

# Chương 2

## JAVA CĂN BẢN

# Mục tiêu



- Viết được chương trình Java theo đúng cú pháp

# Nội dung



- 2.1. Giới thiệu Java
- 2.2. Cấu trúc chương trình Java
- 2.3. Tổng quan lập trình Java
  - 2.3.1. Kiểu dữ liệu
  - 2.3.2. Biến, hằng
  - 2.3.3. Toán tử, biểu thức
  - 2.3.4. Các cấu trúc lệnh (cấu trúc điều khiển, lặp)
  - 2.3.5. Viết phương thức trong Java
  - 2.3.6. Mảng
- 2.4. Sử dụng một số lớp có sẵn
- 2.5. Ngoại lệ (Exception)

# 2.1. Giới thiệu Java

## Java là gì?



- Java: là ngôn ngữ lập trình hướng đối tượng do Sun Microsystem đưa ra vào giữa thập niên 90
- Java: vừa biên dịch (compiler) vừa thông dịch (interpreter)
- Java: độc lập nền (phần cứng và hệ điều hành)

# 2.1. Giới thiệu Java

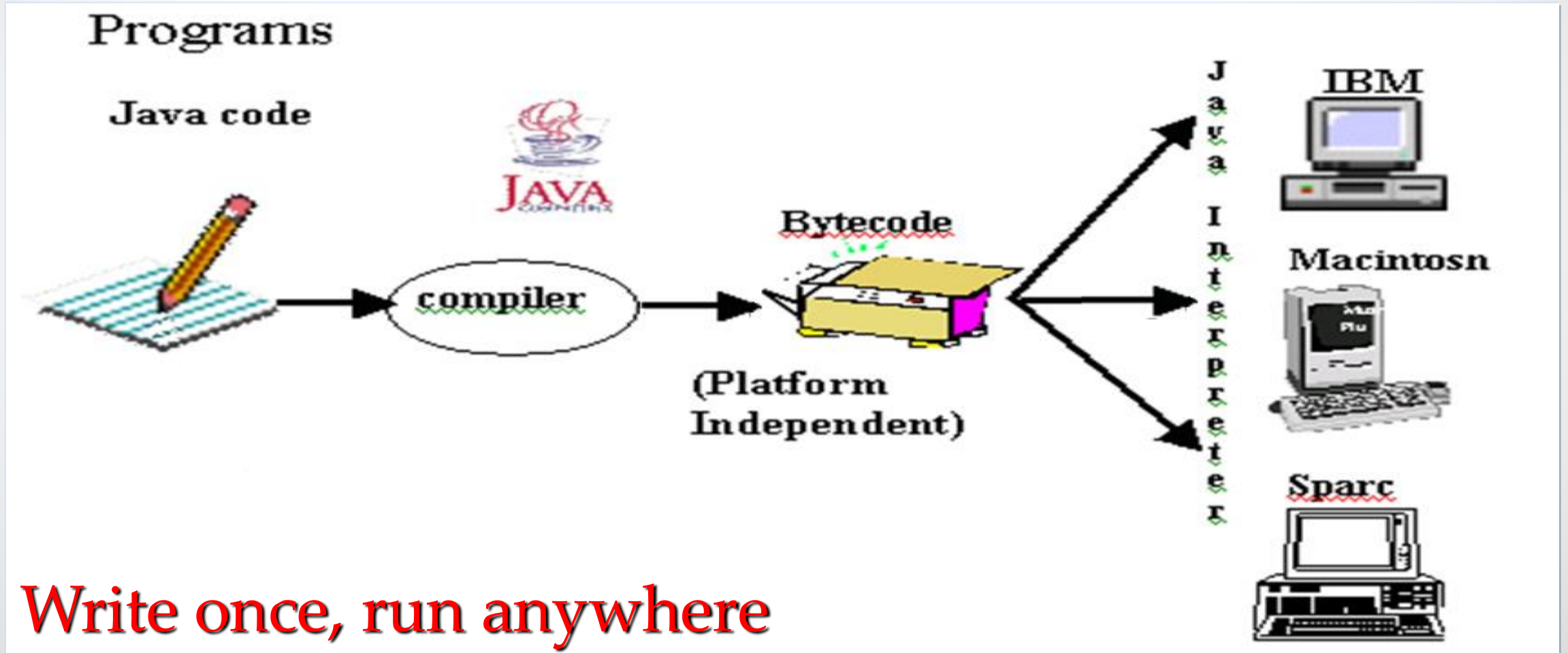
## Các phiên bản chính của Java



- Standard edition (Java SE hoặc J2SE)
  - Còn gọi là ngôn ngữ lập trình Java
  - Dùng để viết: ứng dụng desktop, applets, java FX, ứng dụng web không cần Java EE
- Enterprise edition (Java EE hoặc J2EE)
  - Là java chạy trên các ứng dụng server
  - Dùng để viết: Servlet, JSP, JSF, Strut, EJB, Spring, Hibernate,...
  - Ví dụ: Google home page, gmail, Google Maps, Google Docs
- Micro edition (Java ME)
  - Là java chạy trên thiết bị di động và nhúng
  - Ứng dụng viết cho: ĐTDT, PDA, TV set-top box, máy in

# 2.1. Giới thiệu Java

## Quá trình dịch chương trình Java



# 2.1. Giới thiệu Java

## JDK – Java Development Kit



- Là môi trường phát triển để xây dựng ứng dụng, applet bằng ngôn ngữ lập trình Java.
- Bao gồm:
  - javac: Chương trình dịch chuyển mã nguồn sang bytecode
  - java: Bộ thông dịch, dùng để thực thi java application
  - appletviewer: Bộ thông dịch, thực thi java applet mà không cần sử dụng trình duyệt như Netscape, hay IE, v.v.
  - javadoc: Bộ tạo tài liệu dạng HTML từ mã nguồn và chú thích
  - javap: Trình dịch ngược bytecode
  - ...

## 2.2. Cấu trúc chương trình Java



### Comments

#### 1. Single Line Comment

```
// insert comments here
```

#### 2. Block Comment

```
/*  
 * insert comments here  
*/
```

#### 3. Documentation Comment

```
/**  
 * insert documentation  
*/
```

```
/*  
 * Created on Jul 14, 2005  
 *  
 * First Java Program  
 */  
package com.jds.sample;  
import java.util.*;  
  
/**  
 * @author JDS  
 */  
public class JavaMain {  
    public static void main(String[] args) {  
        // print a message  
        System.out.println("Welcome to Java!");  
    }  
}  
  
class Extra {  
    /*  
     * class body  
     */  
}
```



## 2.2. Cấu trúc chương trình Java



### Declaration order

#### 1. Package declaration

Used to organize a collection of related classes.

#### 2. Import statement

Used to reference classes.

#### 3. Class declaration

A Java source file can have several classes but only one public class is allowed.

A file can have more classes, but the file name must match with one in it

```
/*
 * Created on Jul 14, 2005
 *
 * First Java Program
 */
package com.jds.sample;
import java.util.*;

/**
 * @author JDS
 */
public class JavaMain {
    public static void main(String[] args) {
        // print a message
        System.out.println("Welcome to Java!");
    }
}

class Extra {
    /*
     * class body
     */
}
```

## 2.3. Tổng quan lập trình Java



- Kiểu dữ liệu
- Biến, hằng
- Toán tử, biểu thức
- Các cấu trúc lệnh (cấu trúc điều khiển, lặp)
- Viết phương thức trong Java
- Mảng

## 2.3. Tổng quan lập trình Java

### Kiểu dữ liệu



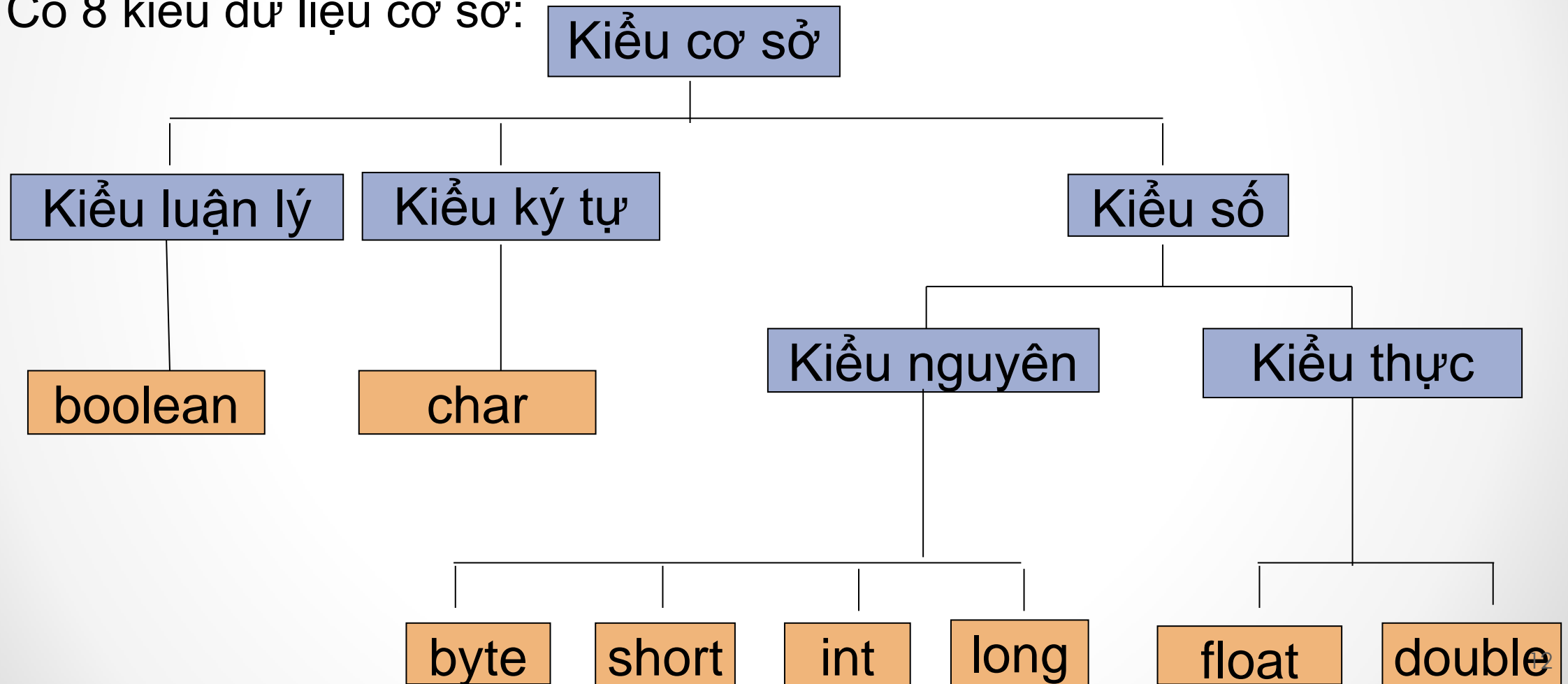
- Có hai kiểu dữ liệu:
  - Kiểu dữ liệu cơ sở (Primitive data type)
  - Kiểu dữ liệu tham chiếu - dẫn xuất (Reference data type)

## 2.3. Tổng quan lập trình Java

### Kiểu dữ liệu cơ sở



- Có 8 kiểu dữ liệu cơ sở:



## 2.3. Tổng quan lập trình Java

### Kiểu dữ liệu cơ sở



- Bảng giá trị của các kiểu cơ sở

Type	Bits	Lowest Value	Highest Value
<b>boolean</b>	(n/a)	false	true
<b>char</b>	16	'\u0000' [0]	'\uffff' [ $2^{16}-1$ ]
<b>byte</b>	8	-128 [ $-2^7$ ]	+127 [ $2^7-1$ ]
<b>short</b>	16	-32,768 [ $-2^{15}$ ]	+32,767 [ $2^{15}-1$ ]
<b>int</b>	32	-2,147,483,648 [ $-2^{31}$ ]	+2,147,483,647 [ $2^{31}-1$ ]
<b>long</b>	64	-9,223,372,036,854,775,808 [ $-2^{63}$ ]	+9,223,372,036,854,775,807 [ $2^{63}-1$ ]
<b>float</b>	32	$\pm 1.40129846432481707e-45$	$\pm 3.40282346638528860e+38$
<b>double</b>	64	$\pm 4.94065645841246544e-324$	$\pm 1.79769313486231570e+308$

## 2.3. Tổng quan lập trình Java

### Kiểu dữ liệu tham chiếu



- Kiểu dữ liệu tham chiếu biểu diễn cho đối tượng (object)
- Java có 3 kiểu dữ liệu tham chiếu:
  - **Class**
  - **Array**
  - **Interface**

## 2.3. Tổng quan lập trình Java

### Biến, hằng: Cách đặt tên



- Bắt đầu bằng ký tự, ký tự gạch dưới ('\_') hay ký tự '\$'
- Sau đó là các ký tự, ký số hay '\_', '\$'. Không dùng các ký tự khác như: khoảng trống, ký hiệu phép toán
- Tên có tính chất phân biệt chữ thường chữ hoa (case-sensitive)

## 2.3. Tổng quan lập trình Java

### Biến: Khai báo



- Khai báo và sử dụng biến
  - Biến là một giá trị có thể thay đổi khi chương trình thực thi
  - Cách khai báo:
  - Ví dụ:

```
<kiểu_dữ_liệu> <tên_biến>;  
<kiểu_dữ_liệu> <tên_biến> = <giá_trị>;
```

Kiểu dữ liệu

Tên biến

```
int total;  
int count, temp, result;  
int sum = 0;  
int base = 32, max = 149;
```



## 2.3. Tổng quan lập trình Java

### Biến: Khai báo



- Khai báo biến có kiểu dữ liệu cơ sở:

```
int    age    =    0;
```

kiểu cơ sở      tên biến      giá trị khởi tạo

- Khai báo biến có kiểu dữ liệu tham chiếu (khai báo object):

```
String name = new String("Jason");
```

```
Date    chrm = new Date(2018, 12, 24);
```

kiểu tham chiếu      tên đối tượng      giá trị khởi tạo

## 2.3. Tổng quan lập trình Java

### Biến: cục bộ, toàn cục



- Biến cục bộ: phải được khởi tạo giá trị trước khi sử dụng
- Biến toàn cục: phải được khai báo bên ngoài các hàm và bên trong class, ví dụ:

```
class Example {  
    static int x = 10;  
    public static void main(String args[ ]) {  
        System.out.println(x);  
    }  
}
```

## 2.3. Tổng quan lập trình Java

### Hằng



- Khai báo và sử dụng hằng (constant)
  - Giá trị hằng không được phép thay đổi trong chương trình, nó chỉ được gán giá trị một lần ở dòng khai báo hằng
  - Trong Java, dùng **final** để khai báo hằng, ví dụ:  
**final** int MIN\_HEIGHT = 69;

## 2.3. Tổng quan lập trình Java

### Toán tử, biểu thức



- Các toán tử số học:

Cộng	+
Trừ	-
Nhân	*
Chia	/
Chia lấy số dư	%

- Toán tử tăng/giảm
  - Toán tử tăng (++)
  - Toán tử giảm (--)
  - Câu lệnh `count++`; tương đương với `count = count + 1;`

## 2.3. Tổng quan lập trình Java

### Toán tử, biểu thức



- Toán tử so sánh (quan hệ)

==      bằng

!=      không bằng

<      nhỏ hơn

>      lớn hơn

<=      nhỏ hơn hoặc bằng

>=      lớn hơn hoặc bằng

- Toán tử luận lý

!    :    not

&& :    and

||   :   or

## 2.3. Tổng quan lập trình Java

### Toán tử, biểu thức



- Toán tử điều kiện
  - Cú pháp: **điều\_kiện ? biểu\_thức1 : biểu\_thức2**
  - Nếu điều\_kiện là *true*, *biểu\_thức1* được thực thi; Nếu là *false*, *biểu\_thức2* được thực thi
  - Ví dụ:  

```
int x = 10, y = 20;  
int Z = (x < y) ? 30 : 40;
```
- Các biểu thức
  - Một biểu thức là một sự kết hợp giữa các toán tử và các toán hạng
  - Nếu trong biểu thức có chứa số thực thì kết quả trả về số thực
  - Lưu ý độ ưu tiên của các toán tử trong 1 biểu thức

## 2.3. Tổng quan lập trình Java

### Các cấu trúc lệnh trên Java



- Cấu trúc điều khiển – Rẽ nhánh

Cấu trúc if

```
if (Condition)
{
    Statements;
}
else
{
    Statement;
}
```

Cấu trúc switch

```
switch (Expression)
{ case Cons1: Statements;
  break;
  case Cons2: Statements;
  break;
  . . .
  default : Statements;
}
```

## 2.3. Tổng quan lập trình Java

### Các cấu trúc lệnh trên Java



- Cấu trúc điều khiển – Rẽ nhánh
  - Thường một lệnh **break** được dùng ở cuối danh sách lệnh của mỗi case. Một cấu trúc switch có thể có một **case default**
  - Kết quả của **Expression** trong switch phải là kiểu số nguyên (byte, short, int, long) hoặc char. Không thể là boolean hoặc số thực (float, double)



## 2.3. Tổng quan lập trình Java

### Các cấu trúc lệnh trên Java



- Cấu trúc lặp

```
while condition)
{
    Statements;
}
```

```
do
{
    Statements;
}while (condition) ;
```

```
for (VarInit; Condition; Statements)
{
    Statements1;
}
```

## 2.3. Tổng quan lập trình Java

### Viết phương thức trong Java



- Cú pháp:

```
ReturnType methodName (Type agr1, Type arg2,...)
{
    ...
    return (somethingOfReturnType);
}
```

- Ví dụ 1: Viết chương trình tính tổng các số nguyên chẵn từ 1-> N. N được nhập từ bàn phím. Yêu cầu viết phương thức tính tổng.
- Ví dụ 2: Viết chương trình tính tổng các số nguyên tố từ 1-> N. N được nhập từ bàn phím. Yêu cầu viết phương thức kiểm tra nguyên tố, phương thức tính tổng.

## 2.3. Tổng quan lập trình Java

# Mảng



- Là một cấu trúc dữ liệu cho phép lưu một tập các giá trị cùng kiểu

The entire array  
has a single name

scores

Each value has an *index*

0	1	2	3	4	5	6	7	8	9
79	87	94	82	67	98	87	81	74	91

## 2.3. Tổng quan lập trình Java

### Khai báo mảng



- Cú pháp: 

```
DataType[] arrayName = new DataType[size];
```

```
DataType[] arrayName; // only declares the variable  
arrayName = new DataType[size];
```
- Ví dụ khai báo mảng:
  - `int[] scores = new int[100];` // mảng scores chứa 100 phần tử số nguyên
  - `String[] names;`  
`names = new names[20];` // mảng names chứa 20 chuỗi
- Ví dụ khai báo và gán giá trị cùng lúc:
  - `int[] smallPrimes = { 2, 3, 5, 7, 11, 13 };`
  - `char[] letterGrades = {'A', 'B', 'C', 'D', 'F'};`

## 2.3. Tổng quan lập trình Java

### Sử dụng mảng



- Duyệt mảng a:

```
for (int i=0; i < a.length; i++)  
    // process a[i]
```

length field  
returns the number  
of elements

```
for (Type x : a)  
    // process x
```

- Ví dụ:
  - for (int i=0; i < scores.length; i++)  
 System.out.println(scores[i]);
  - for (int score : scores)  
 System.out.println(score);
  - for (String n : names)  
 System.out.println(n);
- Bài tập: tạo mảng số nguyên có 10 phần tử, gán giá trị bất kỳ và xuất mảng.

## 2.4. Sử dụng một số lớp có sẵn

### Lớp Scanner



- Nằm trong gói java.util
- Công dụng: Đọc dữ liệu từ bàn phím
- Cách dùng:
  - Tạo đối tượng để đọc từ bàn phím:
    - `Scanner scan = new Scanner (System.in);`
  - Dùng các phương thức của Scanner để nhập dữ liệu từ bàn phím, như:
    - `string answer = scan.nextLine();` → Nhập một chuỗi ký tự
    - `int a = scan.nextInt();` → Nhập một số nguyên
    - `double d = scan.nextDouble();` → Nhập một số thực
    - ...

## 2.4. Sử dụng một số lớp có sẵn

### Lớp Scanner: ví dụ



```
import java.util.*;
public class Cong2so {
    public static void main (String[] args)    {
        Scanner nhap = new Scanner(System.in);
        int x,y;

        System.out.print("Nhap so thu nhat: ");
        x = nhap.nextInt();

        System.out.print("Nhap so thu hai: ");
        y = nhap.nextInt();

        int tong = x+y;
        System.out.println ("The sum is: "+ tong);
    }
}
```

## 2.4. Sử dụng một số lớp có sẵn

### Lớp Random



- Lớp Random nằm trong gói `java.util`
- Công dụng: Dùng để phát sinh số ngẫu nhiên
- Cách dùng:
  - Tạo đối tượng:
    - `Random rd = new Random();`
  - Phát sinh số ngẫu nhiên có miền giá trị thuộc  $[0, n-1]$ 
    - `int a = rd.nextInt(n);`
  - Phát sinh số ngẫu nhiên có miền giá trị thuộc  $[0.0, 1.0)$ 
    - `float a = rd.nextFloat();`
- Vd/ Phát sinh 3 số a,b,c thuộc  $[0,50]$ , tìm số lớn nhất.



## 2.4. Sử dụng một số lớp có sẵn

### Lớp Math



- Lớp Math nằm trong gói **java.lang** (gói mặc định)
- Chứa các phương thức có chức năng tính toán về toán học:
  - pow: Lũy thừa
  - sqrt: Căn bậc 2
  - abs: Trị tuyệt đối
  - ...
- Các phương thức trong lớp Math là những phương thức tĩnh (static methods). Vì vậy, chỉ cần gọi trực tiếp phương thức thông qua tên lớp mà không cần tạo đối tượng
  - Ví dụ: `double value = Math.sqrt(25.0);`

## 2.4. Sử dụng một số lớp có sẵn

# Định dạng kết quả xuất ra màn hình

---



- Lớp `NumberFormat` cho phép định dạng giá trị theo kiểu tiền tệ hoặc phần trăm
- Lớp `DecimalFormat` cho phép định dạng giá trị theo mẫu định dạng cho trước
- Cả 2 nằm trong gói `java.text`

## 2.4. Sử dụng một số lớp có sẵn

# Định dạng kết quả xuất ra màn hình



```
import java.text.NumberFormat;
import java.util.*;
public class Price {
    public static void main (String[] args) {
        final double TAX_RATE = 0.06;           // 6% sales tax
        int quantity;
        double subtotal, tax, totalCost, unitPrice;
        Scanner scan = new Scanner(System.in);
        System.out.print ("So luong: ");
        quantity = scan.nextInt();
        System.out.print ("Don gia: ");
        unitPrice = scan.nextDouble();

        subtotal = quantity * unitPrice;
        tax = subtotal * TAX_RATE;
        totalCost = subtotal + tax;
        // Print output with appropriate formatting
        NumberFormat money = NumberFormat.getCurrencyInstance();
        NumberFormat percent = NumberFormat.getPercentInstance();
        System.out.println ("Thanh tien: " + money.format(subtotal));
        System.out.println ("Thue: " + money.format(tax) + " at " + percent.format(TAX_RATE));
        System.out.println ("Tong tien: " + money.format(totalCost));
    }
}
```

**So luong: 5**  
**Don gia: 3.87**  
**Thanh tien: \$19.35**  
**Thue: \$1.16 at 6%**  
**Tong tien: \$20.51**

## 2.4. Sử dụng một số lớp có sẵn

# Định dạng kết quả xuất ra màn hình



```
import java.text.DecimalFormat;
import java.util.*;
public class CircleStats {
    public static void main (String[] args) {
        int radius;
        double area, circumference;
        Scanner scan = new Scanner(System.in);
        System.out.print ("Enter the circle's radius: ");
        radius = scan.nextInt();

        area = Math.PI * Math.pow(radius, 2);
        circumference = 2 * Math.PI * radius;

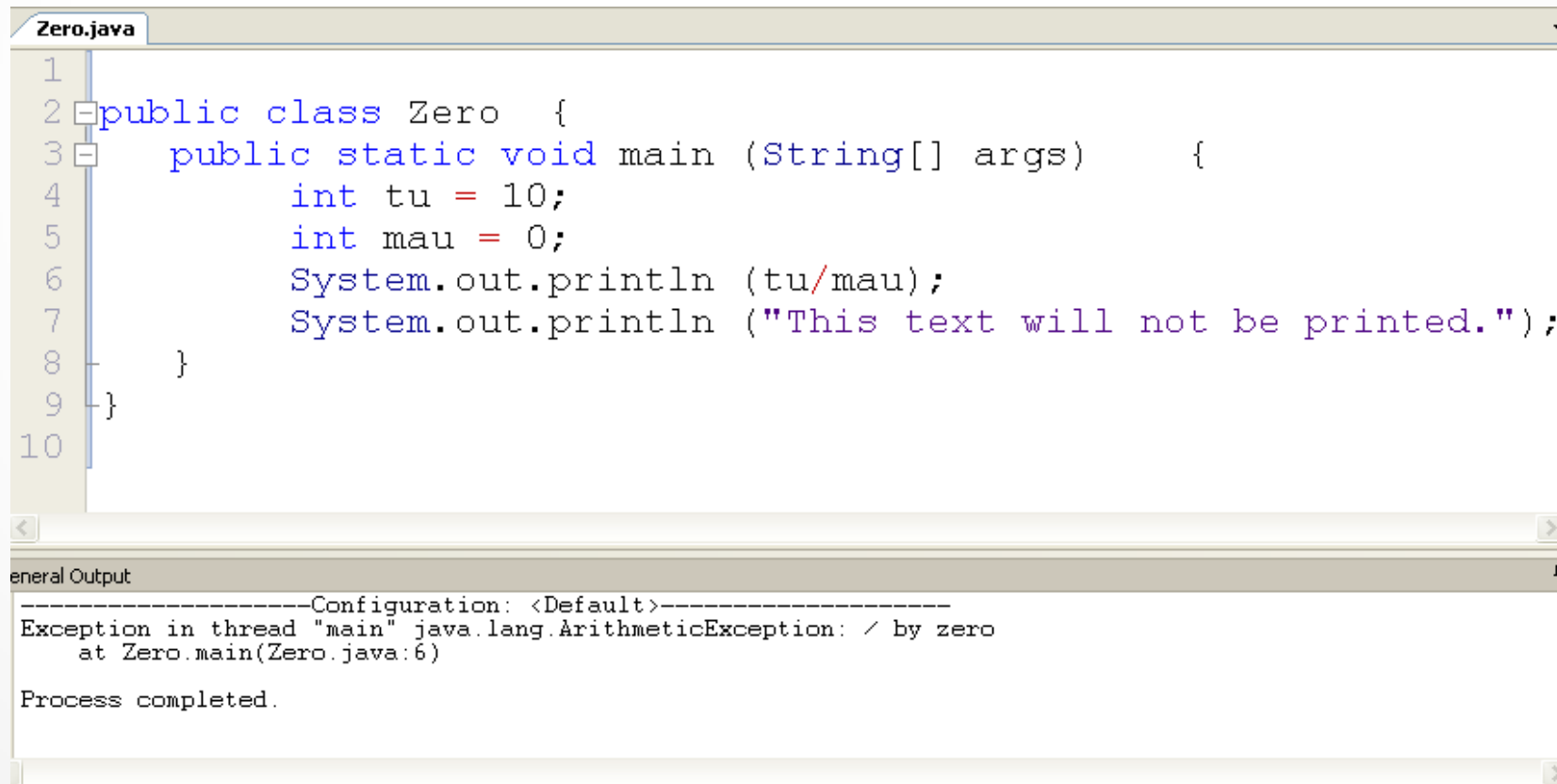
        // Round the output to three decimal places
        DecimalFormat fmt = new DecimalFormat ("0.###");
        System.out.println ("The circle's area: " + fmt.format(area));
        System.out.println ("The circle's circumference: " + fmt.format(circumference));
    }
}
```

Enter the circle's radius: 5  
The circle's area: 78.54  
The circle's circumference: 31.416

## 2.5. Ngoại lệ



- Exception là một sự kiện xảy ra trong quá trình thực thi chương trình, phá vỡ luồng bình thường của chương trình
- Ví dụ:



```
Zero.java
1
2 public class Zero {
3     public static void main (String[] args) {
4         int tu = 10;
5         int mau = 0;
6         System.out.println (tu/mau);
7         System.out.println ("This text will not be printed.");
8     }
9 }
10

General Output
-----Configuration: <Default>-----
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at Zero.main(Zero.java:6)

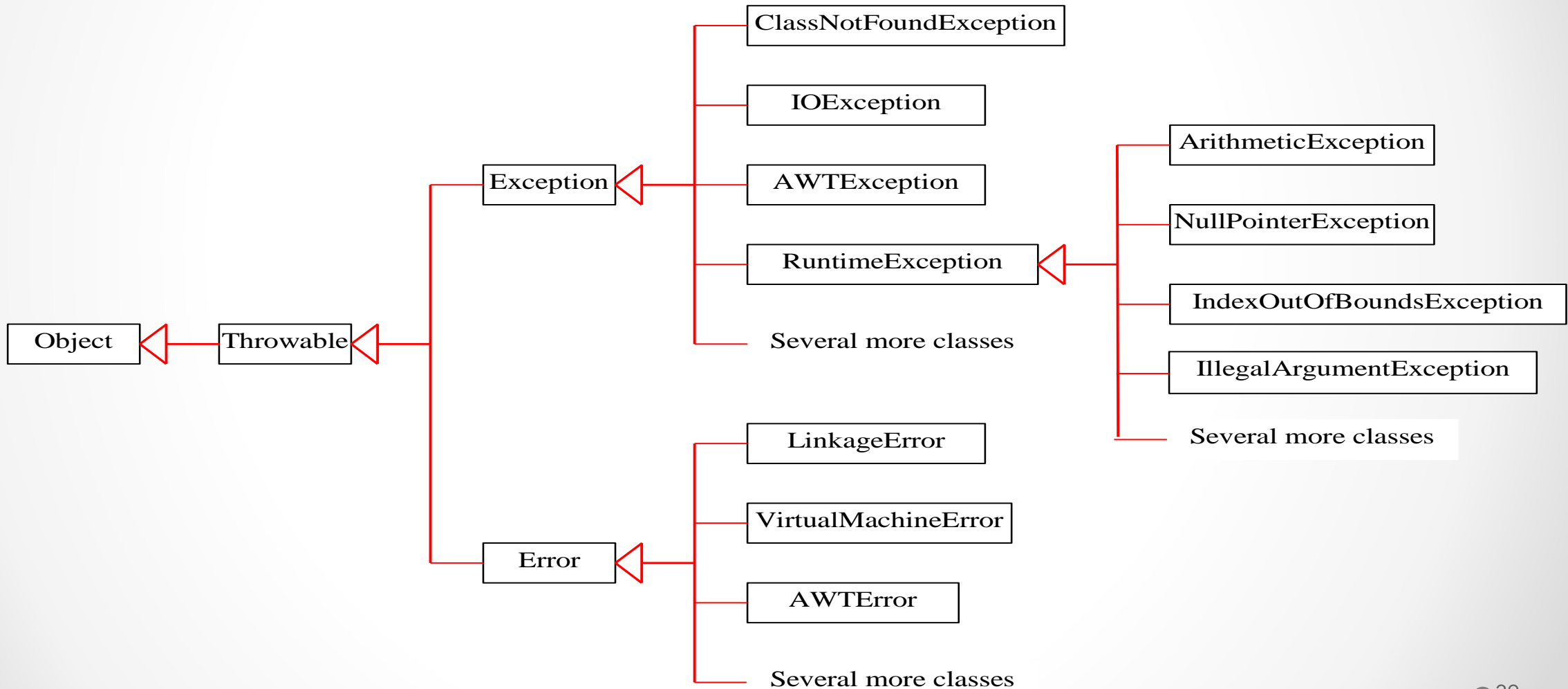
Process completed.
```

## 2.5. Ngoại lệ Phân loại



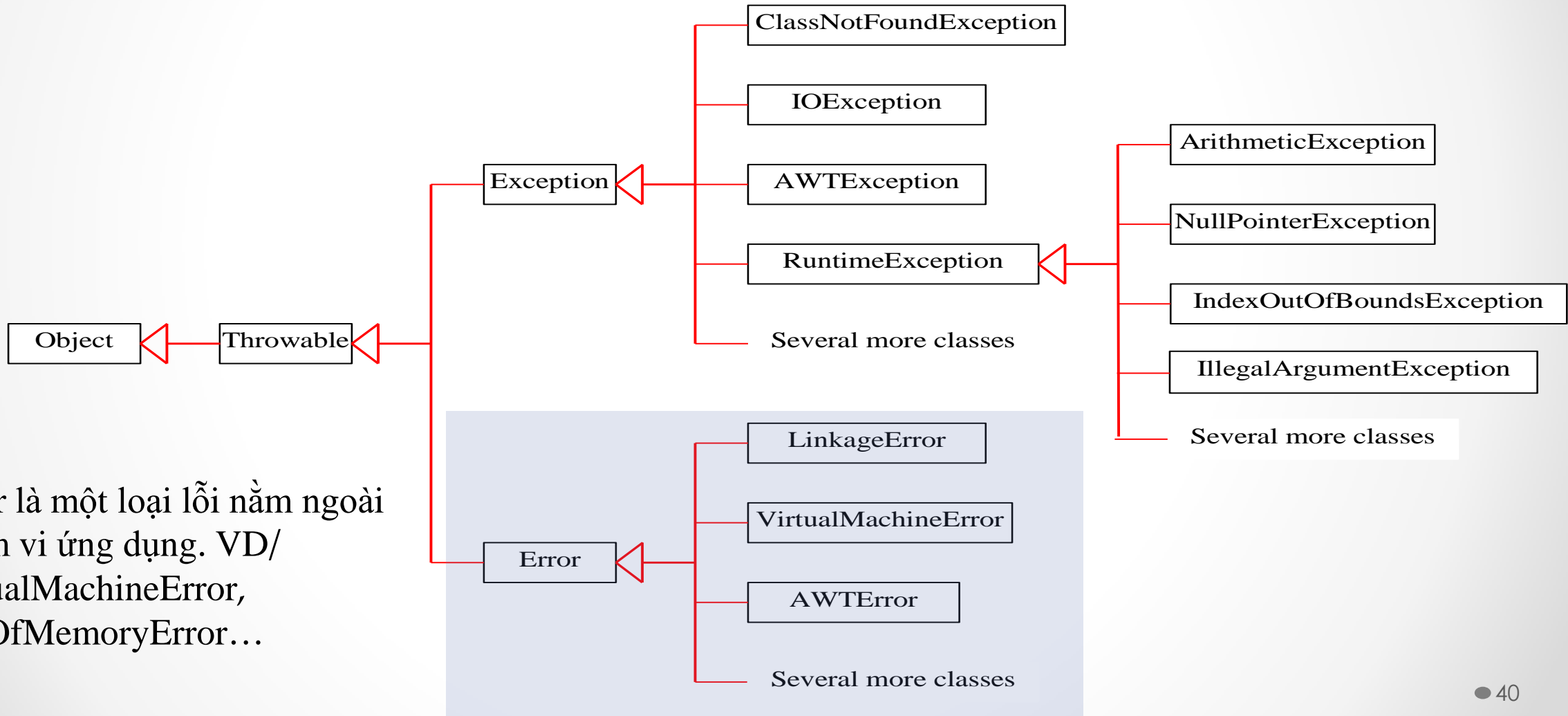
- Trong Java có 2 loại exception là `Checked Exception` và `Unchecked Exception`
  - `Checked Exception`: là các exception xảy ra tại thời điểm compile. Những exception này thường liên quan đến lỗi cú pháp (syntax) và bắt buộc chúng ta phải "bắt" (catch) nó.
  - `Unchecked Exception`: là các exception xảy ra tại thời điểm runtime. Những exception này thường liên quan đến lỗi logic và không bắt buộc chúng ta phải "bắt" (catch) nó.

## 2.5. Ngoại lệ Phân cấp



## 2.5. Ngoại lệ

### Phân cấp: Error



Error là một loại lỗi nằm ngoài phạm vi ứng dụng. VD/  
VirtualMachineError,  
OutOfMemoryError...

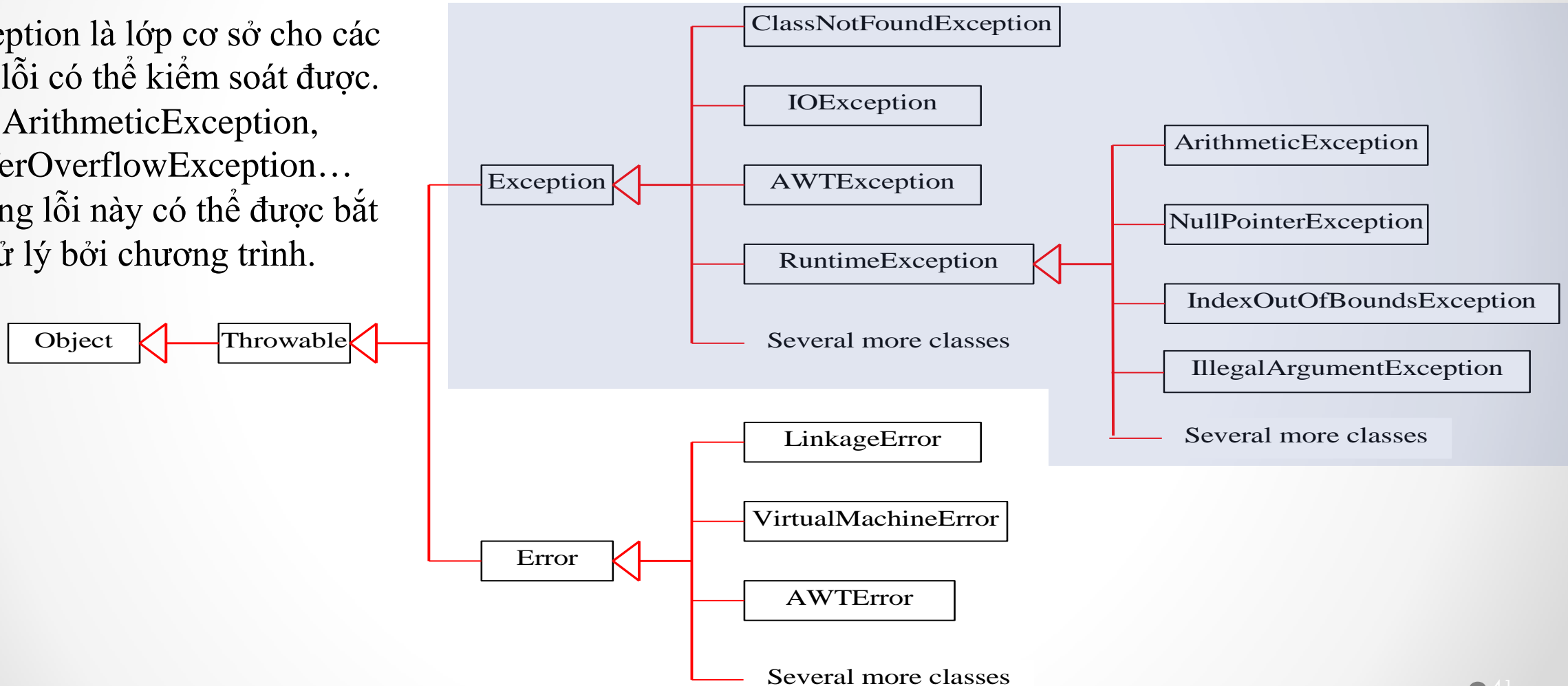


## 2.5. Ngoại lệ

# Phân cấp: Exception

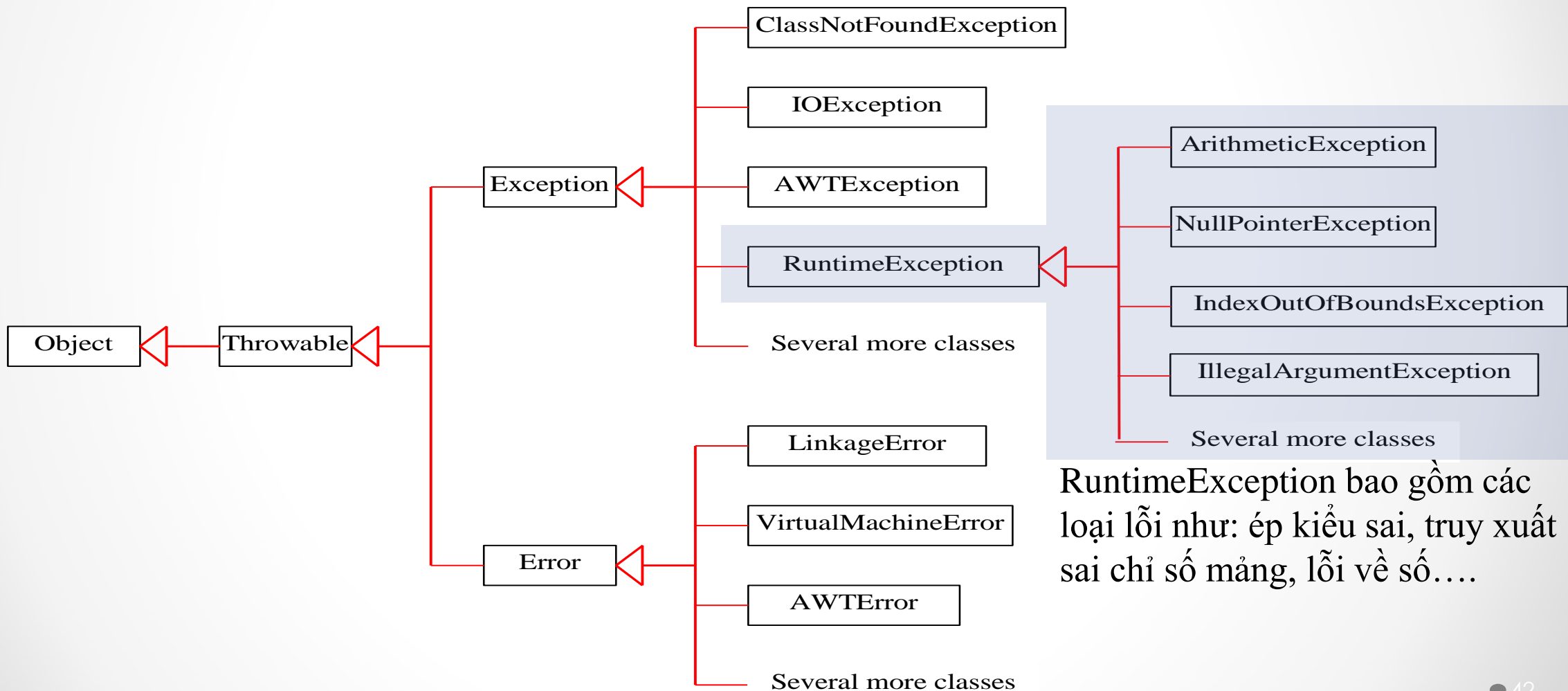


Exception là lớp cơ sở cho các loại lỗi có thể kiểm soát được.  
VD: `ArithmeticException`, `BufferOverflowException`...  
Những lỗi này có thể được bắt và xử lý bởi chương trình.



# 2.5. Ngoại lệ

## Phân cấp: RuntimeException



## 2.5. Ngoại lệ

# Xử lý ngoại lệ trong Java

---



- Keyword
  - try ... catch
  - finally
  - throw
  - throws

## 2.5. Ngoại lệ

# Sử dụng 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 với ngoại lệ

```
try {  
    // Đoạn mã có thể gây ra Exception  
}  
catch (ExceptionType e) {  
    // Xử lý Exception  
}
```

- ExceptionType là một đối tượng của lớp Throwable

## 2.5. Ngoại lệ

### Sử dụng try...catch: Ví dụ 1



```
public class Zero {  
    public static void main (String[] args)    {  
        try {  
            int tu = 10;  
            int mau = 0;  
            System.out.println (tu/mau);  
            System.out.println ("This text will not be printed.");  
        }  
        catch (ArithmeticException e) {  
            System.out.println ("Không thể chia cho 0");  
        }  
    }  
}
```

## 2.5. Ngoại lệ

# ★ Sử dụng try...catch: Nhiều khối catch

- Một đoạn mã có thể gây ra nhiều hơn một ngoại lệ: Sử dụng nhiều khối catch

```
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 (trong cây phân cấp kế thừa)

## 2.5. Ngoại lệ

### Sử dụng try...catch: Ví dụ 2



```
import java.util.Scanner;
class Zero2 {
    public static void main (String args[]){
        int no1, no2;
        try {
            Scanner sc = new Scanner (System.in);
            System.out.print("Input no1: ");    no1 = sc.nextInt();
            System.out.print("Input no2: ");    no2 = sc.nextInt();
            System.out.println("Division result is: " + no1/no2);
        }
        catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero");    }
        catch (InputMismatchException e) {
            System.out.println("You must input an integer");    }
    }
}
```

## 2.5. Ngoại lệ Khối finally



- Đảm bảo thực hiện tất cả các công việc cần thiết dù ngoại lệ có xảy ra hay không
  - Đóng file, đóng socket, connection
  - Giải phóng tài nguyên (nếu cần)...

```
try {  
    // Đoạn mã có thể gây ra ngoại lệ  
}  
catch (ExceptionType e) {  
    // Xử lý ngoại lệ  
}  
finally {  
    /* Thực hiện tất cả các công việc dù ngoại lệ có xảy ra hay không */  
}
```



## 2.5. Ngoại lệ

# Sử dụng throw



- Sử dụng throw khi người lập trình muốn tự phát sinh ra ngoại lệ, ví dụ:

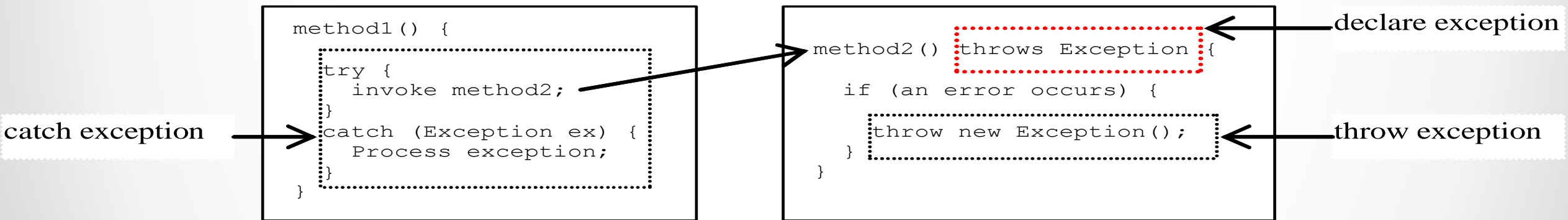
```
public static void main (String[] args)  {  
    try {  
        final int MIN = 25, MAX = 40;  
        Scanner scan = new Scanner (System.in);  
        System.out.print ("Enter an integer value between " + MIN + " and " + MAX + ", inclusive: ");  
        int value = scan.nextInt();  
        if (value < MIN || value > MAX)  
            throw new Exception();           // make an exception  
        else  
            System.out.println("Gia tri nhap hop le");  
        System.out.println ("End of main method."); // may never be reached  
    }  
    catch( Exception x ) {                 // catch the exception  
        System.out.println("Number is not valid.");  
    }  
}
```

## 2.5. Ngoại lệ

# Sử dụng throws



- Trong trường hợp phương thức có phát sinh ra ngoại lệ, nếu không muốn “bắt” nó thì sử dụng throws để báo cho chương trình biết là phương thức này sẽ ném ra ngoại lệ...



## 2.5. Ngoại lệ

### Sử dụng throws: Ví dụ



```
public class DelegateExceptionDemo {  
    public static void main(String args[]){  
        try {  
            int num = calculate(9,3);  
            System.out.println("Lan 1: " + num);  
            num = calculate(9,0);  
            System.out.println("Lan 2: " + num);  
        }  
        catch(Exception e) {  
            System.out.println(e.getMessage());  
        }  
    }  
}
```

```
static int calculate(int tu, int mau) throws ArithmeticException {  
    if (mau== 0)  
        throw new ArithmeticException("Khong the chia cho 0!");  
    int num = tu/ mau;  
    return num;  
}  
}
```



# Kiến thức cần nắm



1. Xuất (dùng cộng chuỗi *print/println* và dùng định dạng *printf*)
2. Nhập (các loại giá trị) (Slide 30, 31)
3. Phân biệt các kiểu dữ liệu (Slide 11-14)
4. Khai báo biến, hằng (Slide 15-17), mảng (Slide 27-29)
5. Phân biệt cách dùng biến toàn cục, biến cục bộ (Slide 18)
6. Ngoại lệ: bắt ngoại lệ, ném ngoại lệ (Slide 37-51)
7. Viết hàm (Slide 26)

# Demo



1. Viết chương trình in ra dòng chữ HELLO
2. Viết chương trình in kết quả cộng 2 số nguyên (cho nhập); ct ghép 2 chuỗi
3. Viết chương trình tính tổng các số chẵn trong khoảng từ 1 đến n
4. Viết chương trình tính tổng các số chẵn trong ds cho trước

# Review questions



1. Cần cài chương trình gì để viết Java?
2. Hàm bắt đầu chạy chương trình Java có tên là gì? Cú pháp hàm như thế nào?
3. Có bao nhiêu kiểu dữ liệu cơ sở trong Java? Bao nhiêu kiểu dữ liệu tham chiếu?
4. 4 kiểu dữ liệu số nguyên trong Java?
5. Toán tử *new* dùng để làm gì?
6. Kiểu chuỗi trong Java có tên là gì? Thuộc về kiểu dữ liệu gì?
7. Kể tên một số kiểu dữ liệu tham chiếu.
8. Ngoại lệ là gì. Làm thế nào để xử lý ngoại lệ.
9. Làm sao bắt được ngoại lệ, ném ngoại lệ. Khi nào cần ném ngoại lệ.

# Exercises



**Bài 1:** Nhập vào 2 số nguyên, in ra tổng, hiệu, tích, thương của 2 số. Chú ý kiểm tra số thứ 2 có khác không hay không, nếu bằng 0 thì thông báo “Không thể chia cho 0!!”.

Yêu cầu kiểm tra dữ liệu nhập phải là số.

**Bài 2:** Giải phương trình bậc nhất.

**Bài 3:** Viết chương trình tính tổng các số nguyên tố từ 1-> N. N được nhập từ bàn phím. Yêu cầu viết phương thức kiểm tra số nguyên tố, phương thức tính tổng, kiểm tra giá trị N.

**Bài 4:** Nhập vào một chuỗi. Đếm số ký tự, số từ trong chuỗi vừa nhập. In mỗi từ trên một dòng. Yêu cầu viết hàm cho từng thao tác.

**Bài 5:** Các bài tập Module 1