

Non-OOP Session 1

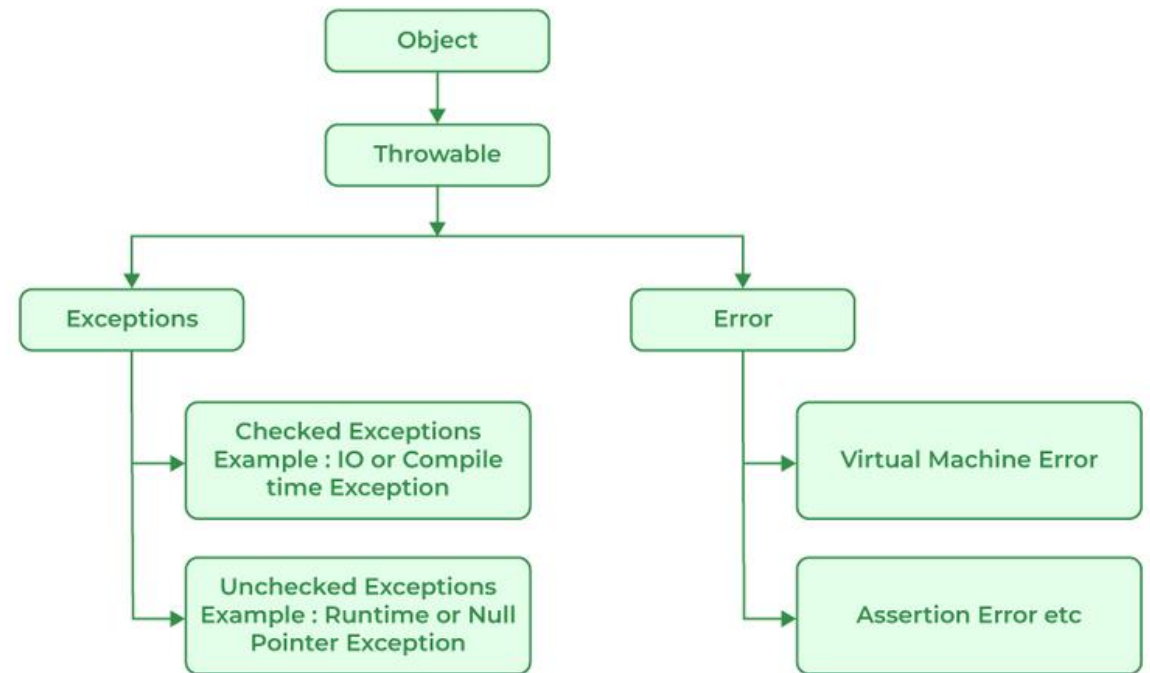
Exception Handling

Exceptions

- Exception (Ngoại lệ): là kết quả chương trình trả về không mong muốn, sẽ làm crash chương trình.
- Một số lý do gây ra Exception:
 - Invalid user input
 - Device failure
 - Loss of network connection
 - Physical limitations (out-of-disk memory)
 - Code errors
 - Opening an unavailable file

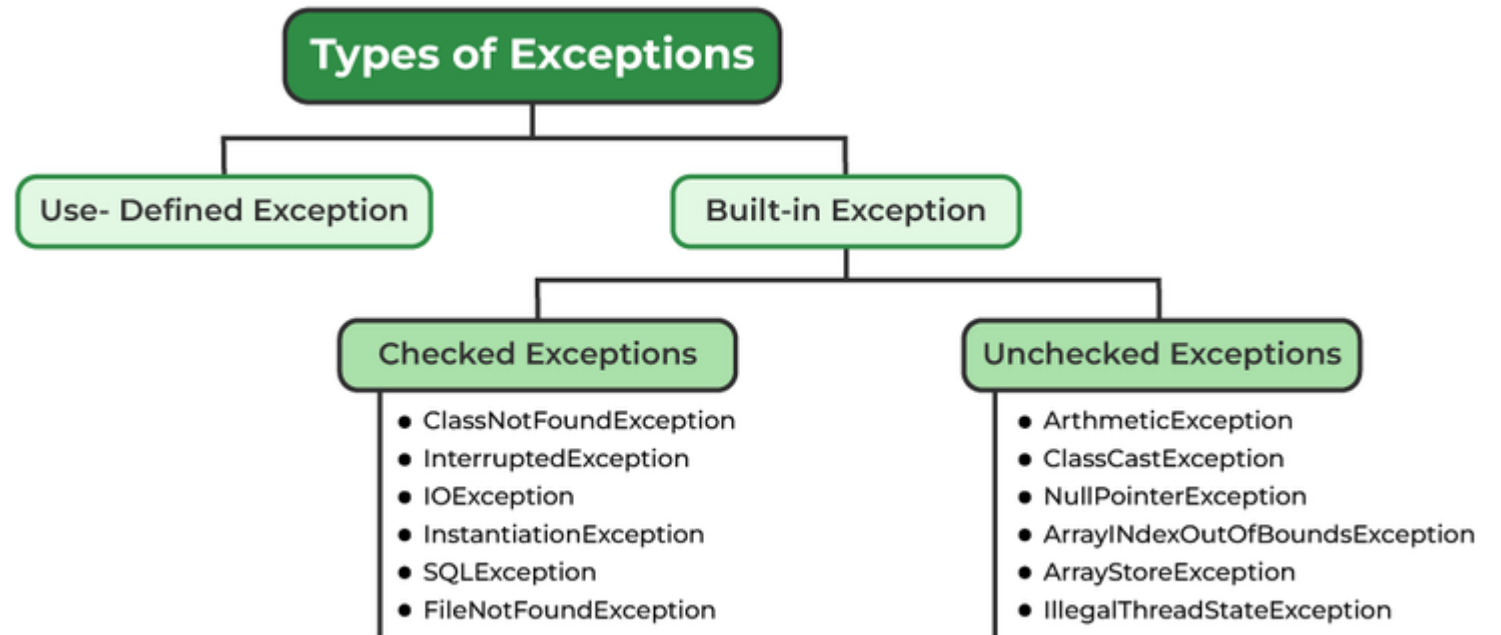
Exception vs Error

- Trong Java, Class Throwable sẽ là lớp cha của Exceptions và Error.
 - Exception: Là các lỗi mà người lập trình viên có thể điều chỉnh xử lý trong quá trình runtime. Ví dụ: FileNotFoundException, NullPointerException
 - Error: Là các lỗi không thể xử lý trong quá trình runtime. Ví dụ: StackOverflowError, Infinite loop, memory leaks.
- Java sẽ tự định nghĩa, đâu là error và đâu là exception



Types of Exception

- 2 loại chính:
 - Exception từ phía người dung tự tạo
 - Exception từ phía Java build sẵn



Handling Exception



Java Program

Programmer nhận exception
(**catch**) từ program



Programmer

Programmer
Xử lý exception và trả về
(**throw**) cho người dùng



End-user

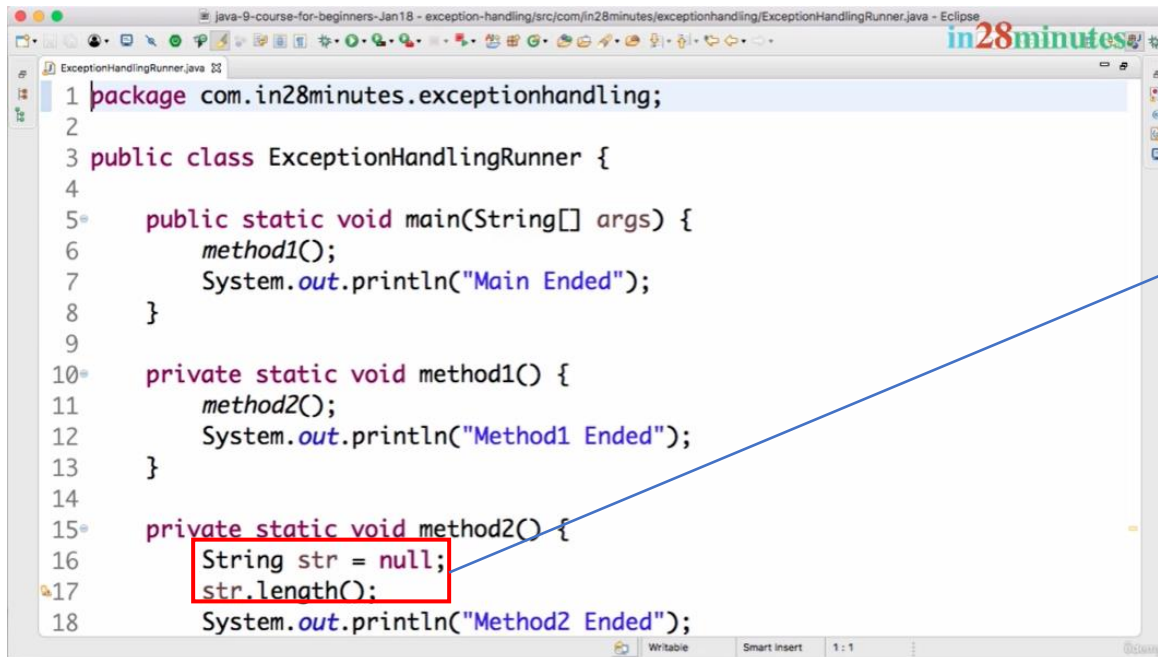
Handling Exception

*Ví dụ về trường hợp gặp exception

- Nhận lỗi (**catch**):
 - 1. Nhận lỗi từ Java Program
 - 2. Nhận lỗi để làm gì?
 - 3. Từ khóa Finally
- Trả lỗi (**Throw**):
 - 4. Trả lỗi mặc định cho người dùng. *ex.printStackTrace*
 - 5. Trả lỗi custom cho người dùng

Handling Exception

* Ví dụ đoạn code dưới gặp exception: NullPointerException – lỗi khi cố gắng truy cập một biến là null.



```
1 package com.in28minutes.exceptionhandling;
2
3 public class ExceptionHandlingRunner {
4
5     public static void main(String[] args) {
6         method1();
7         System.out.println("Main Ended");
8     }
9
10    private static void method1() {
11        method2();
12        System.out.println("Method1 Ended");
13    }
14
15    private static void method2() {
16        String str = null;
17        str.length();
18        System.out.println("Method2 Ended");
19    }
20 }
```

Đoạn code sẽ trả về lỗi ở dòng code này

Và ngay lập tức dừng chương trình
-> Không chạy các lệnh System.out.println

Handling Exception

- 1. Nhận lỗi từ Java Program

```
6      method1();
7      System.out.println("Main Ended");
8  }
9
10 private static void method1() {
11     method2();
12     System.out.println("Method1 Ended");
13 }
14
15 private static void method2() {
16     try {
17         String str = null;
18         str.length();
19         System.out.println("Method2 Ended");
20     } catch (Exception ex) {
21     }
22 }
23 }
```

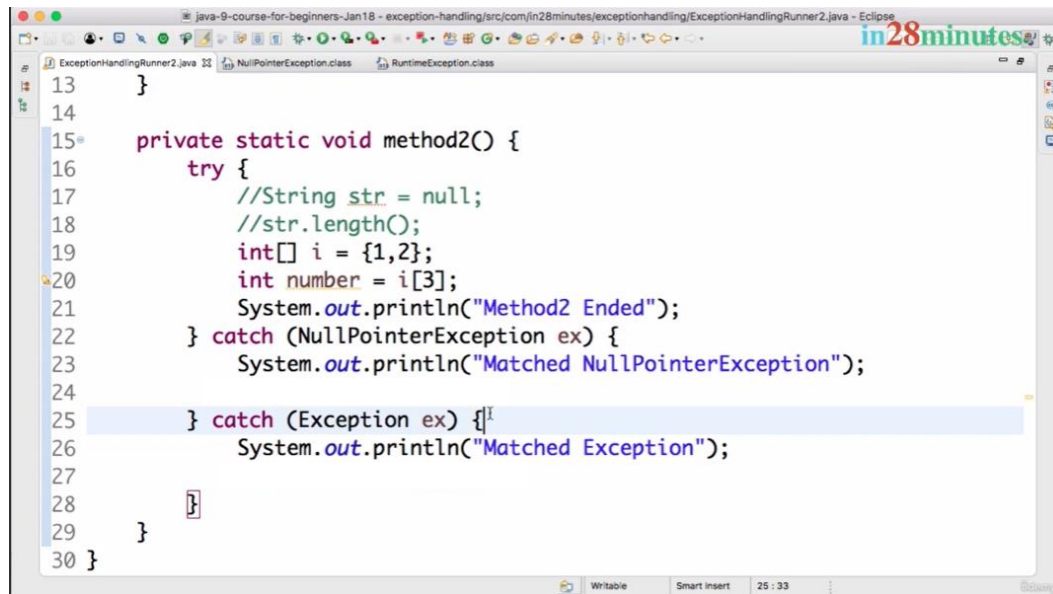
Báo rằng: Đoạn code trong try có thể sẽ phát sinh lỗi khi chạy.
-> Tricky code – đoạn code có rủi ro lỗi

Khi xảy ra lỗi, thì sẽ chạy trong đoạn code ở dưới

Chương trình vẫn tiếp tục chạy và in ra hai dòng lệnh println.

Handling Exception

- 2. Nhận lỗi từ Java Program để làm gì?
 - Làm chương trình không dừng lại
 - Xử lý lỗi ngay khi chương trình chạy



```
13 }
14
15 private static void method2() {
16     try {
17         //String str = null;
18         //str.length();
19         int[] i = {1,2};
20         int number = i[3];
21         System.out.println("Method2 Ended");
22     } catch (NullPointerException ex) {
23         System.out.println("Matched NullPointerException");
24     }
25     catch (Exception ex) {
26         System.out.println("Matched Exception");
27     }
28 }
29 }
30 }
```

Một try block có thể xử lý nhiều trường hợp lỗi khác nhau trong block try:

- Có khả năng NullPointerException khi biến trong là null và bị truy cập
- Có khả năng bị lỗi out of array khi biến truy cập out index.

Ứng với mỗi trường hợp lỗi, ta có thể xử lý riêng biệt.

Handling Exception

- 3. Vậy thì nếu một statement phải luôn thực thi dù có lỗi hay không thì sao? **Từ khóa Finally**

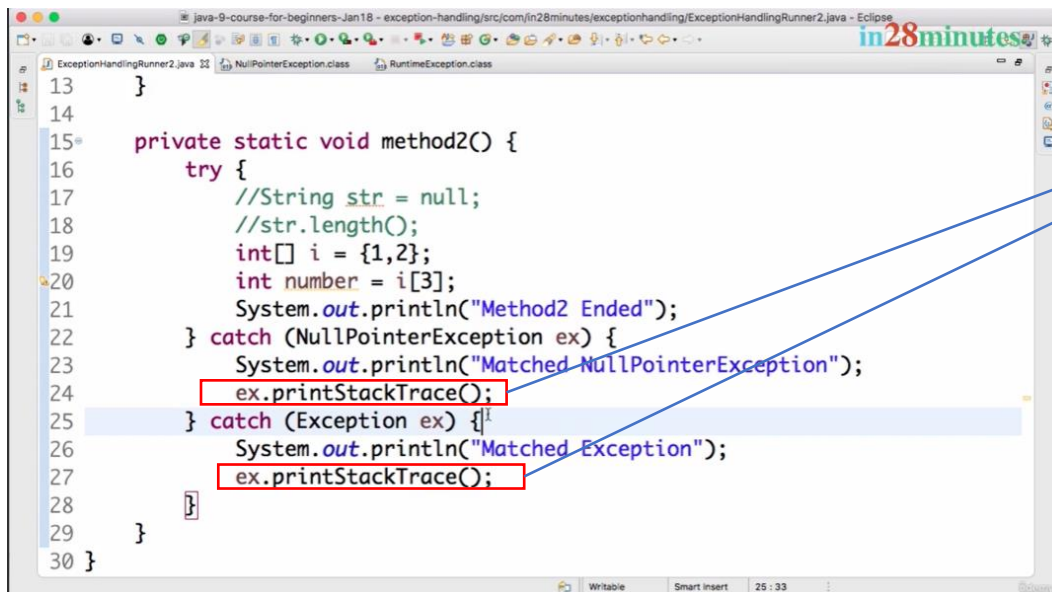
```
ObjectOutputStream oos = null;
try {
    oos = new ObjectOutputStream(new FileOutputStream(file));
    oos.writeObject(shapes);
    oos.flush();
} catch (FileNotFoundException ex) {
    // complain to user
} catch (IOException ex) {
    // notify user
} finally {
    if (oos != null) {
        try {
            oos.close();
        } catch (IOException ex) {
            // ignore ... any significant errors should already have been
            // reported via an IOException from the final flush.
        }
    }
}
```

Đoạn code dùng để mở file và ghi dữ liệu

Dù đọc file có lỗi hay không thì vẫn thực thi đoạn code trong finally.

Handling Exception

- 4. Trả lỗi mặc định cho người dùng
 - Đưa một good message cho người dùng biết
 - Và chương trình vẫn được chạy tiếp tục

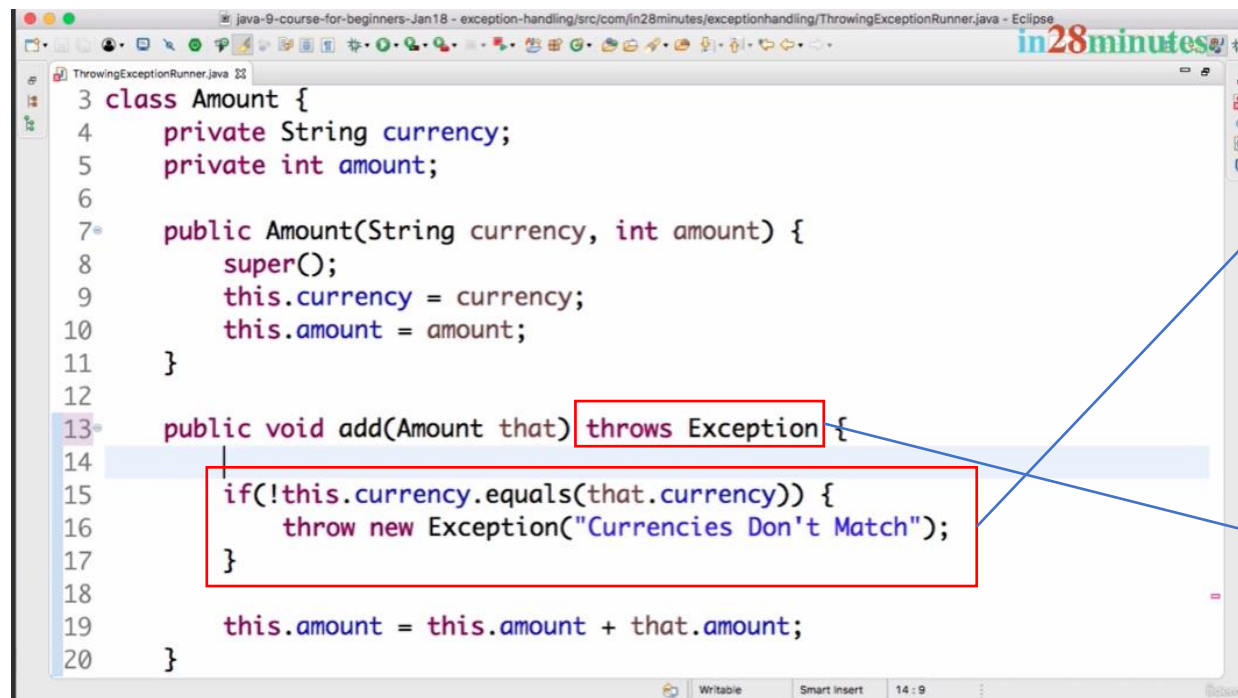


```
13 }
14
15 private static void method2() {
16     try {
17         //String str = null;
18         //str.length();
19         int[] i = {1,2};
20         int number = i[3];
21         System.out.println("Method2 Ended");
22     } catch (NullPointerException ex) {
23         System.out.println("Matched NullPointerException");
24         ex.printStackTrace();
25     } catch (Exception ex) {
26         System.out.println("Matched Exception");
27         ex.printStackTrace();
28     }
29 }
30 }
```

Trả về lỗi của chương trình cho người dùng
và vẫn chạy chương trình

Handling Exception

- 5. Trả lỗi custom của người lập trình viên.



```
1 class Amount {
2     private String currency;
3     private int amount;
4
5     public Amount(String currency, int amount) {
6         super();
7         this.currency = currency;
8         this.amount = amount;
9     }
10
11     public void add(Amount that) throws Exception {
12         if(!this.currency.equals(that.currency)) {
13             throw new Exception("Currencies Don't Match");
14         }
15
16         this.amount = this.amount + that.amount;
17     }
18 }
```

Lỗi không nằm trong danh sách mặc định của ngôn ngữ Java

Dùng từ khóa **throw** để tạo và “ném” lỗi cho người dùng

Định dạng hàm này sẽ có thể trả về lỗi

Ví dụ thực tế cho từng trường hợp