0, lớp `NullPointerException` để xử lý lỗi truy xuất đối tượng chưa được khởi tạo...

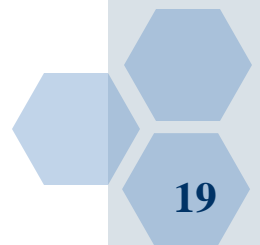


# Mô hình xử lý ngoại lệ

❖ Ngoại lệ cần phải được xử lý tại nơi phương thức sinh ra ngoại lệ hoặc uỷ nhiệm cho phương thức gọi đến.

❖ Các từ khoá:

- try
- catch
- finally
- throw
- throws





# Xử lý ngoại lệ

throw(s) và catch

Bắt đầu

some bugs: có lỗi runtime



kết thúc cách  
bất thường

phát sinh ngoại lệ



kết thúc cách  
bình thường

\* Là một dạng như cấu trúc điều khiển:  
Khi gặp phải một lỗi, luồng chương  
trình đang chạy sẽ bị dừng, ngoại lệ  
sẽ được xử lý sao cho nó được kết  
thúc một cách bình thường (theo kịch  
bản sẵn có của người lập trình)

Bắt đầu

Xử lý ngoại lệ



thả lỗi  
(throw)



bắt lỗi  
(catch)



kết thúc cách  
bình thường

198



# Cấu trúc mô hình xử lý ngoại lệ

```
try
{
    // place code that is expected to throw an exception
}
catch(Exception e1)
{
    // If an exception is thrown in 'try', which is of type e1, then perform
    // necessary actions here, else go to the next catch block
}
catch(Exception e2)
{
    // If an exception is thrown in, try which is of type e2, then perform
    // necessary actions here, else go to the next catch block
}
catch(Exception eN)
{
    // If an exception is thrown in, try which is of type eN, then perform
    // necessary actions here, else go to the next catch block
}
finally
{
    // this block is executed, whether or not the exception is throw.
}
```



# XỬ LÝ NGOẠI LỆ

- ❖ Giới thiệu
- ❖ Mô hình xử lý ngoại lệ
- ❖ Sử dụng các khối “try”, “catch” và “finally”
- ❖ Sử dụng các từ khoá “throw” và “throws”.
- ❖ Định nghĩa mới một ngoại lệ



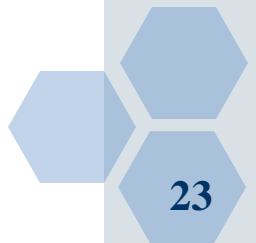
# Khối try/catch

❖ **Khối try...catch: phân tách đoạn chương trình thông thường và phần xử lý ngoại lệ.**

- `try { ... }`: Khối lệnh có khả năng gây ra ngoại lệ.
- `catch() { ... }`: Bắt và xử lý ngoại lệ.

❖ **Cú pháp:**

```
try {  
    // Doan ma co the gay ngoai le  
}  
  
catch (ExceptionType e) {  
    // Xu ly ngoai le  
}
```





# Ví dụ lỗi vượt quá chỉ số mảng

```
package xulyloi;  
public class DemoLoi {  
    public static void main(String[] args) {  
        int a[] = {1, 2, 4, 7, 10};  
        try {  
            System.out.println("a[6] = " + a[6]);  
  
        } catch (ArrayIndexOutOfBoundsException ex) {  
            System.out.println("Lỗi! Vượt quá chỉ mục của mảng! " + ex);  
        }  
    }  
}
```





# Ví dụ lỗi vượt quá chỉ số mảng

❖ Để tránh **ArrayIndexOutOfBoundsException** ta nên kiểm tra mảng thay vì sử dụng try-catch:

```
if (a.length > 6) {  
    int a2 = a[6];  
    System.out.println("Int at 6 = " + a2);  
} else {  
    System.out.println(" Không có số nào tại vị trí 6");  
}
```



# Ví dụ không xử lý ngoại lệ

```
class NoException {  
    public static void main(String args[]) {  
        String text = args[0];  
        System.out.println(text);  
    }  
}
```



```
D:\FIT-HUT\Lectures\OOP\OOP-Java\Demo>java NoException  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0  
    at NoException.main(NoException.java:3)  
D:\FIT-HUT\Lectures\OOP\OOP-Java\Demo>
```



# Ví dụ có xử lý ngoại lệ

```
class ArgExceptionDemo {  
    public static void main(String args[]) {  
        try {  
            String text = args[0];  
            System.out.println(text);  
        }  
        catch(Exception e) {  
            System.out.println("Hay nhap tham so khi chay!");  
        }  
    }  
}
```

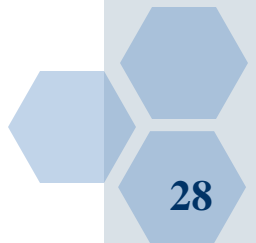


```
D:\FIT-HUT\Lectures\OOP\OOP-Java\Demo>java ArgExceptionDemo  
Hay nhap tham so khi chay?  
D:\FIT-HUT\Lectures\OOP\OOP-Java\Demo>_
```



# Ví dụ chia cho 0

```
public class HelloCatchException {  
    public static void main(String[] args) {  
        System.out.println("Three");  
        int value = 10 / 2;  
        System.out.println("Two");  
        value = 10 / 1;  
        System.out.println("One");  
        value=10/0; // Lỗi chia 0  
        // Dòng lệnh sau sẽ không được thực hiện  
        System.out.println("Value =" + value);  
        System.out.println("Let's go!");  
    }  
}
```





# Ví dụ chia cho 0

```
public class HelloCatchException {  
    public static void main(String[] args) {  
        System.out.println("Three");  
        int value = 10 / 2;  
        System.out.println("Two");  
        value = 10 / 1;  
        System.out.println("One");  
    }  
}
```



# Ví dụ chia cho 0


```
try {  
    value = 10 / 0; // Lỗi đã xảy ra tại đây.  
    // Dòng code này sẽ không được chạy.  
    System.out.println("Value =" + value);  
} catch (ArithmeticException e) {  
    // Các dòng lệnh trong catch được thực thi  
    System.out.println("Error: " + e.getMessage());  
    // Các dòng lệnh trong catch được thực thi  
    System.out.println("Ignore...");  
}  
// Dòng lệnh này được thực hiện.  
System.out.println("Let's go!");  
}
```

```
Problems @ Javadoc Declaration Console  
<terminated> HelloCatchException [Java Application] D:\DevPro  
Three  
Two  
One  
Error: / by zero  
Ignore...  
Let's go!
```



# Ví dụ IOException

```
import java.io.InputStreamReader;
import java.io.IOException;
public class HelloWorld{
    public static void main(String[] args) {
        InputStreamReader isr = new
            InputStreamReader(System.in);
        try {
            System.out.print("Nhap vao 1 ky tu: ");
            char c = (char) isr.read();
            System.out.println("Ky tu vua nhap: " + c);
        }catch(IOException ioe) {
            ioe.printStackTrace();
        }
    }
}
```



```
Nhap vao 1 ky tu: h
Ky tu vua nhap: h
Press any key to continue . . .
```



## Ví dụ số Fibonacci

```
package Demo;  
import java.util.Scanner;  
public class Main {  
    public static int nhap(){  
        Scanner input= new Scanner(System.in);  
        boolean check= false;  
        int n=0;  
        while(!check){  
            System.out.print(" ");
```





## Ví dụ

```
try{  
    n= input.nextInt();  
    check= true;  
}catch(Exception e){  
    System.out.println("Ban phai nhap so!  
hay nhap lai...");  
    input.nextLine();    }  
}  
return (n);  
}
```

```
public static void main(String[] args) {  
    System.out.print("Nhap n");  
    int n= nhap();  
    int[] f= new int[n+1];  
    f[0]= 1; f[1]= 1;  
    for(int i=2;i<=n;i++){  
        f[i]= f[i-1]+f[i-2];  
    }  
    System.out.println("So Fibonacci thu "+n+" la:  
f["+n+"]= "+f[n]);  
}
```



# Nhiều khối catch

❖ Một đoạn code có thể gây ra nhiều hơn một ngoại lệ -> Sử dụng nhiều khối catch.

❖ Cú pháp:

```
try {  
    // Đoạn mã có thể gây ra nhiều ngoại lệ  
} catch (ExceptionType1 e1) {  
    // Xử lý ngoại lệ 1  
} catch (ExceptionType2 e2) {  
    // Xử lý ngoại lệ 2  
} ...
```

❖ ExceptionType1 phải là lớp con hoặc ngang hàng với ExceptionType2



## Ví dụ

```
class MultipleCatch1 {  
    public static void main(String args[])  
    {  
        try {  
            String num = args[0];  
            int numValue = Integer.parseInt(num);  
            System.out.println("Dien tich hv la: "  
                               + numValue * numValue);  
        } catch (Exception e1) {  
            System.out.println("Hay nhap canh cua  
hv!");  
        } catch (NumberFormatException e2) {  
            System.out.println("Not a number!");  
        }  
    }  
}
```



D:\exception java.lang.NumberFormatException  
has already been caught



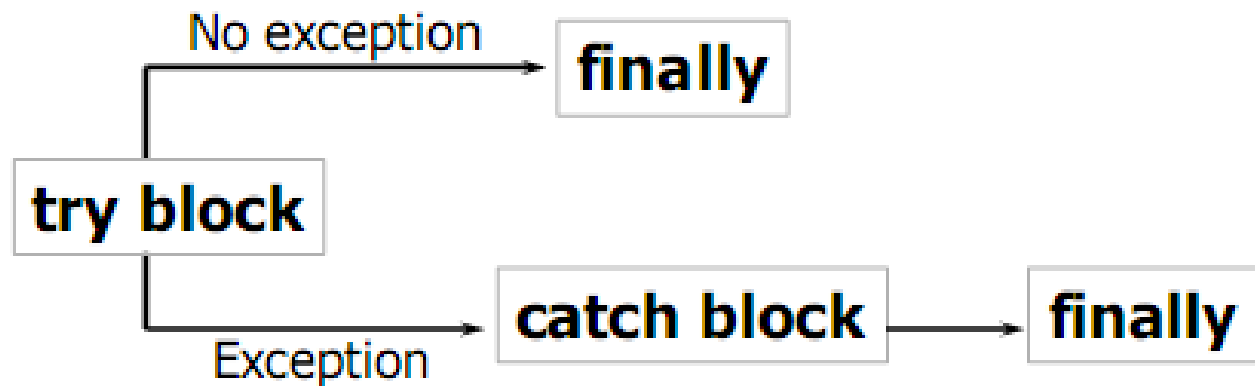
# Ví dụ

```
class MultipleCatch1 {
    public static void main(String args[])
    {
        try {
            String num = args[0];
            int numValue = Integer.parseInt(num);
            System.out.println("Dien tich hv la: "
                               + numValue * numValue);
        } catch (ArrayIndexOutOfBoundsException e1) {
            System.out.println("Hay nhap canh cua hv!");
        } catch (NumberFormatException e2) {
            System.out.println("Hay nhap 1 so!");
        }
    }
}
```



# Khối finally

- ❖ **Đảm bảo thực hiện tất cả các công việc cần thiết khi có ngoại lệ xảy ra**
  - Giải phóng tài nguyên
  - Đóng file, đóng socket, kết nối
- ❖ **Chắc chắn sẽ thực hiện dù ngoại lệ có xảy ra hay không**

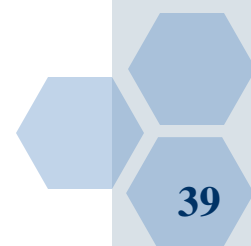




# try...catch...finally

```
try {  
    // Khoi lenh co the sinh ngoai le  
}  
catch(ExceptionType e) {  
    // Bat va xu ly ngoai le  
}  
finally {  
    /* Thuc hien cac cong viec can thiet du  
    ngoai le co xay ra hay khong */  
}
```

- ❖ Nếu đã có khối try thì bắt buộc phải có khối catch hoặc khối finally hoặc cả 2.



```
package xulyloi;
public class Demo{
    public void chia()
    {
        int a=10, b=0;
        try { System.out.println(a+" / “ b+" = "+(a/b));
        } catch (ArithmeticException e) {
            System.out.println("Gap phai loi: "+e);
        }finally {
            System.out.println("Nhưng việc cần thực hiện");
        }
    }
}
```





## Ví dụ (tiếp)

```
public static void main(String args[]){  
    new Demo().chia();  
    System.out.println("Quay lai tu ham  
main!");  
}
```



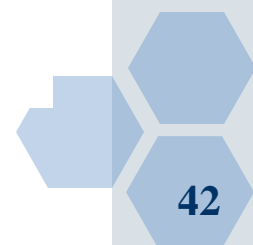
# Ví dụ

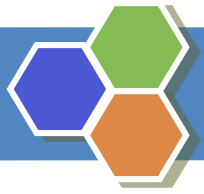
```
1 package exceptions;
2
3 public class ExceptionDemo {
4
5     public static void main(String[] args) {
6         int a[] = new int[2];
7         try{
8             System.out.println("Access element three: " + a[3]);
9         } catch (ArrayIndexOutOfBoundsException e) {
10             System.out.println("Exception thrown: " + e);
11         } finally {
12             a[0] = 6;
13             System.out.println("First element value: " + a[0]);
14             System.out.println("The finally statement is executed");
15         }
16     }
17 }
```

Exception thrown: [java.lang.ArrayIndexOutOfBoundsException](#): 3

First element value: 6

The finally statement is executed





# XỬ LÝ NGOẠI LỆ

- ❖ Giới thiệu
- ❖ Mô hình xử lý ngoại lệ
- ❖ Sử dụng các khối “try”, “catch” và “finally”
- ❖ Sử dụng các từ khoá “throw” và “throws”.
- ❖ Định nghĩa mới một ngoại lệ



# Throw và throws

- ❖ **Từ khoá throw:** để ném ra một ngoại lệ (do JVM hoặc LTV)
  - Cú pháp: `throw throwableObject;`
  - VD: `throw new ArrayIndexOutOfBoundsException("Lỗi chỉ số mảng");`
- ❖ **Từ khoá throws:** để ném ra một hoặc nhiều ngoại lệ của phương thức.
  - Cú pháp: `[modifiers] returnType methodName(params) throws class_Exception1,...,class_Exception_n{  
 // thân phương thức  
}`
  - VD: `void tinhMang(int []a) throws NullPointerException,  
 ArithmeticException{  
 // có thể mảng chưa được khởi tạo hoặc lỗi chia cho 0  
}`



# Throw và throws

- ❖ Throw: để ném ra Exception ở bất kỳ dòng nào trong phương thức. Sau đó dùng try...catch để bắt hoặc throws cho chỗ khác xử lý.
- ❖ Throws: chỉ có phương thức mới được sử dụng. Khi 1 phương thức có throw bên trong mà không bắt lại bởi try...catch thì phải ném đi (throws) cho chỗ khác xử lý.
- ❖ Chú ý: Nếu phương thức đang ném ra lỗi(throws) mà phương thức khác gọi phương thức đang ném ra lỗi thì phương thức đó vẫn phải ném ra lỗi hoặc xử lý luôn (try-catch).
- ❖ Dùng throws khi ta không muốn đặt nhiều khối try-catch bên trong.



# Ví dụ

```
1 package exceptions;
2
3 import java.util.Scanner;
4
5 public class ExceptionDemo {
6
7     public static void ps (int a, int b) throws ArithmeticException {
8         if (b == 0)
9             throw new ArithmeticException ();
10        else
11            System.out.print ("Kết quả: " + a/b);
12        }
13
14    public static void main(String[] args) {
15        int tuso, mauso;
16        System.out.println ("Chương trình tính phân số: ");
17        Scanner ca = new Scanner (System.in);
18        System.out.print ("Tử số: ");
19        tuso = ca.nextInt ();
20        System.out.print ("Mẫu số: ");
21        mauso = ca.nextInt ();
22        try {
23            ps (tuso, mauso);
24        } catch (ArithmeticException ex) {
25            System.out.println ("Error: " + ex.toString());
26        }
27        System.out.println("Chạy đến đây");
28        ca.close();
29    }
30 }
```



❖ **Dùng throw hoặc throws bắt các ngoại lệ khi nhập một số từ bàn phím trong các trường hợp:**

- Không nhập
- Nhập sai định dạng

```
import java.util.*;
import java.lang.*;
class Main{
    static int nhapso() throws Exception{
        int so;
        Scanner input = new Scanner(System.in);
        System.out.print("Nhap so: ");
        String str = input.nextLine();
        if (str.trim().isEmpty()){
            throw new Exception("Ban chua nhap chu so hoac toan
khoang trang"); }
        so = Integer.parseInt(str);
        return so; }
```



```
public static void main(String[] args) {  
    try{  
        int n;  
        n = nhapso();  
        System.out.println(n+"\t");  
    } catch(NumberFormatException ex){  
        System.out.println("Loi: Day nhap vao khong phai la so");  
    } catch(Exception ex){ System.out.println("Loi: " +  
ex.getMessage());  
    }  
    }  
}
```



# Bài tập

- ❖ **Viết chương trình nhập vào họ tên, và năm sinh với điều kiện họ tên không quá 25 ký tự, và năm sinh từ 1990 đến 1995**



# Ví dụ

```
import java.io.*;
class Nhap {
//Cho qua kiểu ngoại lệ IOException là lỗi nhập xuất phát sinh khi
thực hiện hàm readLine() của lớp BufferedReader
public static void main (String[] args) throws IOException{
BufferedReader kbd = new BufferedReader(new InputStreamReader
(System.in));
String s = null;
// Nhập ho ten
do { System.out.print("Nhap Ho va ten khong qua 25 ky tu: ");
s = kbd.readLine();
} while (s.length()>25 || s.length()==0);
System.out.println("Ho va ten là : "+s);
}
```



# Ví dụ (tiếp)

```
// Nhập nam sinh
while (true) {
    try {
        System.out.print("Nhập nam sinh : ");
        s = kbd.readLine();
        //Hàm parseInt() của lớp Integer phát sinh lỗi định dạng số
        int ns = Integer.parseInt(s);
        //Điều kiện phát sinh lỗi định dạng số NumberFormatException
        if (ns < 1990 || ns > 1995) throw new NumberFormatException();
        System.out.println("Nam sinh là : "+ns);
        break;
    }
    //Bắt kiểu ngoại lệ NumberFormatException
    catch (NumberFormatException e){
        System.out.println("Bạn nhập lại nam sinh tu 1990 -1995");
    }
}
}
```



# XỬ LÝ NGOẠI LỆ

- ❖ Giới thiệu
- ❖ Mô hình xử lý ngoại lệ
- ❖ Sử dụng các khối “try”, “catch” và “finally”
- ❖ Sử dụng các từ khoá “throw” và “throws”.
- ❖ Định nghĩa mới một ngoại lệ



# Định nghĩa mới 1 ngoại lệ

- ❖ Ta có thể tạo lớp ngoại lệ để phục vụ các mục đích riêng.
- ❖ Lớp ngoại lệ mới phải được kế thừa từ lớp **Exception** hoặc lớp dẫn xuất của lớp này.
- ❖ Có thể cung cấp hai constructor:
  - Constructor mặc định (không tham số).
  - Constructor nhận một tham số String và truyền tham số này cho phương thức khởi tạo của lớp cơ sở.



# Định nghĩa mới 1 ngoại lệ

## ❖ Chú ý:

- Tất cả các exceptions phải là con của lớp Throwable.
- Nếu ta muốn viết checked exception, thì phải extends từ lớp Exception.
- Nếu ta muốn viết các RuntimeException thì phải extends từ Runtime lớp exception.



# Định nghĩa mới 1 ngoại lệ

## ❖ Ví dụ:

```
class SimpleException extends Exception {  
}  
  
class MyException extends Exception {  
    public MyException() {}  
    public MyException(String msg) {  
        super(msg);  
    }  
}
```





# Ví dụ

```
1 package exceptions;
2
3 public class NgoaiLeTuDinhNghia {
4
5     public static void main(String[] args) {
6         try{
7             throw new MyException(2);
8             // throw is used to create a new exception and throw it.
9         } catch(MyException e){
10             System.out.println(e) ;
11         }
12     }
13 }
14
15 class MyException extends Exception{
16
17     int a;
18
19     public MyException(int b) {
20         a=b;
21     }
22
23     public String toString(){
24         return ("Exception Number = " + a) ;
25     }
26 }
```

```
package exception;  
import java.util.Scanner;  
public class NgoaiLeTuDinhNghia{  
public static void nhap() throws MyException{  
    Scanner sc=new Scanner(System.in);  
    System.out.println(“Nhập số người:”);  
    int a=sc.nextInt();  
    if(a<=0) throw new MyException();  
}
```



## Ví dụ

```
public static void main(String [] args){  
    try{  
        nhap();  
    }catch (MyException e){  
        System.out.println(“Lỗi”+e.toString());  
    }  
}
```

```
Class MyException extends Exception{  
    public MyException(){  
        super("Số người phải lớn hơn 0");  
    }  
    public MyException(String message){  
        super(message);  
    }  
}
```

```
Nhập số người: 0  
Lỗi: exceptions.MyException:
```



## Ví dụ Demo

**Bài tập:** Rút tiền ngân hàng, sử dụng định nghĩa ngoại lệ khi người dùng rút tiền vượt quá số tiền có trong TK ngân hàng.

- Tạo ra các class `TaiKhoanNganHang` để thực hiện các tác vụ gửi tiền, rút tiền và một class `LoiChuyenTien` để xử lý nếu có ngoại lệ xảy ra khi ta rút số tiền vượt quá trong TK.



# Ví dụ

```
28 class TaiKhoanNganHang{
29
30     double tienHeThong;
31
32     public void guitien(double tienGui){
33         tienHeThong += tienGui;
34     }
35
36     public void ruttien(double tienRut) throws LoiChuyenTien {
37         if(tienHeThong >= tienRut)
38             tienHeThong = tienHeThong - tienRut;
39         else
40             throw new LoiChuyenTien(tienRut);
41     }
42 }
43
44
45
46 class LoiChuyenTien extends Exception{
47
48     double tienRut;
49
50     public LoiChuyenTien(double tienRut){
51         this.tienRut = tienRut;
52     }
53
54     public String toString(){
55         return ("Số tiền " + tienRut + " cần rút lớn hơn số tiền hiện có!");
56     }
57 }
```



# Ví dụ

```
1 package exceptions;
2
3 import java.util.Scanner;
4
5 public class RutTienNganHang {
6
7     private static double tienGui, tienRut;
8
9     public static void main(String[] args) {
10         Scanner ca = new Scanner(System.in);
11         System.out.print("Nhập số tiền cần gửi: ");
12         tienGui = ca.nextDouble();
13         TaiKhoanNganHang tk = new TaiKhoanNganHang();
14         tk.guitien(tienGui);
15         System.out.println("Tài khoản trước khi rút = " + tk.tienHeThong);
16         System.out.print("Nhập số tiền cần rút: ");
17         tienRut = ca.nextDouble();
18         try {
19             tk.ruttien(tienRut);
20             System.out.println("Tài khoản sau khi rút là: " + tk.tienHeThong);
21         } catch (LoiChuyenTien e) {
22             System.out.println(e.toString());
23         }
24         ca.close();
25     }
26 }
```



# Ví dụ Demo

Kết quả: Khi nhập số tiền nhỏ hơn số tiền trong tài khoản

```
Nhập số tiền cần gửi: 1500000  
Tài khoản trước khi rút = 1500000.0  
Nhập số tiền cần rút: 500000  
Tài khoản sau khi rút là: 1000000.0
```

Kết quả: Khi nhập số tiền lớn hơn số tiền trong tài khoản

```
Nhập số tiền cần gửi: 1500000  
Tài khoản trước khi rút = 1500000.0  
Nhập số tiền cần rút: 2000000  
Số tiền 2000000.0 cần rút lớn hơn số tiền hiện có!
```





# Ví dụ Demo

❖ **// File Name CheckingAccount.java**

```
import java.io.*;
```

```
public class CheckingAccount {
```

```
    private double balance;
```

```
    private int number;
```

```
    public CheckingAccount(int number) {
```

```
        this.number = number;
```

```
    }
```

```
    public void deposit(double amount) {
```

```
        balance += amount;
```

```
    }
```





## Ví dụ Demo

```
public void withdraw(double amount) throws  
InsufficientFundsException {  
    if(amount <= balance) {  
        balance -= amount;  
    } else {  
        double needs = amount - balance;  
        throw new InsufficientFundsException(needs);  
    }  
    public double getBalance() { return balance; }  
    public int getNumber() { return number; } }
```





# Định nghĩa mới 1 ngoại lệ

**// File Name BankDemo.java**

```
public class BankDemo { public static void main(String [] args) {  
    CheckingAccount c = new CheckingAccount(101);  
    System.out.println("Depositing $500...");  
    c.deposit(500.00);  
    try { System.out.println("\nWithdrawing $100...");  
        c.withdraw(100.00);    System.out.println("\nWithdrawing  
$600...");    c.withdraw(600.00);  
    }catch(InsufficientFundsException e) {  
        System.out.println("Sorry, but you are short $" +  
        e.getAmount());  
        e.printStackTrace(); }  
    }  
}
```





## Ví dụ

### ❖ Kết quả chạy chương trình:

```
Depositing $500...

Withdrawing $100...

Withdrawing $600...
Sorry, but you are short $200.0
InsufficientFundsException
    at CheckingAccount.withdraw(CheckingAccount.java:25)
    at BankDemo.main(BankDemo.java:13)
```