Lập trình hướng đối tượng

Xử lý ngoại lệ

GV: ThS. Ngô Tiến Đức

Nội dung chính

- Ngoại lệ
- Xử lý ngoại lệ với try – catch
- Xử lý ngoại lệ với throws
- Tùy chỉnh ngoại lệ

Ngoại lệ

- Trong quá trình thực thi chương trình Java có thể sẽ gặp các lỗi
- "Lỗi" mà chúng ta vẫn nhắc tới: Các sự cố xảy ra khiến cho chương trình không cho kết quả như mong đợi
- Định nghĩa về **Error** của Java:
  - Các sự cố nghiêm trọng xảy ra dưới các điều kiện bất thường (abnormal conditions)
  - Không nên cố gắng xử lý
  - VD: OutOfMemoryError, StackOverflowError, UnknownError,...

Ngoại lệ

- Ngoại lệ (**exceptions**): Các vấn đề xảy ra trong quá trình thực thi
  - Java có thể phát hiện được
  - Không phải vấn đề nghiêm trọng ảnh hưởng đến hệ thống
- Khi xảy ra ngoại lệ, chương trình bị gián đoạn và đưa ra các thông báo lỗi (error message)

→ Thuật ngữ: Quăng/ném ra ngoại lệ (**throws** an exception)

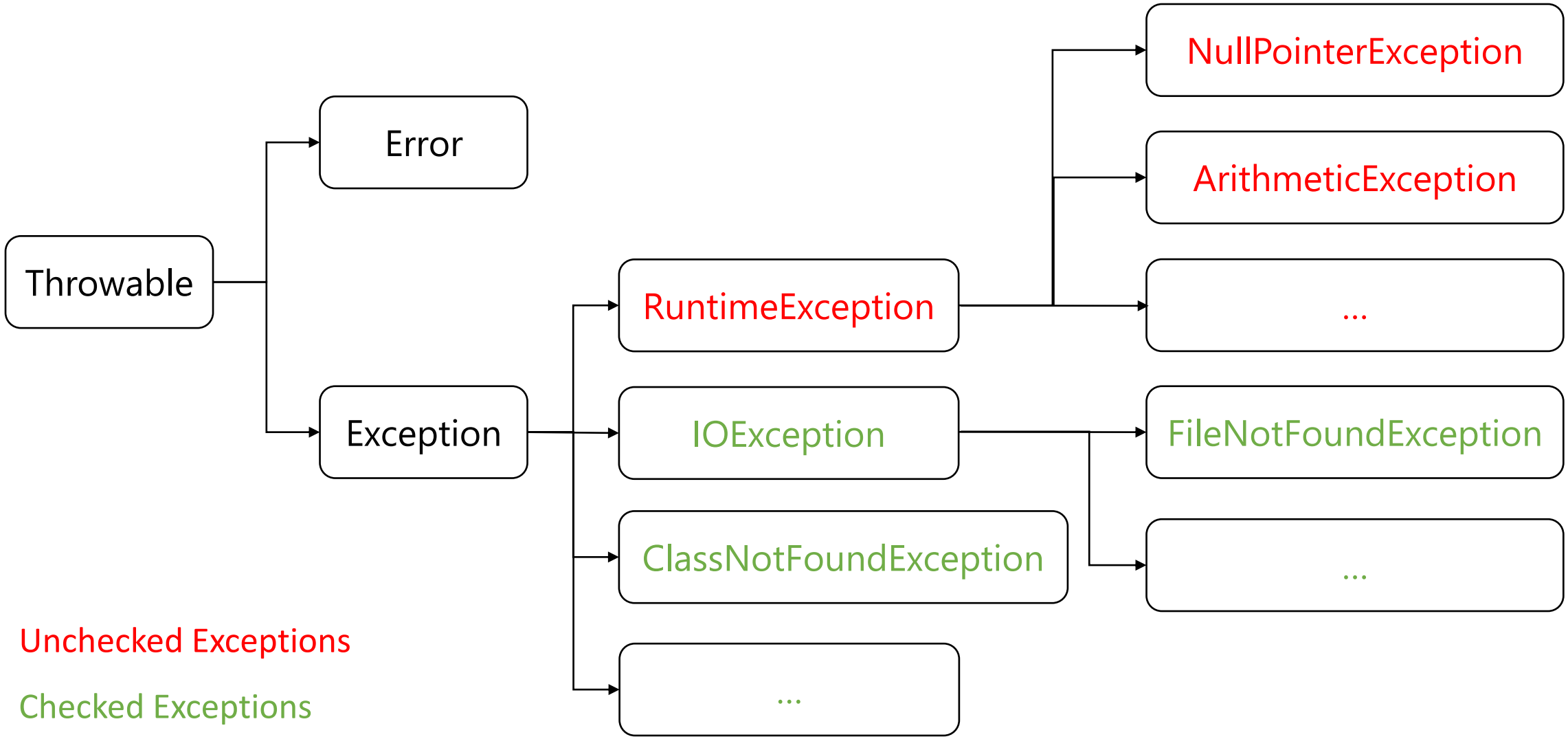
- Để chương trình có thể chạy mà không bị gián đoạn cần bắt (**catch**) được các ngoại lệ → **Exception handling**

Ngoại lệ

Ngoại lệ được chia làm 2 loại:

Checked Exceptions	**Unchecked Exceptions**
Xảy ra trong quá trình biên dịch	Xảy ra trong quá trình thực thi
Có thể phát hiện bởi complier	Không bị phát hiện bởi complier
Được cảnh báo bởi IDE	Không bị phát hiện bởi IDE
Bắt buộc phải xử lý	Không bắt buộc phải xử lý

Ngoại lệ


Throwable
Error
Exception
RuntimeException
IOException
ClassNotFoundException
...
NullPointerException
ArithmeticException
...
FileNotFoundException
...
Unchecked Exceptions
Checked Exceptions

Ngoại lệ

Một số phương thức của lớp Exception:

- `String getMessage()`: In ra mô tả về lỗi dẫn đến ngoại lệ dưới dạng String
- `String toString()`: In ra thông tin về ngoại lệ dưới dạng String (tên ngoại lệ + kết quả từ `getMessage()`)
- `void printStackTrace()`: In ra trình tự các lời gọi phương thức dẫn đến câu lệnh gây ra ngoại lệ

Xử lý ngoại lệ với try - catch

```
try {
      // các câu lệnh có thể gây ra ngoại lệ
} catch (<ExceptionType1> <exception_name_1>) {
      // các câu lệnh thực thi nếu xảy ra ngoại lệ ExceptionType1
}
…
catch (<ExceptionTypeN> <exception_name_N>) {
      // các câu lệnh thực thi nếu xảy ra ngoại lệ ExceptionTypeN
}
```

Xử lý ngoại lệ với try - catch

- Nếu không có ngoại lệ nào xảy ra: Bỏ qua (các) khối catch và thực thi tiếp các câu lệnh sau (các) khối cactch
- Nếu có ngoại lệ xảy ra và không bắt được: Chương trình bị gián đoạn
- Nếu có ngoại lệ xảy ra và bắt được: Chương trình thực thi các câu lệnh trong khối catch đầu tiên khớp với ngoại lệ, và thực thi tiếp các câu lệnh sau (các) khối cactch

Xử lý ngoại lệ với try - catch

Các ví dụ minh họa:

```
int[] a = { 1, 2, 3 };
System.out.println("No try-catch"); // thực thi
System.out.println(1/0); // exception xảy ra tại đây
System.out.println(a[10]); // không chạy vào
System.out.println("After exception occured"); // không chạy vào
// kết quả: chương trình gián đoạn
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không chạy vào
} catch (ArithmeticException e) {
    System.out.println("Exception"); // thực thi khối này
}
System.out.println("After try - catch"); // vẫn chạy bình thường
// kết quả?
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không chạy vào
} catch (Exception e) {
    System.out.println("Exception"); // thực thi khối này
}
System.out.println("After try - catch"); // vẫn chạy bình thường
// kết quả?
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không chạy vào
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception"); // bắt không đúng ngoại lệ
}
System.out.println("After try - catch"); // không chạy vào
// kết quả: chương trình bị gián đoạn
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không được thực thi
} catch (ArithmeticException e) {
    System.out.println("ArithmeticException"); // thực thi khối này
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("ArrayIndexOutOfBoundsException"); // bỏ qua
}
System.out.println("After try - catch"); // vẫn chạy bình thường
// kết quả?
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không được thực thi
} catch (ArithmeticException e) {
    System.out.println("ArrayIndexOutOfBoundsException"); // bỏ qua
} catch (Exception e) {
    System.out.println("Exception"); // thực thi khối này
}
System.out.println("After try - catch"); // vẫn chạy bình thường
// kết quả?
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không được thực thi
} catch (ArithmeticException e) {
    System.out.println("ArithmeticException"); // thực thi khối này
} catch (Exception e) {
    System.out.println("Exception"); // bỏ qua
}
System.out.println("After try - catch"); // vẫn chạy bình thường
// kết quả?
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không được thực thi
} catch (Exception e) {
    System.out.println("Exception"); // thực thi khối này
} catch (ArithmeticException e) {
    System.out.println("ArithmeticException"); // bỏ qua
}
System.out.println("After try - catch"); // vẫn chạy bình thường
// kết quả?
```

Xử lý ngoại lệ với try - catch

- Tại một thời điểm chỉ xảy ra một ngoại lệ

→ Chỉ có một khối catch được thực thi: Khi exception đã bị bắt ở một catch thì các catch tiếp theo bị bỏ qua

- Các khối catch phải được sắp xếp theo thứ tự từ exception chi tiết đến exception chung chung. VD: *NullPointerException* đến *RuntimeException* đến *Exception*...

Xử lý ngoại lệ với try - catch

- Khối **finally** được đặt sau khối try – catch: Luôn được thực thi không phụ thuộc vào việc ngoại lệ có xảy ra hay không

```
try {
    …
} catch (Exception e) {
    …
} finally {
    // khối lệnh luôn được thực thi
}
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không được thực thi
} catch (Exception e) {
    System.out.println("Exception"); // thực thi khối này
} finally {
    System.out.println("Finally block"); // luôn chạy vào
}
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không được thực thi
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception"); // bắt không đúng ngoại lệ
} finally {
    System.out.println("Finally block"); // luôn chạy vào
}
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không được thực thi
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception"); // bắt không đúng ngoại lệ
} finally {
    System.out.println("Finally block"); // luôn chạy vào
}
System.out.println("After try - catch"); // không chạy vào
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không được thực thi
} catch (Exception e) {
    System.out.println("Exception"); // thực thi khối này
} finally {
    System.out.println("Finally block"); // luôn chạy vào
}
System.out.println("After try - catch"); // vẫn chạy
```

Xử lý ngoại lệ với try - catch

```
int[] a = { 1, 2, 3 };
try {
    System.out.println("Try block");
    System.out.println(1/0); // exception xảy ra tại đây
    System.out.println(a[10]); // không được thực thi
} catch (Exception e) {
    return; // thực thi khối này
} finally {
    System.out.println(“Finally block"); // luôn chạy vào
}
System.out.println(“After try - catch"); // không chạy vào
```

Xử lý ngoại lệ với try - catch

- Khi sử dụng try bắt buộc phải có catch hoặc finally (hoặc cả hai)
- Một try có thể có nhiều catch nhưng chỉ có một finally
- Có thể sử dụng lệnh `System.exit(0)` trong catch để bỏ qua finally

```
try {
    … // xảy ra ngoại lệ
} catch (Exception e) {
    System.exit(0);
} finally {
    // khối lệnh không được thực thi
}
```

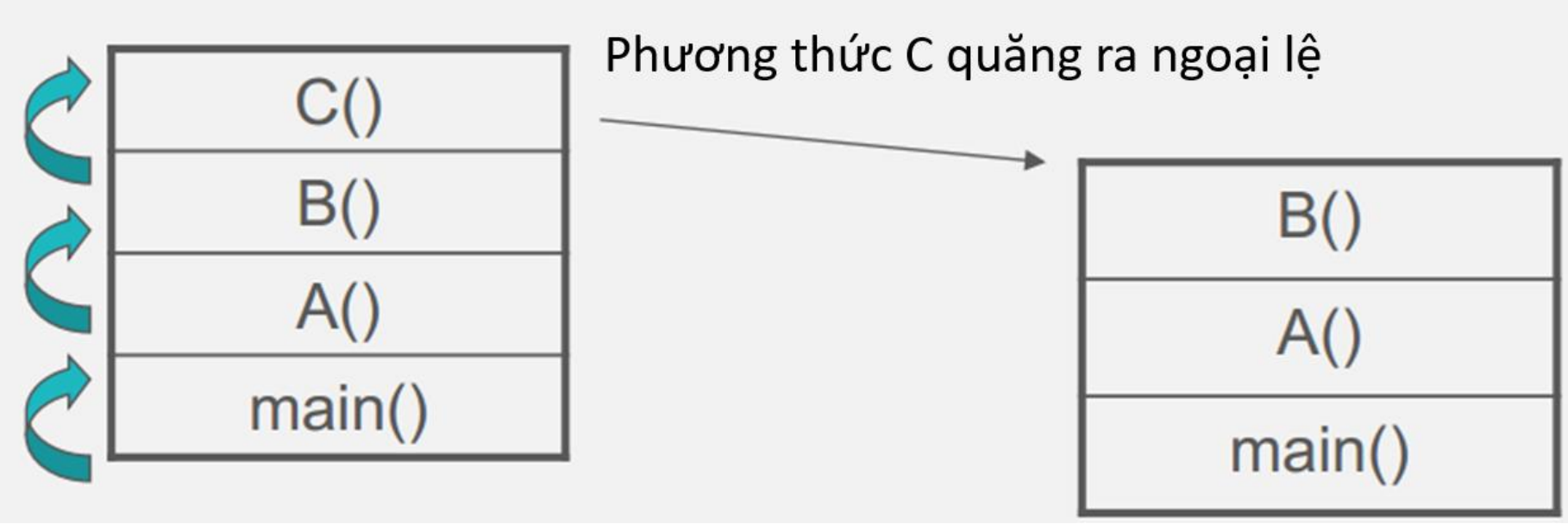
Xử lý ngoại lệ với throws

- Khi complier phát hiện một checked exception, lập trình viên sẽ được yêu cầu thực dùng **try – catch** để bắt ngoại lệ hoặc dùng **throws:**
  - Có thể khai báo nhiều exception sau phần tham số của phương thức

    `public static <functionType> <functionName>(<params>)` **`throws <ExceptionType1>,…,<ExceptionTypeN> { … }`**
  - Đẩy quyền xử lý ngoại lệ cho phương thức gọi nó
  - Phương thức gọi sẽ xử lý thay (try – catch) hoặc tiếp tục đẩy cho phương thức cấp cao hơn

Xử lý ngoại lệ với throws

- Nếu C không xử lý mà quăng ra ngoại lệ thì B phải xử lý, nếu B không xử lý thì mà tiếp tục quăng thì A phải xử lý…
- Nếu đến main mà vẫn không xử lý thì chương trình bị gián đoạn



Xử lý ngoại lệ với throws

```
public static void main(String [] args) throws FileNotFoundException
{
    File inputFile = new File(“in.txt");
    Scanner input = new Scanner(inputFile);
    while(input.hasNextLine()) {
        System.out.println(input.nextLine());
    }
}
```

Xử lý ngoại lệ với throws

```
public static void printFile() throws FileNotFoundException {
    File inputFile = new File("in.txt");
    Scanner input = new Scanner(inputFile);
    while(input.hasNextLine()) {
            System.out.println(input.nextLine());
    }
}

public static void main(String[] args) {
    try {
        printFile();
    } catch (FileNotFoundException ex) {
        System.out.println(ex);
    }
}
```

Xử lý ngoại lệ với throws

- Try – catch hay throws?
  - Nếu trong catch chỉ in ra lỗi sau đó thoát khỏi chương trình (không thực hiện thêm gì) thì chỉ cần dùng throws là đủ
  - Nếu trong catch có các bước xử lý lỗi để không ảnh hưởng đến việc thực thi tiếp chương trình thì nên dùng try - catch

Tùy chỉnh ngoại lệ

- Có thể tùy chỉnh ngoại lệ với từ khóa **throw** (khác với throws)
  - Quăng ra những ngoại lệ người dùng tự định nghĩa
  - Kế thừa từ lớp Exception hoặc một trong những lớp con của Exception
    - Với unchecked exception: Kế thừa từ lớp RuntimeException hoặc một trong những lớp con của RuntimeException
  - Khai báo trong phương thức cần xử lý và chỉ ra một ngoại lệ rõ ràng
  - Đẩy quyền xử lý ngoại lệ cho phương thức gọi nó

Tùy chỉnh ngoại lệ

```
public static void checkAge(int age) {
    if (age < 18) {
        throw new ArithmeticException("Access denied!");
    } else {
        System.out.println("Access granted.");
    }
}

public static void main(String[] args) {
    checkAge(13);
}
```

Tùy chỉnh ngoại lệ

```
public static class InvalidAgeException extends ArithmeticException {
    public InvalidAgeException(String message) {
        super(message);
    }
}
public static void checkAge(int age) {
    if (age < 18) {
        throw new InvalidAgeException("Access denied!");
    } else {
        System.out.println("Access granted.");
    }
}
```

Tài liệu tham khảo

- Liang, Y. Daniel, Introduction to Java programming and Data Structures, 11th edition, Pearson Prentice Hall, 2019
- N. M. Sơn, Bài giảng Lập trình hướng đối tượng, HVCNBCVT, 2020
- N. M. Sơn, Slide giảng dạy môn Lập trình hướng đối tượng
- T. T. V. Anh, Slide giảng dạy môn Lập trình hướng đối tượng