Ministry of SMEs and Startups × KOSME Korea SMEs and Startups Agency × LIKELION

# Stream API, Base64, forEach

Nguyễn Anh Tuấn


K TECH COLLEGE

# Nội dung bài giảng

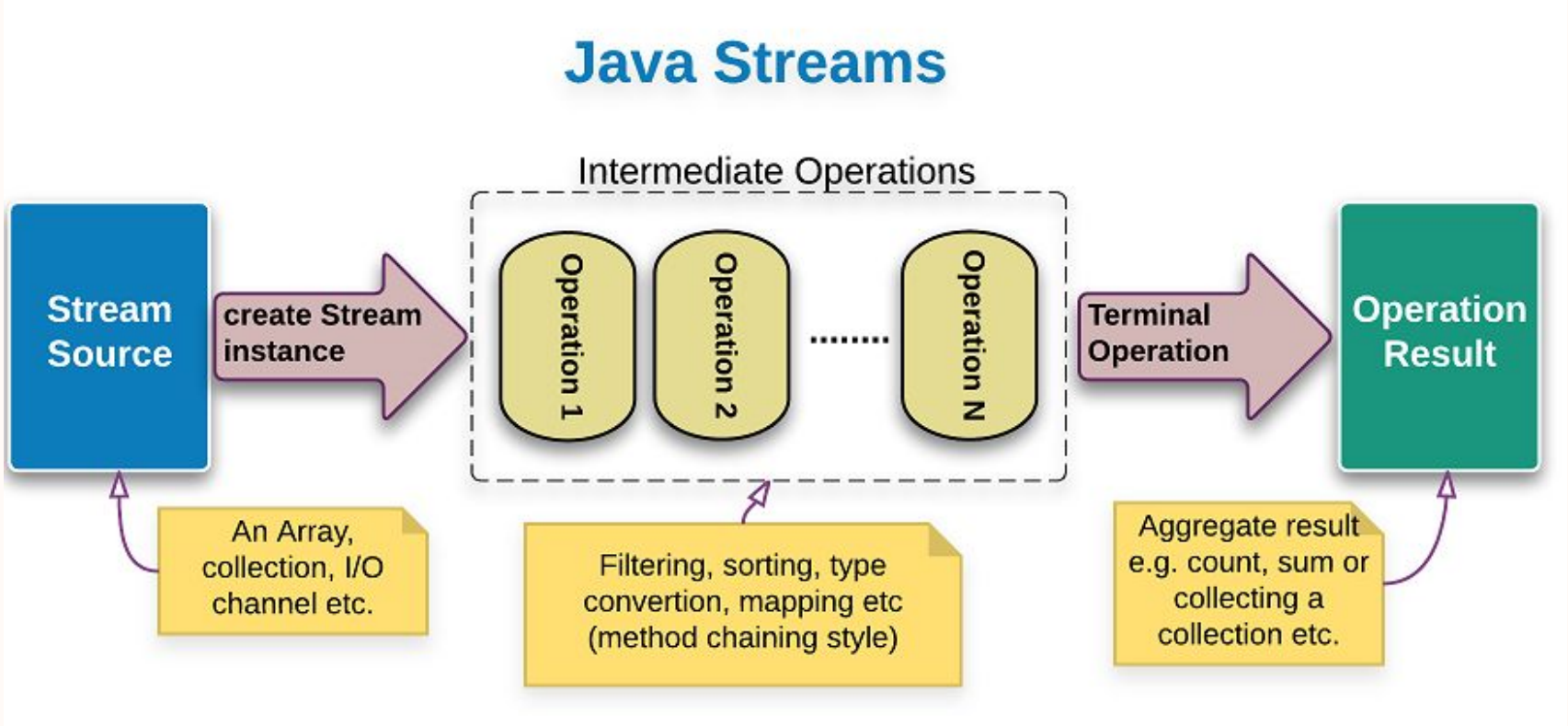
1 Stream API

2 Base64, encoding, decoding

3 Phương thức forEach

# Stream API

# Stream API

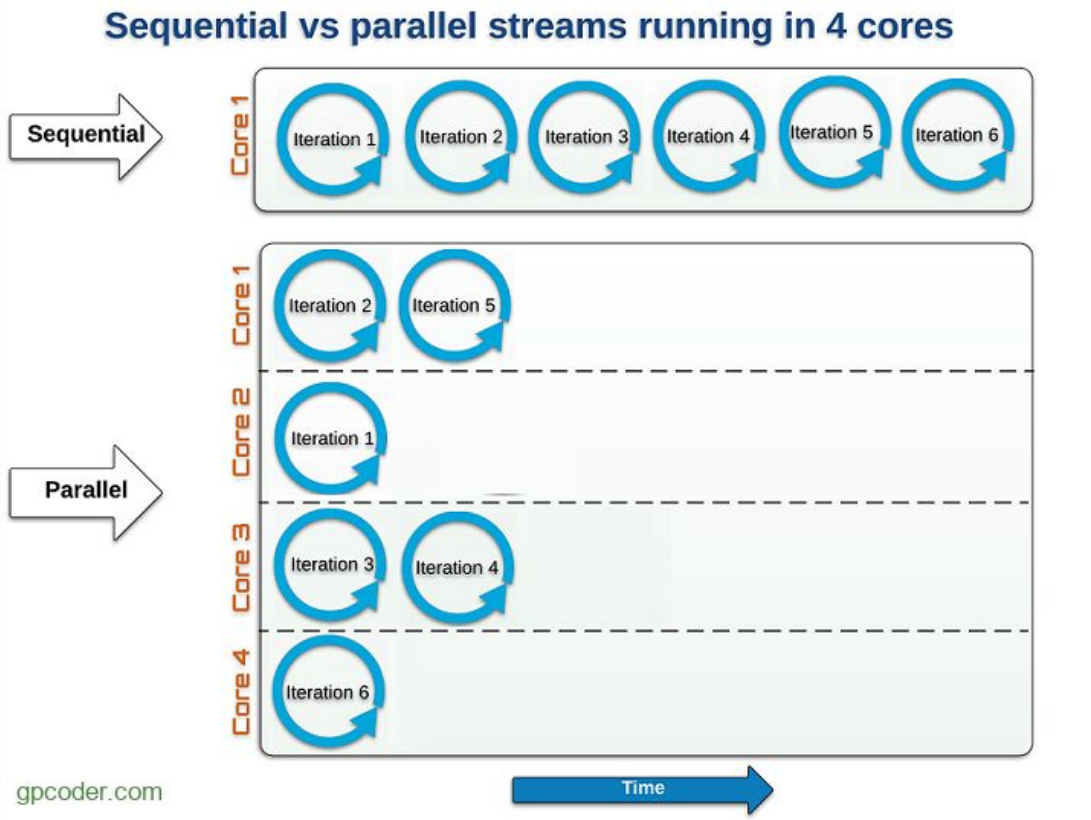

Java Streams
Intermediate Operations
Stream Source
create Stream instance
Operation 1
Operation 2
.....
Operation N
Terminal Operation
Operation Result
An Array, collection, I/O channel etc.
Filtering, sorting, type convertion, mapping etc (method chaining style)
Aggregate result e.g. count, sum or collecting a collection etc.

# Stream API

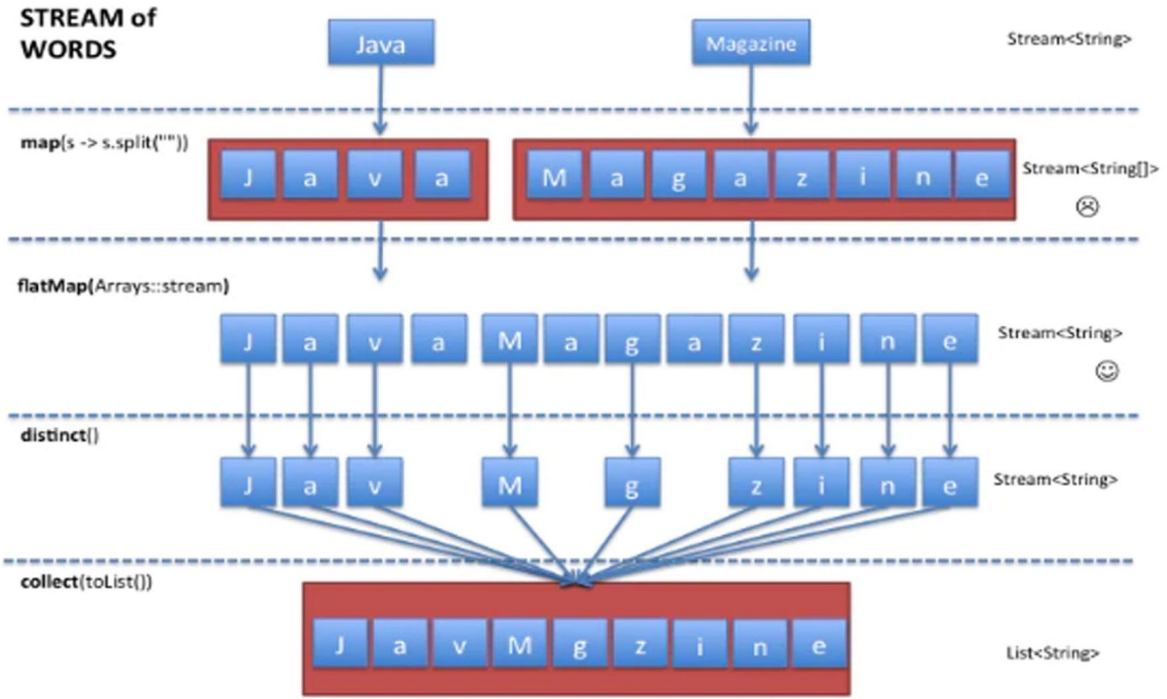
- **Định nghĩa**:
  - **Stream** (luồng) giúp cho việc thao tác trên collection và array trở nên dễ dàng và tối ưu hơn
  - Một Stream đại diện cho một chuỗi các phần tử hỗ trợ các hoạt động tổng hợp tuần tự (**sequential**) và song song (**parallel**)

- **Lưu ý**:
  - Stream không phải là một cấu trúc dữ liệu (data structure). Stream không lưu trữ các phần tử của collection hay array.
  - Các hoạt động được thực hiện trên Collection, Array hoặc bất kỳ nguồn dữ liệu nào khác đều không làm thay đổi dữ liệu của nguồn mà chỉ trả lại stream mới
  - Stream không dùng lại được, nghĩa là một khi đã sử dụng nó xong, chúng ta không thể gọi nó lại để sử dụng lần nữa.
  - Các phần tử của luồng chỉ được truy cập một lần trong suốt vòng đời của Stream.

# Stream API


Sequential vs parallel streams running in 4 cores
Sequential
Core 1
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Parallel
Core 1
Iteration 2
Iteration 5
Core 2
Iteration 1
Core 3
Iteration 3
Iteration 4
Core 4
Iteration 6
gpcoder.com
Time

# Java Streams




# The Code City

# Khởi tạo stream tuần tự

- **Từ collection**

```
List<Integer> list = Arrays.asList(1,2,3,4,5);
Stream<Integer> filterList = list.stream();
```

- **Từ array**

```
String[] myArray = {"a", "b", "c", "d"};
Stream<String> stream = Arrays.stream(myArray);
```

- **Từ I/O**

```
Stream<String> lines = Files.lines(Paths.get("file.txt"));
```

# Khởi tạo stream song song

- **Từ collection**

```
List<Integer> list = Arrays.asList(1,2,3,4,5);
Stream<Integer> filterList = list.parallelStream();
```

- **Từ array**

```
String[] myArray = {"a", "b", "c", "d"};
Stream<String> stream = Arrays.stream(myArray).parallel();
```

- **Từ I/O**

```
Stream<String> lines = Files.lines(Paths.get("file.txt")).parallel();
```

# Các thao tác trung gian (Intermediate Operations)

- **filter: Lọc các phần tử thỏa mãn điều kiện cho trước**

```
// Khởi tạo 1 danh sách bao gồm tên các ngôn ngữ lập trình
List<String> list = Arrays.asList("ReactJS", "Java", "PHP", "Angular", "VueS", "Python");

// Lọc các phần tử trong danh sách trên, phần thử nào bắt đầu với chữ cái A thì tập hợp lại
List<String> result = list.stream()
    .filter(s -> s.startsWith("A"))
    .collect(Collectors.toList());

// Kết quả trả về
[Angular]
```

# Các thao tác trung gian (Intermediate Operations)

- **map: Ánh xạ các phần tử thành phần tử mới**

```
// Khởi tạo 1 danh sách bao gồm tên các ngôn ngữ lập trình
List<String> list = Arrays.asList("ReactJS", "Java", "PHP", "Angular", "VueS", "Python");

// Lọc các phần tử trong danh sách trên, phần thử nào bắt đầu với chữ cái A thì tập hợp lại
List<String> result = list.stream()
        .map(s -> s.substring(0,3))
        .collect(Collectors.toList());

// Kết quả trả về
[Rea, Jav, PHP, Ang, Vue, Pyt]
```

# Các thao tác trung gian (Intermediate Operations)

- **sorted: Sắp xếp các phần tử theo thứ tự tự nhiên**

```
// Khởi tạo 1 danh sách bao gồm tên các ngôn ngữ lập trình
List<String> list = Arrays.asList("ReactJS", "Java", "PHP", "Angular", "VueS", "Python");

// Lọc các phần tử trong danh sách trên, phần thử nào bắt đầu với chữ cái A thì tập hợp lại
List<String> result = list.stream()
        .sorted()
        .collect(Collectors.toList());

// Kết quả trả về
[Angular, Java, PHP, Python, ReactJS, VueS]
```

# Các thao tác kết thúc (Terminal Operations)

- **forEach: Duyệt các phần tử và thực hiện một hành động khác**

```
// Khởi tạo 1 danh sách bao gồm tên các ngôn ngữ lập trình
List<String> list = Arrays.asList("ReactJS", "Java", "PHP", "Angular", "VueS", "Python");

// Lọc các phần tử trong danh sách trên, phần thử nào bắt đầu với chữ cái A thì tập hợp lại
list.stream().forEach(s -> System.out.println(s + " - " + s.contains("J")));

// Kết quả trả về
ReactJS - true
Java - true
PHP - false
Angular - false
VueS - false
Python - false
```

# Các thao tác kết thúc (Terminal Operations)

- **collect: Tập hợp các phần tử stream vào một collection**

```
// Khởi tạo 1 danh sách bao gồm tên các ngôn ngữ lập trình
List<String> list = Arrays.asList("ReactJS", "Java", "PHP", "Angular", "VueS", "Python");

// Lọc các phần tử trong danh sách trên, phần thử nào bắt đầu với chữ cái A thì tập hợp lại
List<String> result = list.stream()
    .collect(Collectors.toList());

// Kết quả trả về
[ReactJS, Java, PHP, Angular, VueS, Python]
```

# Các thao tác kết thúc (Terminal Operations)

- **reduce: Gộp các phần tử lại để tạo ra một kết quả duy nhất**

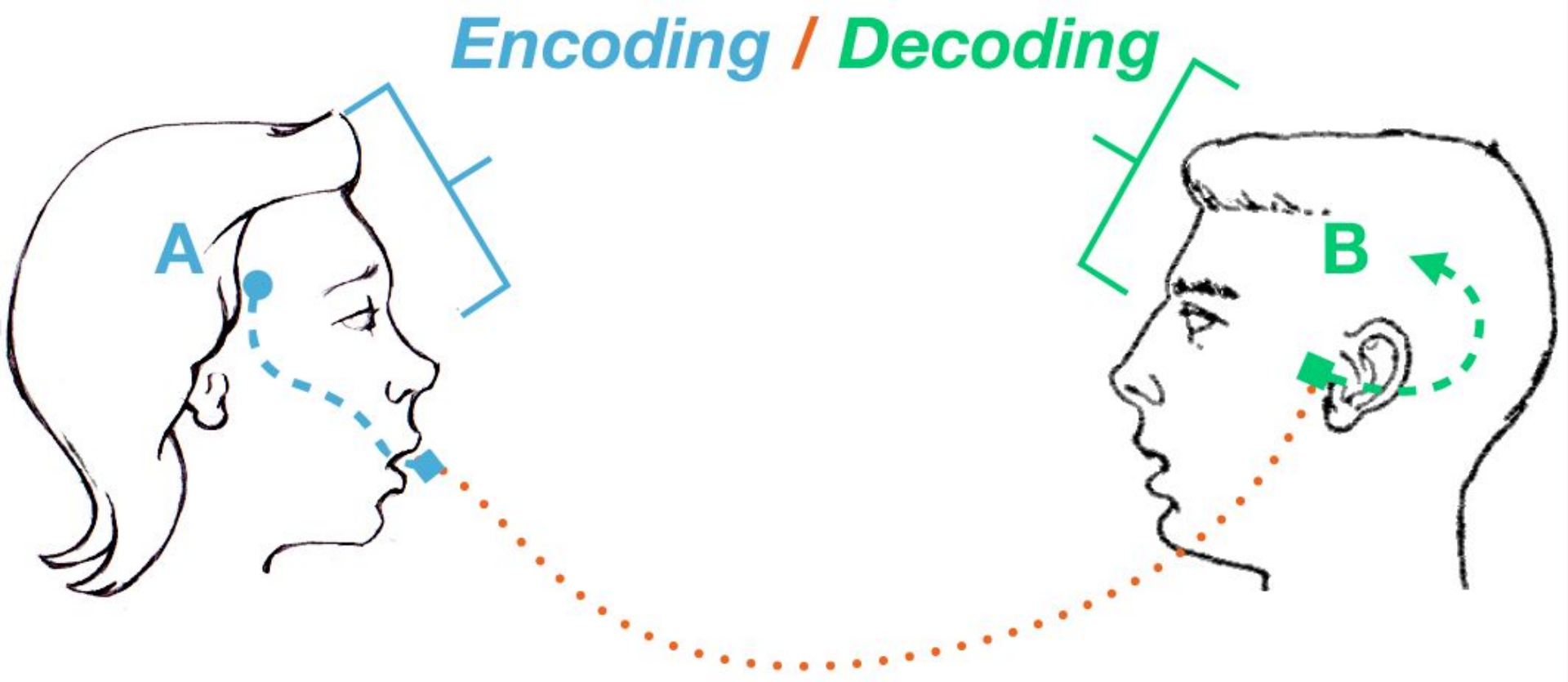
```
// Khởi tạo 1 danh sách bao gồm tên các ngôn ngữ lập trình
List<String> list = Arrays.asList("ReactJS", "Java", "PHP", "Angular", "VueS", "Python");

// Lọc các phần tử trong danh sách trên, phần thử nào bắt đầu với chữ cái A thì tập hợp lại
List<String> result = list.stream()
    .collect(Collectors.toList());

// Kết quả trả về
[ReactJS, Java, PHP, Angular, VueS, Python]
```

# Base64, encoding, decoding

- **Định nghĩa**:
  - **Encoding** là quá trình tạo ra dữ liệu phục vụ cho việc lưu trữ.
  - **Decoding** là quá trình chuyển đổi dữ liệu lưu trữ thành dữ liệu sử dụng trong ứng dụng.
  - **Base64** là một chương trình mã hóa chuỗi ký tự bằng cách dùng thay thế các ký tự trong bảng mã ASCII 8 bit thông dụng thành bảng mã 6 bit. Ký tự 64 trong Base64 là đại diện cho 64 ký tự (A-Za-z0-9+/) trong bảng mã ASCII.
- **Như vậy**:
  - **Base64 encoding** dùng để mã hóa một mảng byte hoặc String.
  - **Base64 encoding** dùng để giải mã một mảng byte hoặc String được mã hóa base64.
  - Có thể sử dụng **base64 encoding** và **base64 decoding** trong gói **java.util.Base64**

- **Tham khảo**:
  - https://vi.wikipedia.org/wiki/Base64

# Encoding, decoding



# Bảng mã Base64

| Valor | Caractere | Valor | Caractere | Valor | Caractere | Valor | Caractere |
|---|---|---|---|---|---|---|---|
| 0 | A | 16 | Q | 32 | g | 48 | w |
| 1 | B | 17 | R | 33 | h | 49 | x |
| 2 | C | 18 | S | 34 | i | 50 | y |
| 3 | D | 19 | T | 35 | j | 51 | z |
| 4 | E | 20 | U | 36 | k | 52 | 0 |
| 5 | F | 21 | V | 37 | l | 53 | 1 |
| 6 | G | 22 | W | 38 | m | 54 | 2 |
| 7 | H | 23 | X | 39 | n | 55 | 3 |
| 8 | I | 24 | Y | 40 | o | 56 | 4 |
| 9 | J | 25 | Z | 41 | p | 57 | 5 |
| 10 | K | 26 | a | 42 | q | 58 | 6 |
| 11 | L | 27 | b | 43 | r | 59 | 7 |
| 12 | M | 28 | c | 44 | s | 60 | 8 |
| 13 | N | 29 | d | 45 | t | 61 | 9 |
| 14 | O | 30 | e | 46 | u | 62 | + |
| 15 | P | 31 | f | 47 | v | 63 | / |

# Mã hoá Base64 Basic

- **Lưu trữ dữ liệu nhị phân dưới dạng chuỗi ký tự trong cơ sở dữ liệu hoặc tệp tin văn bản.**

```
// Mã hoá và giải mã chuỗi mật khẩu
String password = "Likelion2024";

// Mã hoá với base64 basic
String encodedString = Base64.getEncoder().encodeToString(password.getBytes());
System.out.println(encodedString);
// Output: TGlrZWxpb24yMDI0

// Giải mã với base64 basic
byte[] decodedBytes = Base64.getDecoder().decode(encodedString);
String decodedString = new String(decodedBytes);
System.out.println(decodedString);
// Output: Likelion2024
```

# Mã hoá Base64 URL-safe

- **Mã hóa dữ liệu nhị phân thành các chuỗi ký tự có thể được sử dụng an toàn trong URL, mà không cần phải mã hóa thêm các ký tự đặc biệt.**

```
// Mã hoá và giải mã chuỗi url
String url = "https://www.likelion.edu.vn/";

// Mã hoá với Base64 URL-safe
String encodedString = Base64.getUrlEncoder().encodeToString(url.getBytes());
System.out.println(encodedString);
// Output: aHR0cHM6Ly93d3cubGlrZWxpb24uZWR1LnZuLw==

// Giải mã với Base64 URL-safe
byte[] decodedBytes = Base64.getUrlDecoder().decode(encodedString);
String decodedString = new String(decodedBytes);
System.out.println(decodedString);
// Output: https://www.likelion.edu.vn/
```

# Mã hoá Base64 MIME

- **Được sử dụng trong email để mã hóa các tệp đính kèm và nội dung nhị phân, đảm bảo rằng dữ liệu có thể được truyền qua các hệ thống chỉ hỗ trợ văn bản.**

```
// Mã hoá và giải mã chuỗi …
String mime = "...";

// Mã hoá với Base64 Mime
String encodedString = Base64.getMimeEncoder().encodeToString(url.getBytes());
System.out.println(encodedString);

// Giải mã với Base64 Mime
byte[] decodedBytes = Base64.getMimeDecoder().decode(encodedString);
String decodedString = new String(decodedBytes);
System.out.println(decodedString);
```

# Chuỗi mime được mã hoá Base64 Mime ở slide trước

Ci0tLS0tLS0tLS0tLS0tLS0tLS0tRWFzdGVyIGVnZy0tLS0tLS0tLS0tLS0tLS0tLS0t
ClRyb25nIG3DoyBow7NhIEJhc2U2NCwgxJHhu5kgZMOgaSBj4bunYSBjaHXhu5dpIMSRxrDhu6Nj
IG3DoyBow7NhIMSR4bqndSByYSBwaOG6o2kgbMOgIGLhu5lpIHPhu5EgY+G7p2EgMy4KTuG6v3Ug
a2jDtG5nIMSR4bunLCDEkeG6p3UgcmEgc+G6vSDEkcaw4bujYyDEkeG7h20gYuG6sW5nIGPDoWMg
a8O9IHThu7EgcGFkIGLhu5Ugc3VuZyAoZOG6pXUgPSkuClRhIGPDsyB0aOG7gyBi4buPIGThuqV1
ID0gYuG6sW5nIGPDoWNoIHPhu60gZOG7pW5nIHdpdGhvdXRQYWRkaW5nIG5oxrAgc2F1OiBCYXNl
NjQuZ2V0RW5jb2RlcigpLndpdGhvdXRQYWRkaW5nKCkuZW5jb2RlVG9TdHJpbmcoc3RyLmdldEJ5
dGVzKCkp

Trong mã hóa Base64, độ dài của chuỗi được mã hóa đầu ra phải là bội số của 3.
Nếu không đủ, đầu ra sẽ được đệm bằng các ký tự pad bổ sung (dấu =).
Ta có thể bỏ dấu = bằng cách sử dụng withoutPadding như sau: Base64.getEncoder().withoutPadding().encodeToString(str.getBytes())

# Phương thức forEach

# Sử dụng forEach() với List

```
public static void main(String[] args) {
        List<String> languages = Arrays.asList("Java", "C#", "C++", "PHP", "Javascript");

        // Sử dụng forEach với Lambda Expression
        languages.forEach(lang -> System.out.println(lang));

        // Sử dụng forEach với Method Reference
        languages.forEach(System.out::println);
}
```

# Sử dụng forEach() với Map

```
public static void main(String[] args) {
        Map<Integer, String> hmap = new HashMap<Integer, String>();
        hmap.put(1, "Java");
        hmap.put(2, "Javascript");
        hmap.put(3, "PHP");
        hmap.put(4, "C#");
        hmap.put(5, "C++");

        // Sử dụng forEach với Lambda Expression
        hmap.forEach((key, value) -> System.out.println(key + " - " + value));

        // Sử dụng forEach với Method Reference
        hmap.forEach(System.out::println);
}
```

# Sử dụng forEach() với forEachOrdered()

```
public static void main(String[] args) {
        List<String> languages = Arrays.asList("Java", "C#", "C++", "PHP", "Javascript");

        // Sử dụng forEach với Lambda Expression
        languages.stream().forEachOrdered(lang -> System.out.println(lang));

        // Sử dụng forEach với Method Reference
        languages.stream().forEachOrdered(System.out::println);


        // Giải thích thêm về forEachOrdered
        //  forEach: Không đảm bảo thứ tự duyệt mảng/danh sách khi sử dụng parallel stream
        // forEachOrdered: Đảm bảo thứ tự duyệt mảng/danh sách, ngay cả khi sử dụng parallel stream

}
```

# ROAD TO KOREA

Nếu có bất kỳ thắc mắc nào, hãy đặt câu hỏi qua

**mail@mail.com hoặc Zalo 0xxx xxx xxx**