Bài 06:
Ngoại lệ (Exceptions) trong java

Giảng Viên: Giang Hào Côn

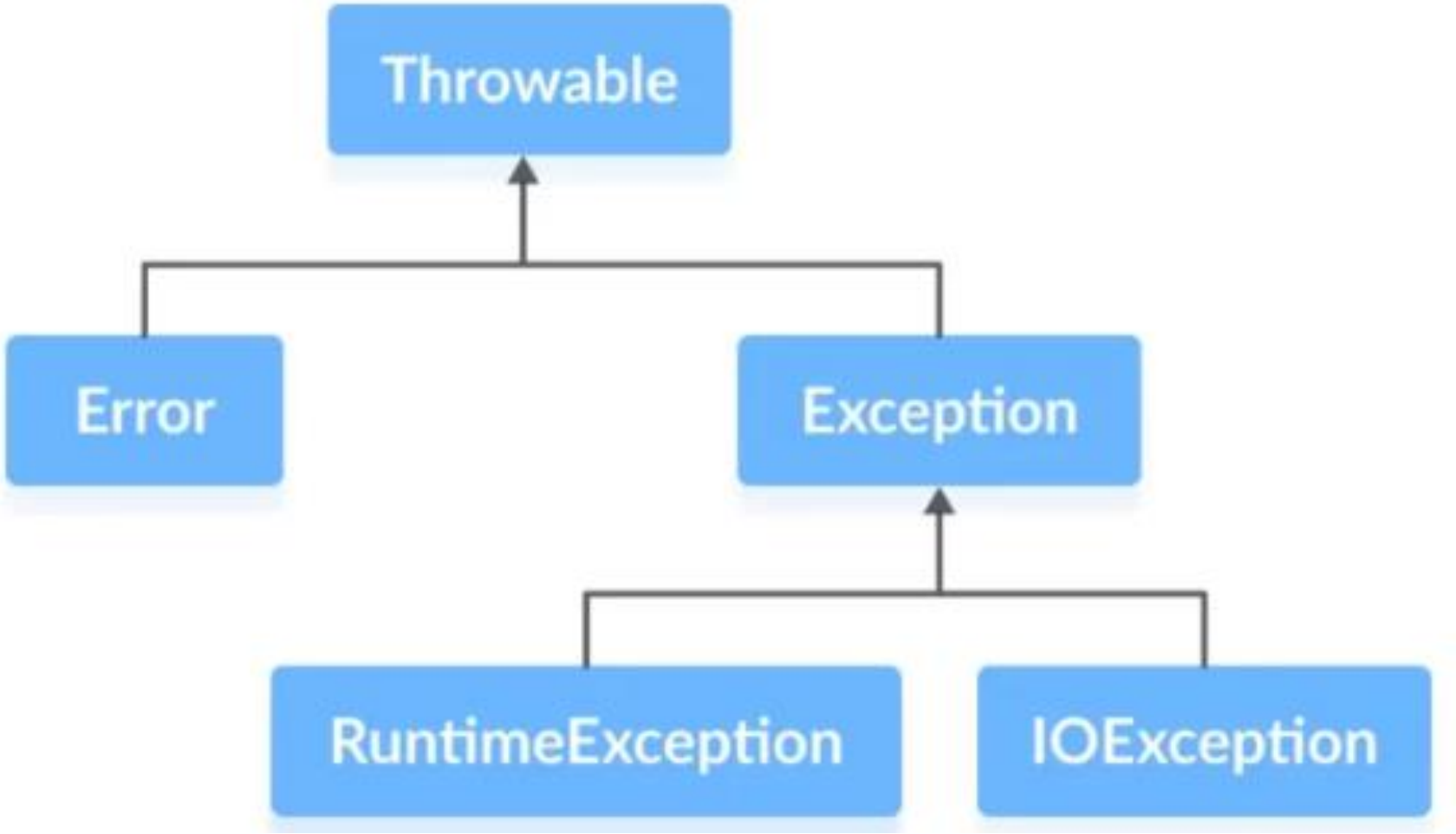
6.1/ Ngoại lệ (exceptions) trong Java là gì?

Một ngoại lệ (exception) là một sự kiện bất ngờ xảy ra trong quá trình thực thi chương trình. Nó có thể khiến chương trình kết thúc bất thường. Có nhiều nguyên nhân gây ra ngoại lệ như:

- Người dùng nhập dữ liệu không hợp lệ
- Lỗi thiết bị
- Mất kết nối mạng
- Giới hạn về phần cứng, có thể là hết bộ nhớ
- Lỗi code
- Mở file không hợp lệ

6.1/ Ngoại lệ (exceptions) trong Java là gì?

Hệ thống phân cấp ngoại lệ trong Java.



6.1/ Ngoại lệ (exceptions) trong Java là gì?

- **Ví dụ một vài error được xuất ra màn hình của lớp Error**

Error: LinkageError occurred while loading main class ClassName
java.lang.UnsupportedClassVersionError: ClassName has been compiled by a more recent version of the Java Runtime (class file version 55.0), this version of the Java Runtime only recognizes class file versions up to 54.0
java.lang.annotation.AnnotationFormatError: Invalid default: public abstract java.lang.Class org.springframework.data.cassandra.repository.config.EnableCassandraRepositories.repositoryBaseClass() java.lang.OutOfMemoryError: Java heap space Exception in thread "main" java.lang.StackOverflowError

6.1/ Ngoại lệ (exceptions) trong Java là gì?

- **Ví dụ một vài error được xuất ra màn hình của RuntimeException**

```
java.lang.IllegalArgumentException: adding container's parent to itself
java.lang.NullPointerException
at twoten.TwoTenB.&lt;init&gt;(TwoTenB.java:29)
at javapractice.JavaPractice.main(JavaPractice.java:32)
Java Result: 1
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

6.1/ Ngoại lệ (exceptions) trong Java là gì?

- **Ví dụ một vài error được xuất ra màn hình của IOException**

```
Exception in thread "main" java.io.FileNotFoundException: file.txt
(The system cannot find the file specified)
Exception in thread "main" java.io.IOException: Cannot run
program "": error=2, No such a file or directory.
```

6.2/ Xử lý Ngoại lệ (exceptions)

Ngoại lệ (exceptions) có thể làm chương trình kết thúc bất thường. Do đó, xử lý ngoại lệ là công việc quan trọng mà lập trình việc phải lưu ý. Chúng ta có thể xử lý ngoại lệ với những cách sau trong Java:

- Sử dụng try…catch
- Sử dụng try…catch…finally
- Sử dụng throw và throws

6.2/ Xử lý Ngoại lệ (exceptions)

1. Xử lý exceptions trong Java với try...catch

Cú pháp:

```
try {
    // code
}
catch(Exception e) {
    // code
}
```

Ví dụ:

```
class Main {
    public static void main(String[] args) {
        try {
            //các lệnh có thể sinh ra exception
            int divideByZero = 5 / 0;
            System.out.println("Rest of code in try block");
        }catch (ArithmeticException e) {
            System.out.println("ArithmeticException => " + e.getMessage());
        }
    }
}
```

Kết quả

```
ArithmeticException => / by zero
```

6.2/ Xử lý Ngoại lệ (exceptions)

1. Xử lý exceptions trong Java với try...catch

Ví dụ: Có thể bắt nhiều exception với một khối lệnh catch

```
class Main {
    public static void main(String[] args) {
        try {
            int array[] = new int[10];
            array[10] = 30 / 0;
        } catch (ArithmeticException | ArrayIndexOutOfBoundsException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

6.2/ Xử lý Ngoại lệ (exceptions)

1. Xử lý exceptions trong Java với try...catch

Ví dụ: Có thể sử dụng nhiều khối lệnh catch với try

```
class ListOfNumbers {
    public int[] arr = new int[10];
    public void writeList() {
        try {
            arr[10] = 11;
        }
        catch (NumberFormatException e1) {
            System.out.println("NumberFormatException => " + e1.getMessage());
        }
        catch (IndexOutOfBoundsException e2) {
            System.out.println("IndexOutOfBoundsException => " + e2.getMessage());
        }
    }
}
```

6.2/ Xử lý Ngoại lệ (exceptions)

2. Xử lý exceptions trong Java với try...catch...finally

Cú pháp:

```
try {
    //code
}
catch (ExceptionType1 e1) {
    // catch block
}
finally {
    // finally block always executes
}
```

Ví dụ :

```
import java.io.FileReader;
import java.io.IOException;
class Main {
    public static void main(String args[]) {
        FileReader fileReader = null;
        try {
            fileReader = new FileReader("D:\\test.txt");
            // do something
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            try {
                if(fileReader != null) {
                    fileReader.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

6.2/ Xử lý Ngoại lệ (exceptions)

3. Khối lệnh try...finally trong Java

Trong Java, chúng ta có thể sử dụng **finally** với **try** mà không cần **catch**. Không khuyến kích sử dụng như thế này bởi **exception** sẽ không được xử lý bởi **catch** nếu xảy ra.

```
class Main {
    public static void main(String[] args) {
        try {
            int divideByZero = 5 / 0;
        }
        finally {
            System.out.println("Finally block is always executed");
        }
    }
}
```

Kết quả

```
Finally block is always executed
Exception in thread "main" java.lang.ArithmeticException: / by zero
at anotherPackage.Main.main(Main.java:6)
```

6.2/ Xử lý Ngoại lệ (exceptions)

3. Xử lý exceptions trong Java với throws và throw

3.1. Từ khóa throws trong Java

Trong Java, chúng ta sử dụng từ khóa throws trong khai báo phương thức để khai báo những ngoại lệ (exceptions) có thể xảy ra trong phương thức đó. Chúng ta có thể nói là throws ném các ngoại lệ của phương thức ra bên ngoài.

Lưu ý: Lập trình viên phải đoán biết trước những loại ngoại lệ của phương thức để khai báo cho chính xác. Cú pháp:

```
accessModifier returnType methodName() throws ExceptionType1, ExceptionType2 … {
  // code
}
```

6.2/ Xử lý Ngoại lệ (exceptions)

3. Xử lý exceptions trong Java với throws và throw

3.1. Từ khóa throws trong Java

Ví dụ

```
import java.io.*;
class Main {
    public static void findFile() throws FileNotFoundException {
        //code that may produce IOException
        File newFile=new File("test.txt");
        FileInputStream stream=new FileInputStream(newFile);
    }

    public static void main(String[] args) {
        try{
            findFile();
        } catch(IOException e){
            System.out.println(e);
        }
    }
}
```

???

Kết quả

```
java.io.FileNotFoundException: test.txt (The system cannot find the file specified)
```

3. Xử lý exceptions trong Java với throws và throw

3.2. Từ khóa throw trong Java

```java
import java.io.*;
class Main {
    public static void findFile() throws IOException {
        //code that may produce IOException
        File newFile=new File("test.txt");
        if(newFile.exists() && !newFile.isDirectory()) {
            // do something
        }else{
            throw new IOException("File not found");
        }
    }

    public static void main(String[] args) {
        try{
            findFile();
        } catch(IOException e){
            System.out.println(e);
        }
    }
}
```

Kết quả

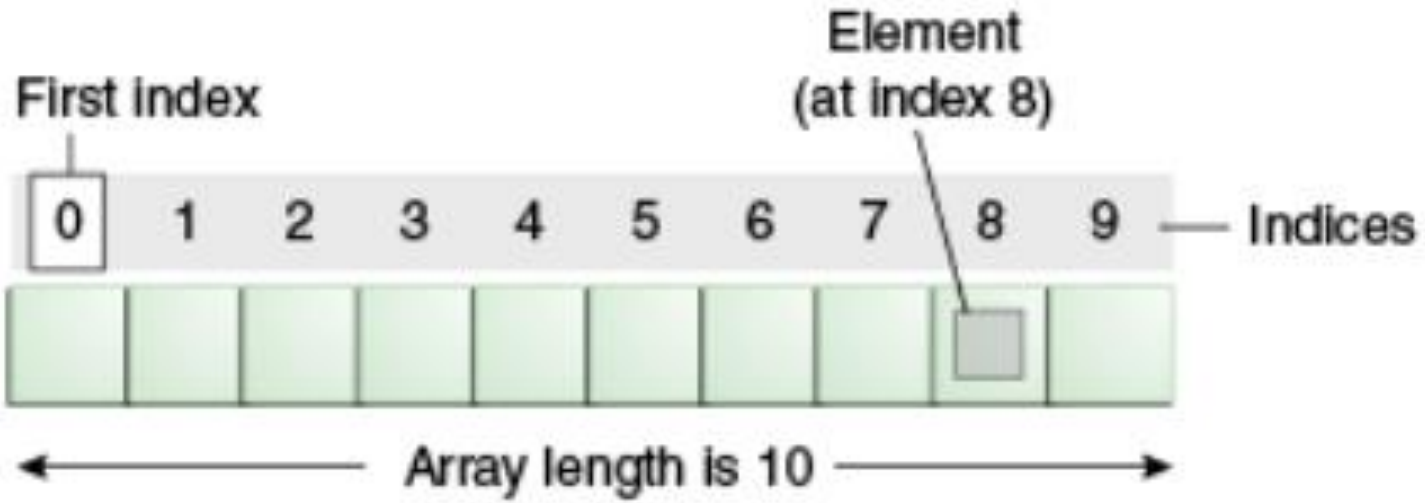
```
java.io.IOException: File not found
```

???

6.3/ Khái niệm mảng

Đặt vấn đề: Nhập 100 số nguyên, tính giá trị trung bình của chúng và cho biết có bao nhiêu số lớn hơn giá trị trung bình? Không khả thi nếu khai báo 100 biến. Sử dụng một cấu trúc dữ liệu gọi là mảng (array) để lưu 100 số nguyên đó.

Minh họa mảng:



Mảng là gì ?

6.3.1.Khai báo và khởi tạo mảng một chiều

Cú pháp khai báo mảng một chiều:

```
elementType[] arrayRefVar;
hoặc:
elementType arrayRefVar[];
```

Trong đó, elementType có thể là kiểu dữ liệu nguyên thủy như int, char, double, byte,… hoặc các lớp (class) trong Java. arrayRefVar là tên mảng, có quy ước đặt tên giống tên biến.

6.3.1.Khai báo và khởi tạo mảng một chiều

Ví dụ: khai báo một biến mảng **myList** lưu trữ các phần tử **double**:

```
double[] myList;
   //hoặc
double myList[];

String[] array;
```

Nhưng có bao nhiêu phần tử trong mảng?

6.3.1.Khai báo và khởi tạo mảng một chiều

Để khai báo số phần tử có trong mảng, chúng ta cần phải cấp phát vùng nhớ cho mảng tương ứng với số phần tử mà có thể lưu trữ. Cú pháp:

```
arrayRefVar = new elementType[arraySize];
```

Sử dụng từ khóa new để cấp phát vùng nhớ cho mảng. Số vùng nhớ được cấp phát là arraySize, mỗi vùng nhớ có thể lưu trữ một giá trị kiểu elementType. Sau đó, gán mảng vừa tạo vào biến tham chiếu arrayRefVar. Ví dụ:

```
myList = new double[10];
```

6.3.1.Khai báo và khởi tạo mảng một chiều

Có thể khai báo và cấp phát vùng nhớ cho mảng trong một câu lệnh.

Cú pháp:

```
elementType[] arrayRefVar = new elementType[arraySize];
hoặc
elementType arrayRefVar[] = new elementType[arraySize];
```

Ví dụ

```
double[] myList = new double[10];

double myList[] = new double[10];

String[] array = new String[100];
```

Khởi tạo mảng một chiều là gì ?

6.3.1.Khai báo và khởi tạo mảng một chiều

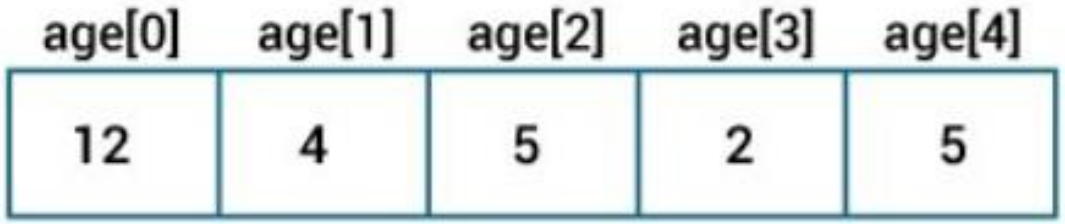
Ví dụ 01:

```
//khai báo và khởi tạo mảng
int[] age = {12, 4, 5, 2, 5};
```

Ví dụ 02:

```
//khai báo một mảng
int[] age = new int[5];

//khởi tạo mảng
age[0] = 12;
age[1] = 4;
age[2] = 5;
```



6.3.2. Kích thước và giá trị mặc định

Khi một mảng được cấp phát, số lượng phần tử mảng là cố định và không thể thay đổi. Để lấy kích thước mảng, chúng ta dùng **arrayRefVar.lenght**.

Ví dụ: myList.lenght = 10;.

Khi một mảng được tạo, giá trị các phần tử mảng được gán một giá trị mặc định:

- 0 nếu kiểu số,
- \u0000 nếu kiểu char
- false nếu kiểu boolean.

6.3.3. Truy cập các phần tử trong mảng

- Các phần tử của mảng được truy cập thông qua chỉ mục (index). Bắt đầu là 0, nghĩa là từ phần tử thứ 0 đến arrayRefVar.length-1.
- Ví dụ 01: myList có 10 phần tử kiểu double có chỉ số từ 0 đến 9.
- Cú pháp:

```
arrayRefVar[index];
```

- Ví dụ 02:

```
double[] myList = {2.0, 2.1, 0.0, 1.0, 3.2, 4.5, 9.9, 5.7, 5.8, 1.2};
System.out.println(myList[0]);
```

6.3.4/ Kỹ thuật lập trình mảng 1 chiều trong Java

1. Nhập xuất mảng một chiều

```
import java.util.Scanner;

public class ViDuMang {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        double[] myList = new double[5];

        System.out.println("Nhập " + myList.length + " của mảng:");
        for (int i = 0; i < myList.length; i++) {
                myList[i] = input.nextDouble();
        }
        System.out.print("Các phần tử mảng: ");
        for (int i = 0; i < myList.length; i++) {
                System.out.print(myList[i] + " ");
        }
    }
}
```

Kết quả

```
Nhap 5 cua mang:
3.2
5.6
8.7
9.9
2.1
Cac phan tu trong mang: 3.2 5.6 8.7 9.9 2.1
```

6.3.4/ Kỹ thuật lập trình mảng 1 chiều trong Java

2. Khởi tạo mảng với giá trị ngẫu nhiên

```
public class ViDuMang {
    public static void main(String[] args) {

        double[] myList = new double[5];

        for (int i = 0; i < myList.length; i++)
        {
            myList[i] = Math.random() * 100;
        }
        System.out.print("Các phần tử trong mảng: ");
        for (int i = 0; i < myList.length; i++)
        {
            System.out.print(myList[i] + " ");
        }
    }
}
```

Kết quả

```
Cac phan tu trong mang: 49.91437233322772 42.229356830793805 44.54614164159645
23.346046110041318 37.47573911419114
```

6.3.4/ Kỹ thuật lập trình mảng 1 chiều trong Java

3. Tính tổng các phần tử trong mảng

```
public class ViDuMang {
    public static void main(String[] args) {

        double[] myList = {2.3, 5.0, 7.1, 5.5, 9.2};
        double total = 0;

        for (int i = 0; i < myList.length; i++) {
            total += myList[i];
        }
        System.out.print("Tong cac phan tu trong mang = " + total);
    }
}
```

4. Tìm phần tử nhỏ nhất trong mảng

```
public class ViDuMang {
    public static void main(String[] args) {
        double[] myList = {2.3, 5.0, 7.1, 5.5, 9.2};
        double min = myList[0];
        int indexMin = 0;

        for (int i = 0; i < myList.length; i++) {
            if (myList[i] < min){
                min = myList[i];
                indexMin = i;
            }
        }
        System.out.println("Phan tu min trong mang = " + min);
        System.out.println("Chi so phan tu min trong mang = " + indexMin);
    }
}
```

Kết Quả

```
Phan tu min trong mang = 2.3
Chi so phan tu min trong mang = 0
```

5. Tìm phần tử lớn nhất trong mảng

```
public class ViDuMang {
    public static void main(String[] args) {
        double[] myList = {2.3, 5.0, 7.1, 5.5, 9.2};
        double max = myList[0];
        int indexMax = 0;

        for (int i = 0; i < myList.length; i++) {
            if (myList[i] > max){
                max = myList[i];
                indexMax = i;
            }
        }
        System.out.println("Phan tu max trong mang = " + max);
        System.out.println("Chi so phan tu min trong mang = " + indexMax);
    }
}
```

Kết Quả

```
Phan tu max trong mang = 9.2
Chi so phan tu min trong mang = 4
```

6.3.4/ Kỹ thuật lập trình mảng 1 chiều trong Java

6. Duyệt mảng với vòng lặp for-each

```java
public class ViDuMang {
    public static void main(String[] args) {
        double[] myList = {2.3, 5.0, 7.1, 5.5, 9.2};
        double sum = 0;
        double average = 0;
        //tính tổng các phần tử trong mảng
        for (double number: myList) {
            sum += number;
        }
        //lấy kích thước của mảng
        int arrayLength = myList.length;
        //tính trung bình cộng các phần tử trong mảng
        average =  sum/arrayLength;
        System.out.println("Tong = " + sum);
        System.out.println("Trung binh = " + average);
    }
}
```

Kết Quả

```
Tong = 29.099999999999998
Trung binh = 5.819999999999999
```

6.3.4/ Kỹ thuật lập trình mảng 1 chiều trong Java

7. Truyền mảng cho phương thức trong Java

```
public class ViDuMang {
    public static void swapFirstTwoInArray(int[] array) {
        int temp = array[0];
        array[0] = array[1];
        array[1] = temp;
    }
    public static void main(String[] args) {
        int[] arr = {1, 2};
        System.out.println("Mang truoc khi goi ham swap");
        System.out.println("array la {" + arr[0] + ", " + arr[1] + "}");
        swapFirstTwoInArray(arr);
        System.out.println("Mang sau khi goi ham swap");
        System.out.println("array la {" + arr[0] + ", " + arr[1] + "}");
    }
}
```

Kết Quả

```
Mang truoc khi goi ham swap
array la {1, 2}
Mang sau khi goi ham swap
array la {2, 1}
```

6.3.4/ Kỹ thuật lập trình mảng 1 chiều trong Java

8. Kiểu dữ liệu trả về của phương thức là một mảng

```
public class ViDuMang {
    public static int[] reverse(int[] list) {
        int[] result = new int[list.length];
        for (int i=0, j=result.length-1; i<list.length; i++,j--) {
            result[j] = list[i];
        }
        return result;
    }
    public static void main(String[] args) {
        int[] list1 = {1, 2, 3, 4, 5, 6};
        int[] list2 = reverse(list1);
        System.out.print("Mang list2: ");
        for(int item:list2){
            System.out.print(item + " ");
        }
    }
}
```

Kết Quả

```
Mang list2: 6 5 4 3 2 1
```

6.3.4/ Kỹ thuật lập trình mảng 1 chiều trong Java

9. Lớp Arrays trong Java

```
import java.util.Arrays;

public class ViDuMang {

    public static void main(String[] args) {
        double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};
        System.out.println("Index cua phan tu 1.9 la:" + Arrays.binarySearch(numbers, 1.9));
        Arrays.sort(numbers);//sắp xếp mảng tang dan
        System.out.println("Mang da sap xep tang dan:" + Arrays.toString(numbers));
    }
}
```

Kết Quả

```
Index cua phan tu 1.9 la:2
Mang da sap xep tang dan:[1.9, 2.9, 3.4, 3.5, 4.4, 6.0]
```

Tham khảo

- **Class String**

  (https://docs.oracle.com/javase/9/docs/api/java/lang/String.html)

- **Arrays, The Java™ Tutorials**

  (https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html)

- **The For-Each Loop**

  (https://docs.oracle.com/javase/8/docs/technotes/guides/language/foreach.html)

- **Java - Strings Class**

  (https://www.tutorialspoint.com/java/java_strings.htm)