ĐẠI NAM
UNIVERSITY

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

CHƯƠNG 7. XỬ LÝ NGOẠI LỆ

NỘI DUNG

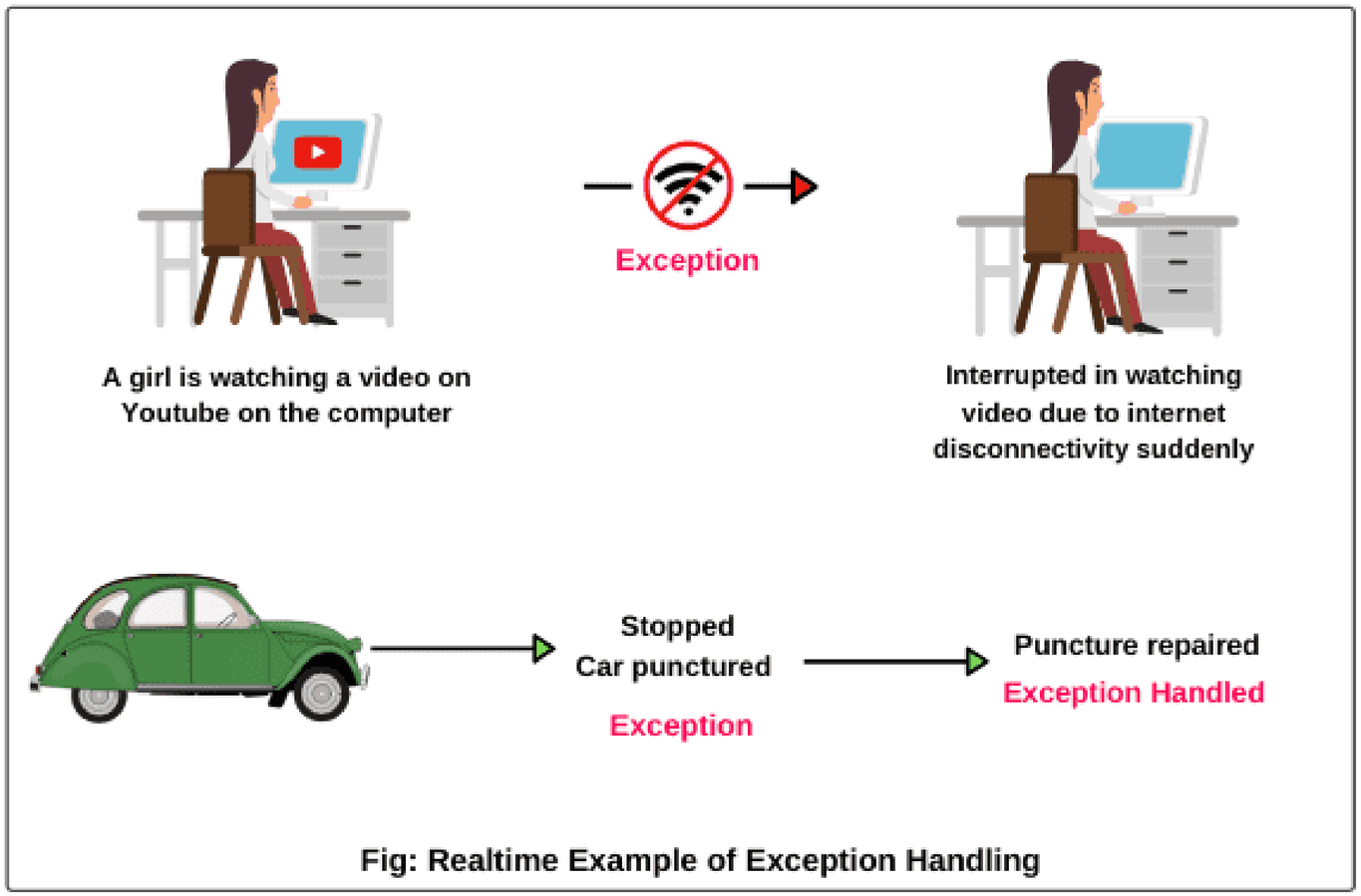
ɪkˈsɛpʃᵊn

1. Giới thiệu về Exception
2. Cơ chế xử lý Exception
3. Các Exception thường gặp và cách xử lý
4. Tạo và sử dụng Custom Exception

1. GIỚI THIỆU VỀ EXCEPTION

- **Liên hệ:** Chương trình máy tính cũng giống như một công thức nấu ăn.
  - Trong công thức, có các bước hướng dẫn rõ ràng để làm ra một món ăn ngon.
  - Chương trình máy tính cũng vậy, nó có các dòng lệnh hướng dẫn máy tính thực hiện từng bước để hoàn thành một nhiệm vụ nào đó. Nhưng đôi khi, có những sự cố bất ngờ xảy ra:
  - Khi nấu ăn, có thể bạn hết nguyên liệu, bếp bị tắt đột ngột, hoặc làm đổ vỡ thứ gì đó.
  - Trong chương trình máy tính cũng vậy, có thể gặp những tình huống không lường trước được, ví dụ:Cố gắng chia một số cho 0 (không thể thực hiện được).Cố gắng mở một tệp tin không tồn tại.Mạng internet bị mất kết nối khi đang tải dữ liệu.
- Những sự cố này gọi là "**Exception**" (ngoại lệ):
  - Exception giống như những tình huống "ngoài ý muốn", làm gián đoạn quá trình thực hiện chương trình.
  - Nếu không xử lý exception, chương trình có thể bị dừng đột ngột và không hoàn thành nhiệm vụ.
- **Exception** là một sự kiện bất thường xảy ra trong quá trình thực thi chương trình, làm gián đoạn luồng thực thi bình thường.

1. GIỚI THIỆU VỀ EXCEPTION




```
int[] numbers = {1, 2, 3};
System.out.println(numbers[5]); // ArrayIndexOutOfBoundsException
```

1. GIỚI THIỆU VỀ EXCEPTION

- **Vậy làm thế nào để xử lý Exception?**
  - Giống như khi nấu ăn, nếu gặp sự cố, bạn sẽ tìm cách giải quyết để tiếp tục nấu ăn.
  - Trong lập trình, chúng ta sử dụng các đoạn mã đặc biệt để "bắt" và "xử lý" exception. Ví dụ:
    - Nếu chia cho 0, chương trình sẽ thông báo lỗi và yêu cầu nhập lại số khác.
    - Nếu không tìm thấy tệp tin, chương trình sẽ tạo một tệp tin mới.
    - Nếu mất kết nối mạng, chương trình sẽ chờ một lát rồi thử kết nối lại.
- **Tầm quan trọng của việc xử lý ngoại lệ**:
  - Xử lý exception giúp chương trình trở nên mạnh mẽ, đáng tin cậy và có khả năng phục hồi khi gặp lỗi.
- **Hãy nhớ**:
  - Exception là một phần không thể tránh khỏi trong lập trình.
  - Học cách xử lý exception là một kỹ năng quan trọng để trở thành một lập trình viên giỏi.

1. GIỚI THIỆU VỀ EXCEPTION

- **Phân loại Exception?**
  - **Checked exceptions (Ngoại lệ được kiểm tra)**: Các exception mà lập trình viên buộc phải xử lý hoặc khai báo bằng từ khóa throws. CheckedException là những exception có thể nhìn thấy và kiểm tra tại thời điểm biên dịch
    - Đi du lịch nước ngoài, cần phải kiểm tra hộ chiếu, visa trước khi lên máy bay. Nếu thiếu giấy tờ, sẽ không được phép đi.
    - Trong lập trình, Checked Exception là những lỗi mà chương trình "biết trước" có thể xảy ra, và "bắt buộc" phải xử lý hoặc thông báo trước.

- **Ví dụ**: IOException, SQLException
  - Khi đọc một tệp tin, chương trình cần kiểm tra xem tệp tin có tồn tại không. Nếu không tồn tại, nó sẽ báo lỗi "FileNotFoundException" và bạn phải xử lý lỗi này.
  - Khi kết nối đến cơ sở dữ liệu, chương trình cần kiểm tra xem kết nối có thành công không. Nếu không thành công, nó sẽ báo lỗi "SQLException" và bạn phải xử lý lỗi này.

1. GIỚI THIỆU VỀ EXCEPTION

- **Phân loại Exception?**
  - **Unchecked exceptions**: UncheckedException là các exception không được kiểm tra tại thời điểm biên dịch. Hầu hết các unchecked exception xảy ra do dữ liệu không hợp lệ trong quá trình tương tác với chương trình
    - Giống như khi bạn đang đi trên đường, có thể bất ngờ gặp một ổ gà hoặc một chiếc xe vượt đèn đỏ. Bạn không thể biết trước những tình huống này, nhưng bạn cần phải phản ứng nhanh để tránh tai nạn.
    - **Ví dụ**: NullPointerException, ArrayIndexOutOfBoundsException.
      - Khi chia một số cho 0, chương trình sẽ báo lỗi "ArithmeticException" vì đây là phép toán không hợp lệ.
      - Khi cố gắng truy cập một phần tử không tồn tại trong mảng, chương trình sẽ báo lỗi "ArrayIndexOutOfBoundsException".

1. GIỚI THIỆU VỀ EXCEPTION

- **Phân loại Exception?**
  - **Errors**: Các lỗi nghiêm trọng mà chương trình thường không thể phục hồi.
    - Giống như những thảm họa thiên nhiên như động đất, sóng thần, rất khó để dự đoán và thường gây ra hậu quả nghiêm trọng.
    - Trong lập trình, Error là những lỗi cực kỳ nghiêm trọng, thường liên quan đến hệ thống hoặc phần cứng, và chương trình rất khó để phục hồi.
    - **Ví dụ**: OutOfMemoryError, StackOverflowError.
      - Khi máy tính hết bộ nhớ, chương trình sẽ báo lỗi "OutOfMemoryError".
      - Khi chương trình gọi đệ quy quá nhiều lần, nó sẽ báo lỗi "StackOverflowError".

2. CƠ CHẾ XỬ LÝ EXCEPTION

- **Khối try-catch-finally**:
  - **try**: Chứa đoạn code có thể phát sinh exception.
  - **catch**: Xử lý exception cụ thể. Có thể có nhiều khối catch để xử lý các loại exception khác nhau.
  - **finally**: Chứa đoạn code luôn được thực thi, dù có exception xảy ra hay không. Thường dùng để giải phóng tài nguyên.
    - *Sau khi tổ chức một bữa tiệc vui vẻ, dù có chuyện gì xảy ra trong bữa tiệc, bạn vẫn cần phải dọn dẹp sạch sẽ sau đó. Dù khách mời có làm đổ nước, làm vỡ ly, hay thậm chí có một cơn mưa bất ngờ ập đến, việc dọn dẹp vẫn phải được thực hiện.*
    - *Anh vẫn đến dù trời gió mưa giông giá rét*
- **Từ khóa throw và throws**:
  - **throw**: Dùng để ném ra một exception một cách tường minh.
  - **throws**: Dùng để khai báo các exception mà một phương thức có thể ném ra.

2. CƠ CHẾ XỬ LÝ EXCEPTION

```
try {
    int result = 10 / 0; // ArithmeticException
} catch (ArithmeticException e) {
    System.out.println("Lỗi chia cho 0: " + e.getMessage());
} finally {
    System.out.println("Khối finally luôn được thực thi");
}
```

```
public void checkAge(int age) throws InvalidAgeException {
    if (age < 0) {
        throw new InvalidAgeException("Tuổi không hợp lệ");
    }
}
```

3. CÁC EXCEPTION THƯỜNG GẶP VÀ CÁCH XỬ LÝ

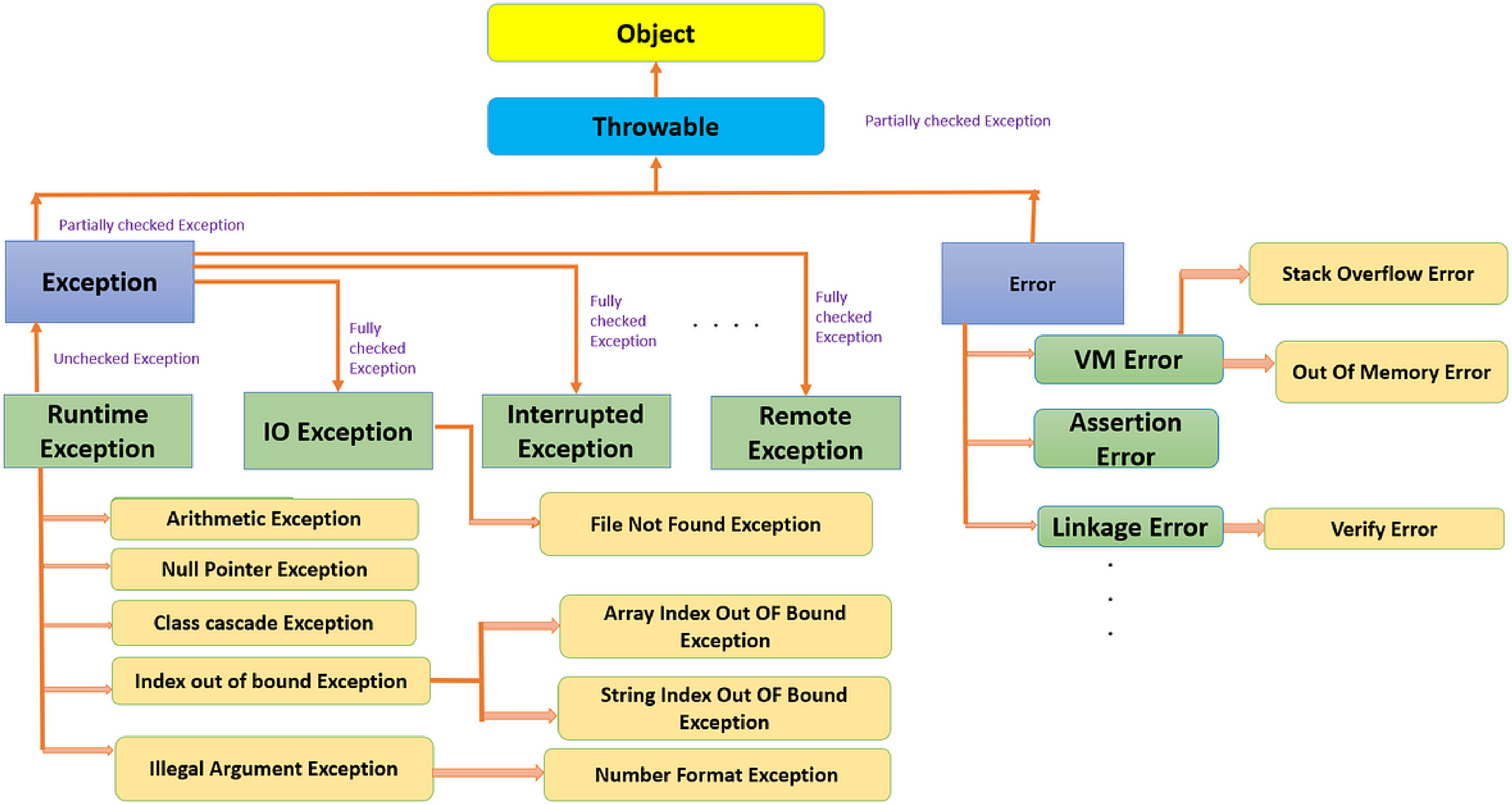


Fig. Exception Hierarchy in Java ~ by Deepti Swain

3. CÁC EXCEPTION THƯỜNG GẶP VÀ CÁCH XỬ LÝ

- **NullPointerException**: Xảy ra khi cố gắng truy cập một đối tượng null.
  - Xử lý: Kiểm tra null trước khi truy cập đối tượng.
- **ArrayIndexOutOfBoundsException**: Xảy ra khi cố gắng truy cập một phần tử ngoài giới hạn của mảng.
  - Xử lý: Kiểm tra chỉ số trước khi truy cập phần tử mảng.
- **IOException**: Xảy ra khi có lỗi trong quá trình nhập xuất dữ liệu.
  - Xử lý: Sử dụng khối try-catch để bắt và xử lý exception, ví dụ như đóng các luồng dữ liệu.
- **SQLException**: Xảy ra khi có lỗi trong quá trình tương tác với cơ sở dữ liệu.
  - Xử lý: Sử dụng khối try-catch để bắt và xử lý exception, ví dụ như đóng kết nối cơ sở dữ liệu.
- **Các exception khác**: Giới thiệu một số exception khác thường gặp và cách xử lý tương ứng.

3. CÁC EXCEPTION THƯỜNG GẶP VÀ CÁCH XỬ LÝ

```
String str = null;
if (str != null) {
    System.out.println(str.length());
}

int[] numbers = {1, 2, 3};
if (index >= 0 && index < numbers.length) {
    System.out.println(numbers[index]);
}

try {
    FileReader file = new FileReader("file.txt");
    // Đọc dữ liệu từ file
    file.close();
} catch (IOException e) {
    System.out.println("Lỗi đọc file: " + e.getMessage());
}
```

4. TẠO VÀ SỬ DỤNG CUSTOM EXCEPTION

- **Custom Exception** (Ngoại lệ tùy chỉnh; Ngoại lệ đặt làm riêng):
  - Trong lập trình, đôi khi chúng ta gặp những tình huống đặc biệt, không giống với các lỗi thông thường như chia cho 0 hay truy cập phần tử ngoài mảng.
    - **Ví dụ**, bạn đang viết một chương trình quản lý thư viện, và muốn đảm bảo rằng sách không được mượn quá hạn. Nếu một người dùng cố gắng mượn sách quá hạn, bạn muốn chương trình báo lỗi một cách cụ thể, ví dụ như "Lỗi: Không thể mượn sách quá hạn".
  - Để làm điều này, chúng ta cần tạo ra một loại ngoại lệ (exception) mới, "đặt làm riêng" cho tình huống này. Loại ngoại lệ này được gọi là "Custom Exception" (ngoại lệ tùy chỉnh).

4. TẠO VÀ SỬ DỤNG CUSTOM EXCEPTION

- **throws và throw**
  - Hãy tưởng tượng bạn đang chơi trò ném bóng với bạn bè. Trong trò chơi này, throws và throw có thể được hiểu như sau:
    - **throws**: Giống như bạn thông báo trước với bạn bè rằng "Tôi có thể sẽ ném bóng về phía các bạn đấy nhé!".
    - **throw**: Là hành động bạn thực sự ném quả bóng đi.
  - Trong Java, throws và throw cũng hoạt động tương tự, nhưng với các ngoại lệ (exception) thay vì bóng:
    - **throws**: Được sử dụng trong khai báo của một phương thức để chỉ ra rằng phương thức đó có thể ném ra một hoặc nhiều ngoại lệ cụ thể.
    - **throw**: Được sử dụng bên trong một phương thức để ném ra một ngoại lệ cụ thể. Khi một ngoại lệ được ném ra, luồng thực thi bình thường của chương trình bị gián đoạn và ngoại lệ đó được truyền lên cho các phương thức gọi nó để xử lý.

4. TẠO VÀ SỬ DỤNG CUSTOM EXCEPTION

- **Tình huống**: Hãy tưởng tượng bạn đang viết một chương trình đơn giản để quản lý điểm số của học sinh. Bạn muốn đảm bảo rằng điểm số luôn nằm trong khoảng từ 0 đến 10. Nếu có ai đó cố tình nhập điểm số nằm ngoài khoảng này, bạn muốn chương trình báo lỗi một cách rõ ràng và cụ thể.

```
class DiemSoKhongHopLeException extends Exception {
    public DiemSoKhongHopLeException(String message) {
        super(message);
    }
}

public void kiemTraDiemSo(int diem) throws DiemSoKhongHopLeException {
    if (diem < 0 || diem > 10) {
        throw new DiemSoKhongHopLeException("Điểm số phải nằm trong khoảng từ 0 đến 10.");
    }
    // ... (xử lý điểm số nếu hợp lệ)
}

try {
    kiemTraDiemSo(12); // Cố tình nhập điểm số không hợp lệ
} catch (DiemSoKhongHopLeException e) {
    System.out.println(e.getMessage()); // In ra thông báo lỗi cụ thể
}
```

4. TẠO VÀ SỬ DỤNG CUSTOM EXCEPTION

- **Thực hành**: Bạn đang viết một chương trình quản lý tài khoản ngân hàng. Bạn muốn đảm bảo rằng số dư tài khoản không bao giờ âm. Nếu có ai đó cố gắng rút tiền vượt quá số dư hiện có, bạn muốn chương trình báo lỗi một cách rõ ràng và cụ thể.
  - **Bước 1**: Tạo lớp Custom Exception
    - **Tạo lớp**: Tạo một lớp Java mới đại diện cho ngoại lệ tùy chỉnh của bạn. Đặt tên lớp mô tả rõ ràng ngoại lệ, ví dụ: SoDuThapException.
    - **Kế thừa**: Cho lớp mới kế thừa từ lớp Exception hoặc một lớp con của nó (thường là RuntimeException nếu bạn muốn tạo một unchecked exception).
    - Constructor: Cung cấp ít nhất một constructor để lớp ngoại lệ có thể nhận và lưu trữ thông tin về lỗi.

```
class SoDuThapException extends Exception { // Hoặc extends RuntimeException
    public SoDuThapException(String message) {
        super(message); // Gọi constructor của lớp cha để thiết lập thông báo
lỗi
    }
}
```

4. TẠO VÀ SỬ DỤNG CUSTOM EXCEPTION

- **Thực hành**: Bạn đang viết một chương trình quản lý tài khoản ngân hàng. Bạn muốn đảm bảo rằng số dư tài khoản không bao giờ âm. Nếu có ai đó cố gắng rút tiền vượt quá số dư hiện có, bạn muốn chương trình báo lỗi một cách rõ ràng và cụ thể.
  - **Bước 2: Sử dụng Custom Exception**
    - **Ném ngoại lệ (throw)**: Trong phương thức xử lý rút tiền, kiểm tra xem số dư có đủ không. Nếu không đủ, sử dụng từ khóa throw để ném ra một đối tượng **SoDuThapException** mới, kèm theo thông báo lỗi cụ thể.

```
public void rutTien(double soTien) throws SoDuThapException {
    if (soTien > soDuHienTai) {
        throw new SoDuThapException("Số dư không đủ để rút tiền.");
    }
    // ... (Tiếp tục xử lý rút tiền nếu số dư đủ)
}
```

4. TẠO VÀ SỬ DỤNG CUSTOM EXCEPTION

- **Thực hành**: Bạn đang viết một chương trình quản lý tài khoản ngân hàng. Bạn muốn đảm bảo rằng số dư tài khoản không bao giờ âm. Nếu có ai đó cố gắng rút tiền vượt quá số dư hiện có, bạn muốn chương trình báo lỗi một cách rõ ràng và cụ thể.
  - **Bước 3: Bắt và xử lý ngoại lệ (try-catch)**
    - **Bắt ngoại lệ**: Tại nơi gọi **phương thức rutTien**, sử dụng khối **try-catch** để bắt và xử lý ngoại lệ SoDuThapException nếu nó được ném ra.

```
try {
    taiKhoan.rutTien(200000);
} catch (SoDuThapException e) {
    System.out.println("Lỗi: " + e.getMessage()); // In ra thông báo lỗi từ
ngoại lệ
    // ... (Các xử lý khác khi gặp lỗi, ví dụ: thông báo cho người dùng)
}
```

ĐẠI NAM
UNIVERSITY

HỎI & ĐÁP