VIETNAM AUSTRALIA Vocational College

Slide-1.4: Properties, Arrays, Collections trong Collection

Giảng viên: Nguyễn Bá Minh Đạo

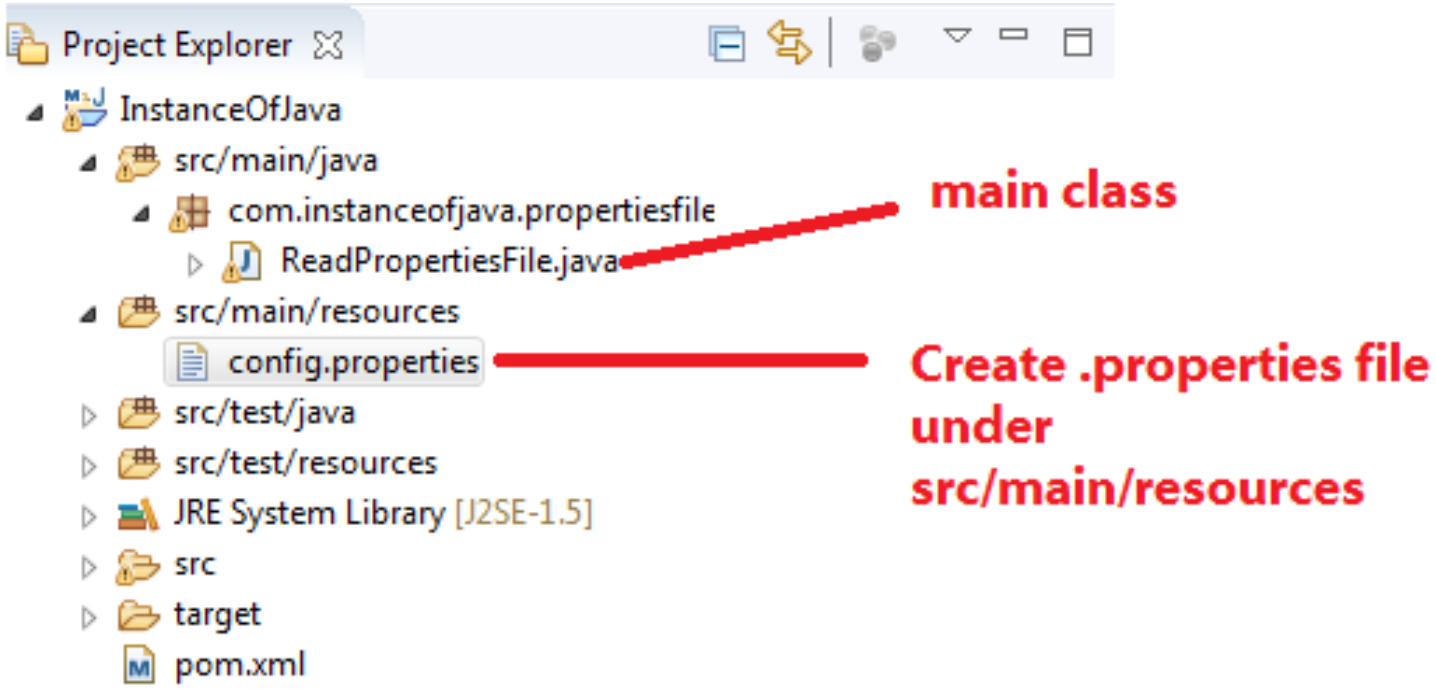
Nội dung

1. Lớp Properties trong Java
2. Lớp Arrays trong Java
3. Lớp Collections trong Java

Lớp Properties trong Java

Giới thiệu lớp Properties

- Lớp **Properties** trong java được sử dụng để tạo ra đối tượng chứa cặp khóa (key) và giá trị (value) như một chuỗi. Lớp **java.util.Properties** là một lớp con của **Hashtable**.
- Lớp **Properties** sử dụng để lấy giá trị (value) dựa trên khóa (key) của thuộc tính.



Lớp Properties trong Java

Giới thiệu lớp Properties

- Lớp **Properties** cung cấp các phương thức lấy dữ liệu từ các file **.properties** và lưu trữ dữ liệu vào file **.properties**.
- Các file **.properties** có thể được sử dụng để có được **properties** của hệ thống.
- File **Properties** không cần biên dịch lại, nếu thông tin được thay đổi từ file **.properties**
- Nó được sử dụng để lưu trữ thông tin mà sẽ được thay đổi thường xuyên.
- Lớp **java.util.Properties** được định nghĩa như sau:

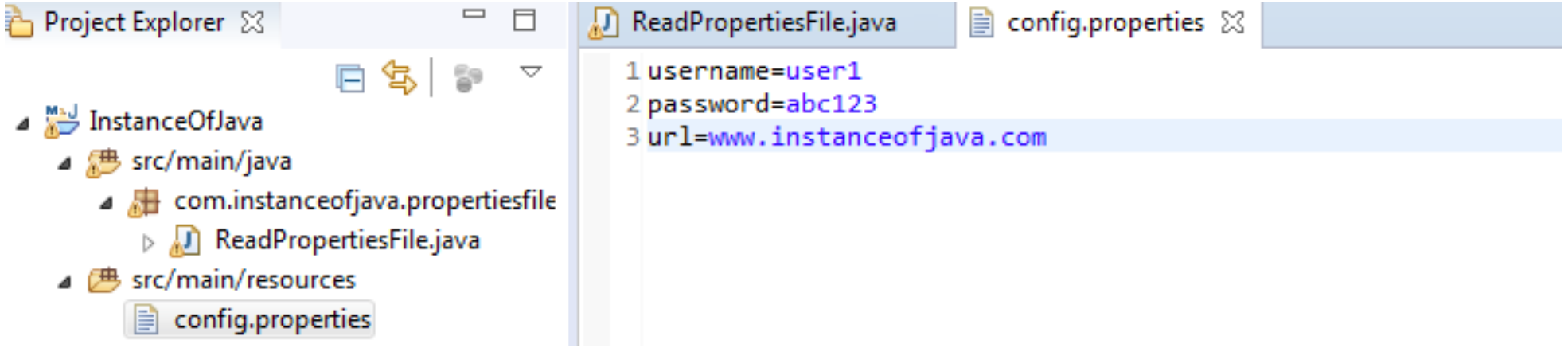
```
public class Properties extends Hashtable<Object,Object> {


}
```

Lớp Properties trong Java

Nội dung của file .properties

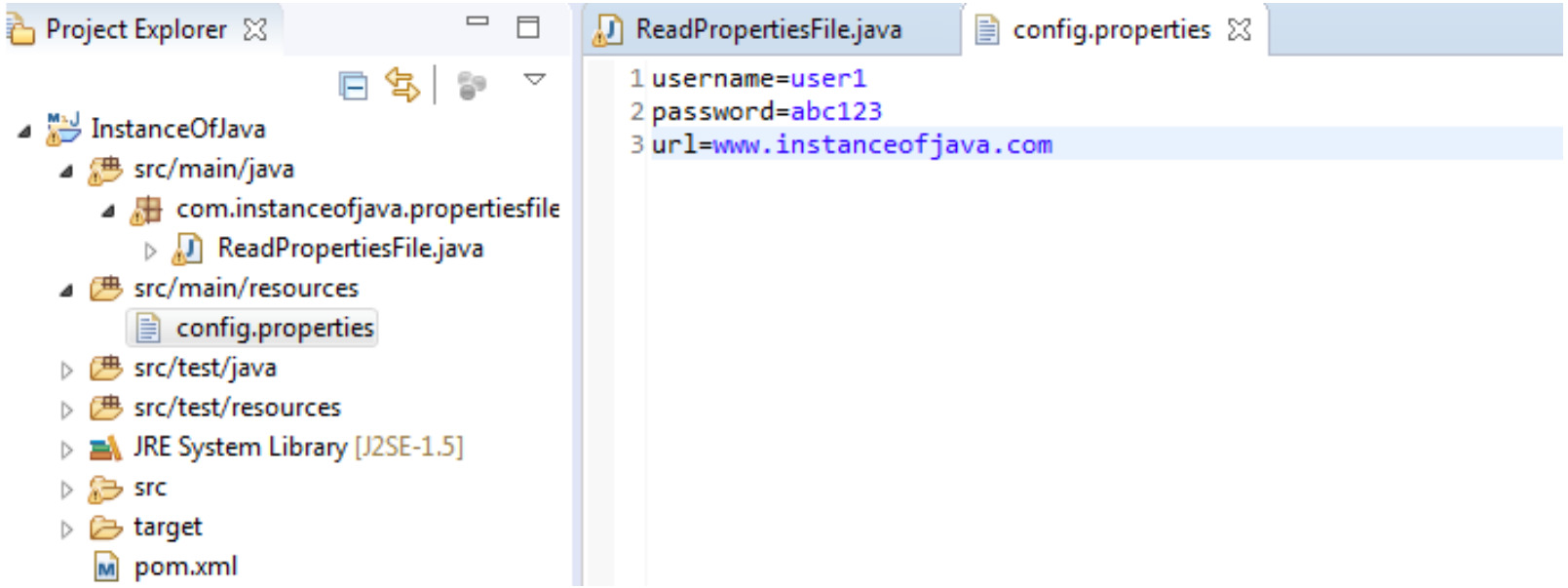
- Mỗi cặp khóa – giá trị (**key-value**) được viết trên một dòng, phân cách bằng dấu **=**
- Sử dụng **dấu #** để thêm ghi chú cho các cặp khóa – giá trị (key-value).
- **Có thể** thêm trùng khóa, tuy nhiên lớp **Properties** chỉ lấy giá trị của khóa cuối cùng, do lớp **Properties** sử dụng map để lưu dữ liệu nên cặp khóa cuối cùng được giữ lại.
- Các dòng bắt đầu bằng các ký tự **!** hoặc **#** bị bỏ qua. **Dòng trắng** cũng bị bỏ qua.
- **Tên key** không được chứa khoảng trắng ở giữa. **Ví dụ**: key 1 là sai, phải đặt là key1.



Lớp Properties trong Java

Nội dung của file .properties

- **Có thể** có thêm khoảng trắng ở 2 đầu của khóa và giá trị: tuy nhiên lớp **Properties** sẽ tự động cắt khoảng trắng ở 2 đầu của khóa, cắt khoảng trắng đầu của giá trị, khoảng trắng ở cuối giá trị vẫn giữ nguyên.
- **Ví dụ**: name=javacoder và name = javacoder là giống nhau.



Lớp Properties trong Java

Nội dung của file .properties

- **Giá trị** có thể kéo dài một vài dòng nếu mỗi dòng được kết thúc bởi dấu gạch chéo ngược (‘****‘). Ví dụ:

```
targetCities=\
             Detroit,\
             Chicago,\
             Los Angeles
```

=> Điều này tương đương với targetCities = Detroit, Chicago, Los Angeles (khoảng trắng ở đầu dòng được bỏ qua).

- Một số lưu ý khác:
  - Ký tự **\n**, **\r**, và **\t** có thể được sử dụng.
  - Ký tự dấu gạch chéo **** ngược phải được thay thế bằng dấu gạch chéo kép (****).
    Ví dụ: path=c:\\docs\\doc1
  - Ký tự **UNICODE** có thể được nhập trong Java sử dụng tiền tố **\u**.
    Ví dụ: \u002c.

Lớp Properties trong Java

Các phương thức (method) của lớp Properties

Phương thức	Mô tả
public void load(Reader r)	Tải dữ liệu từ đối tượng Reader.
public void load(InputStream is)	Tải dữ liệu từ đối tượng InputStream.
public String getProperty(String key)	Trả về giá trị dựa trên key.
public void setProperty(String key,String value)	Gán giá trị (value) dựa vào khóa (key).

Lớp Properties trong Java

Các phương thức (method) của lớp Properties

Phương thức	Mô tả
public void store(Writer w, String comment)	Lưu các thuộc tính trong đối tượng Writer.
public void store(OutputStream os, String comment)	Lưu các thuộc tính trong đối tượng OutputStream.
storeToXML(OutputStream os, String comment)	Lưu các thuộc tính trong đối tượng Writer để tạo tài liệu xml.
public void storeToXML(Writer w, String comment, String encoding)	Lưu các thuộc tính trong đối tượng Writer để tạo ra tài liệu xml với mã hoá được chỉ định.

Lớp Properties trong Java

Ví dụ đọc thông tin của file config.properties bất kỳ

```
public class PropertiesExample01 {
    public static void main(String[] args) throws Exception {
        // create reader object
        FileReader reader = new FileReader("[your-path]\\config.properties");

        // crate properties object
        Properties properties = new Properties();
        properties.load(reader);

        // show file info
        System.out.println(properties.getProperty("courseName"));
        System.out.println(properties.getProperty("courseDuration"));
    }
}
```

Lớp Properties trong Java

Ví dụ về lớp Properties lấy các thuộc tính của hệ thống

```
public class PropertiesExample02 {
    public static void main(String[] args) throws Exception {
        // get system properties
        Properties p = System.getProperties();
        Set<Entry<Object, Object>> set = p.entrySet();

        // show system properties
        Iterator<Entry<Object, Object>> itr = set.iterator();
        while (itr.hasNext()) {
            Map.Entry<Object, Object> entry = itr.next();
            Object key = entry.getKey();
            Object value = entry.getValue();
            System.out.println(key + " = " + value);
        }
    }
}
```

Lớp Properties trong Java

Ví dụ sử dụng lớp Properties để tạo file .properties

```java
public class PropertiesExample03 {
    public static void main(String[] args) throws Exception {
        OutputStream output = null;
        try {
            // Properties File output at project root folder
            output = new FileOutputStream("[your-path]\\database.config.properties");

            // create properties object
            Properties prop = new Properties();

            // set the properties value
            prop.setProperty("database", "localhost");
            prop.setProperty("dbuser", "admin");
            prop.setProperty("dbpassword", "admin");

            // save properties to a file
            prop.store(output, "Config Database connection");
        } catch (IOException io) {
            io.printStackTrace();
        } finally {
            if (output != null) {
                try {
                    output.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

        }
    }
}
```

PropertiesEx... PropertiesEx... database.con... ×

```
1 #Config Database connection
2 #Wed Apr 20 17:42:51 ICT 2022
3 database=localhost
4 dbpassword=admin
5 dbuser=admin
6
```

Lớp Arrays trong Java

Giới thiệu lớp tiện ích Arrays

- Lớp **java.util.Arrays** trong java được sử dụng để thực hiện một số thao tác như sao chép, sắp xếp và tìm kiếm các phần tử trên các mảng.
- Lớp tiện ích này có các phương thức rất hữu ích để hỗ trợ chúng ta trong quá trình lập trình. Vì vậy, chúng ta nên biết một số phương thức phổ biến của lớp Arrays này.

Lớp Arrays trong Java

Phương thức toString()

- Phương thức **toString()** được sử dụng để hiển thị tất cả các phần tử của một mảng.
- Phương thức này trả về chuỗi của tất cả các phần tử của một mảng.
-
- Chuỗi này bao gồm tất cả các phần tử được bao quanh trong "[]".
- Tất cả các phần tử được phân tách bằng "," (Dấu phẩy và khoảng cách).

```
public static void main(String[] args) {
    // An array of byte
    byte[] b = { 10, 20, 30 };

    System.out.println(Arrays.toString(b)); // Output : [10, 20, 30]

    // An array of short
    short[] s = { 40, 50, 60, 70 };
    System.out.println(Arrays.toString(s)); // Output : [40, 50, 60, 70]

    // An array of int
    int[] i = { 12, 21, 42, 68 };
    System.out.println(Arrays.toString(i)); // Output : [12, 21, 42, 68]

    // An array of long
    long[] l = { 100, 110, 120, 130, 140, 150 };
    System.out.println(Arrays.toString(l)); // Output : [100, 110, 120, 130, 140, 150]

    // An array of double
    double[] d = { 12.5, 14.9, 87.4, 55.8 };
    System.out.println(Arrays.toString(d)); // Output : [12.5, 14.9, 87.4, 55.8]

    // An array of char
    char[] c = { 'A', 'B', 'C', 'D', 'E' };
    System.out.println(Arrays.toString(c)); // Output : [A, B, C, D, E]

    // An array of boolean
    boolean[] bln = { true, false, false, true };
    System.out.println(Arrays.toString(bln)); // Output : [true, false, false, true]

    // An array of String
    String[] str = { "welcome", "to", "javacoder" };
    System.out.println(Arrays.toString(str)); // Output : [welcome, to, javacoder]
}
```

Lớp Arrays trong Java

Phương thức deepToString()

- Phương thức **deepToString()** được sử dụng để nhận được biểu diễn chuỗi của các mảng đa chiều.
- Phương thức này trả về nội dung sâu của mảng được chỉ định.
- Nếu mảng được chỉ định chứa các mảng khác như là các phần tử thì nó sẽ trả lại nội dung của các mảng đó.

```
public static void main(String[] args) {
    // One Dimensional Array

    String[] oneDArray = new String[] {
        "ONE", "TWO", "THREE", "FOUR", "FIVE"
    };
    System.out.println("One Dimensional Array : ");
    // Printing one dimensional array contents using deepToString() method
    System.out.println(Arrays.deepToString(oneDArray));

    // Two Dimensional Array

    String[][] twoDArray = new String[][] {
        { "ONE", "TWO", "THREE", "FOUR" },
        { "FIVE", "SIX", "SEVEN" },
        { "EIGHT", "NINE", "TEN", "ELEVEN", "TWELVE" }
    };
    System.out.println("Two Dimensional Array : ");
    // Printing two dimensional array contents using deepToString() method
    System.out.println(Arrays.deepToString(twoDArray));

    // Three Dimensional Array
    String[][][] threeDArray = new String[][][] {
        {
            { "ONE", "TWO", "THREE" },
            { "FOUR", "FIVE", "SIX", "SEVEN" }
        }, {
            { "EIGHT", "NINE", "TEN", "ELEVEN" },
            { "TWELVE", "THIRTEEN", "FOURTEEN" }
        }, {
            { "FIFTEEN", "SIXTEEN" },
            { "SEVENTEEN", "EIGHTEEN", "NINETEEN" },
            { "TWENTY", "TWENTY ONE" }
        }
    };
    System.out.println("Three Dimensional Array : ");
    // Printing three dimensional array contents using deepToString() method
    System.out.println(Arrays.deepToString(threeDArray));
}
```

Lớp Arrays trong Java

Phương thức sort()

- Phương thức **sort()** được sử dụng để sắp xếp các phần tử của một mảng theo thứ tự tăng dần.
- Phương thức này, bên trong sử dụng thuật toán **Quicksort** để sắp xếp các phần tử có kiểu nguyên thủy (primitive type).
- Sử dụng giải thuật **MergeSort** để sắp xếp các phần tử có kiểu đối tượng (Object).

```
public static void main(String[] args) {
    // An array of byte
    byte[] b = { 51, 22, 8, 37 };
    Arrays.sort(b); // sorts elements of the specified array in ascending order
    System.out.println(Arrays.toString(b)); // Output : [8, 22, 37, 51]

    // An array of short
    short[] s = { 24, 5, 21, 12, 19 };
    Arrays.sort(s);
    System.out.println(Arrays.toString(s)); // Output : [5, 12, 19, 21, 24]

    // An array of int
    int[] i = { 42, 12, 68, 21 };
    Arrays.sort(i);
    System.out.println(Arrays.toString(i)); // Output : [12, 21, 42, 68]

    // An array of long
    long[] l = { 879, 412, 258, 985, 856 };
    Arrays.sort(l);
    System.out.println(Arrays.toString(l)); // Output : [258, 412, 856, 879, 985]

    // An array of double
    double[] d = { 12.5, 87.4, 41.24, 14.9, 55.8 };
    Arrays.sort(d);
    System.out.println(Arrays.toString(d)); // Output : [12.5, 14.9, 41.24, 55.8, 87.4]

    // An array of char
    char[] c = { 'Z', 'B', 'X', 'L', 'b', 'A' };
    Arrays.sort(c);
    System.out.println(Arrays.toString(c)); // Output : [A, B, L, X, Z, b]

    // An array of String
    String[] str = { "welcome", "to", "javacoder" };
    Arrays.sort(str);
    System.out.println(Arrays.toString(str)); // Output : [javacoder, to, welcome]

    // An array of boolean
    boolean[] bln = { true, false, false, true };
    // Arrays.sort(bln); // Compile time error
    // There is no sort method in Arrays class which sorts boolean type array
}
```

Lớp Arrays trong Java

Phương thức binarySearch()

- Phương thức **binarySearch()** được sử dụng để tìm vị trí của phần tử trong mảng bằng phương thức tìm kiếm nhị phân (**binary search**).
- Các phần tử trong mảng phải được sắp xếp trước khi gọi phương thức này.
- Nếu chưa sắp xếp, kết quả sẽ không chính xác.

```
public static void main(String[] args) {
    // An array of int
    int[] i = { 24, 13, 45, 37, 84, 13, 28 };

    // Sorting the int array
    Arrays.sort(i);

    // Printing sorted array
    System.out.println(Arrays.toString(i)); // Output : [13, 13, 24, 28, 37, 45, 84]

    // Searching the value
    System.out.println(Arrays.binarySearch(i, 37)); // Output : 4
    System.out.println(Arrays.binarySearch(i, 13)); // Output : 1
    System.out.println(Arrays.binarySearch(i, 1));  // Output : -1
    System.out.println(Arrays.binarySearch(i, 55)); // Output : -7

    // An array of char
    char[] c = { 'X', 'n', 'F', 's', 'D', 'J', 'j', 'F' };

    // Sorting the char array
    Arrays.sort(c);

    // Printing Sorted array
    System.out.println(Arrays.toString(c)); // Output : [D, F, F, J, X, j, n, s]

    // Searching the character
    System.out.println(Arrays.binarySearch(c, 'F')); // Output : 1
    System.out.println(Arrays.binarySearch(c, 'J')); // Output : 3
    System.out.println(Arrays.binarySearch(c, 'H')); // Output : -4
    System.out.println(Arrays.binarySearch(c, 'Z')); // Output : -6

    // An array of String
    String[] str = { "First", "second", "Third", "second", "Four", "fifth" };

    // Sorting the String array
    Arrays.sort(str);

    // Printing Sorted array
    System.out.println(Arrays.toString(str)); // Output : [First, Four, Third, fifth, second, second]

    // Searching the string in the array
    System.out.println(Arrays.binarySearch(str, "Third"));  // Output : 2
    System.out.println(Arrays.binarySearch(str, "second")); // Output : 4
    System.out.println(Arrays.binarySearch(str, "One"));    // Output : -3
    System.out.println(Arrays.binarySearch(str, "sixth"));  // Output : -7
}
```

Lớp Arrays trong Java

Phương thức fill()

- Phương thức **fill()** được sử dụng để gán giá trị xác định cho mỗi phần tử của một mảng.

- Phương thức này hữu ích trong việc khởi tạo tất cả các phần tử của một mảng với một giá trị.

```
public static void main(String[] args) {
    // An array of int
    int[] i = new int[5];
    Arrays.fill(i, 10); // Assigns 10 to each element of the array
    System.out.println(Arrays.toString(i)); // Output : [10, 10, 10, 10, 10]

    // An array of double
    double[] d = { 12.5, 14.8, 45.9, 23.5 };
    Arrays.fill(d, 53.6); // Assigns 53.6 to each element of the array
    System.out.println(Arrays.toString(d)); // Output : [53.6, 53.6, 53.6, 53.6]

    // An array of boolean
    boolean[] bln = new boolean[5];
    Arrays.fill(bln, true); // Assigns true to each element of the array
    System.out.println(Arrays.toString(bln)); // Output : [true, true, true, true, true]

    // An array of char
    char[] c = new char[10];
    Arrays.fill(c, 'P'); // Assigns P to each element of the array
    System.out.println(Arrays.toString(c)); // Output : [P, P, P, P, P, P, P, P, P, P]

    // An array of String
    String[] str = { "Java", "Concepts", "basic java", "Arrays Class" };
    Arrays.fill(str, "value"); // Assigns value to each element of the array
    System.out.println(Arrays.toString(str)); // Output : [value, value, value, value]
}
```

Lớp Arrays trong Java

Phương thức copyOf()

- Phương thức **copyOf()** được sử dụng để sao chép mảng được chỉ định vào mảng mới của cùng một kiểu.

- Trong khi sao chép, mảng mới có thể được cắt ngắn hoặc có đệm với các giá trị mặc định để nó có chiều dài chỉ định.

```
public static void main(String[] args) {
    // An array of int
    int[] i = { 1, 21, 15, 48, 79 };
    // Copying array i into array i1
    int[] i1 = Arrays.copyOf(i, 10);
    System.out.println(Arrays.toString(i1)); // Output : [1, 21, 15, 48, 79, 0, 0, 0, 0, 0]

    // An array of double
    double[] d = { 12.5, 45.8, 56.2, 47.9, 23.6, 89.5 };
    // Copying array d into array d1
    double[] d1 = Arrays.copyOf(d, 4);
    System.out.println(Arrays.toString(d1)); // Output : [12.5, 45.8, 56.2, 47.9]

    // An array of boolean
    boolean[] bln = { true, false, true, true, false };
    // Copying array bln into array bln1
    boolean[] bln1 = Arrays.copyOf(bln, 10);
    System.out.println(Arrays.toString(bln1)); // Output : [true, false, true, true, false,
                                               // false, false, false, false, false]

    // An array of char
    char[] c = { 'X', 'B', 'Z', 'H', 'I', 'J' };
    // Copying array c into array c1
    char[] c1 = Arrays.copyOf(c, 5);
    System.out.println(Arrays.toString(c1)); // Output : [X, B, Z, H, I]

    // An array of String
    String[] str = { "java", "j2ee", "struts", "hibernate" };
    // Copying array str into array str1
    String[] str1 = Arrays.copyOf(str, 7);
    System.out.println(Arrays.toString(str1)); // [java, j2ee, struts, hibernate, null, null, null]
}
```

Lớp Arrays trong Java

Phương thức copyOfRange()

- Phương thức **copyOfRange()** được sử dụng để sao chép một phần của một mảng vào mảng khác cùng kiểu.

- Trong khi sao chép mảng mới có thể được cắt ngắn hoặc đệm với các giá trị mặc định để có được chiều dài yêu cầu.

```
public static void main(String[] args) {
    // An array of int
    int[] i = { 1, 21, 15, 48, 79 };
    // Copying some part of array i into array i1
    int[] i1 = Arrays.copyOfRange(i, 2, 7);
    System.out.println(Arrays.toString(i1)); // Output : [15, 48, 79, 0, 0]

    // An array of double
    double[] d = { 12.5, 45.8, 56.2, 47.9, 23.6, 89.5 };
    // Copying some part of array d into array d1
    double[] d1 = Arrays.copyOfRange(d, 2, 5);
    System.out.println(Arrays.toString(d1)); // Output : [56.2, 47.9, 23.6]

    // An array of boolean
    boolean[] bln = { true, false, true, true, false };
    // Copying some part of array bln into array bln1
    boolean[] bln1 = Arrays.copyOfRange(bln, 1, 8);
    System.out.println(Arrays.toString(bln1)); // Output : [false, true, true, false,
                                               //           false, false, false]

    // An array of char
    char[] c = { 'X', 'B', 'Z', 'H', 'I', 'J' };
    // Copying some part of array c into array c1
    char[] c1 = Arrays.copyOfRange(c, 2, 4);
    System.out.println(Arrays.toString(c1)); // Output : [Z, H]

    // An array of String
    String[] str = { "java", "j2ee", "struts", "hibernate" };
    // Copying some part of array str into array str1
    String[] str1 = Arrays.copyOfRange(str, 4, 8);
    System.out.println(Arrays.toString(str1)); // Output : [null, null, null, null]
}
```

Lớp Arrays trong Java

Phương thức asList()

❑ Phương thức **asList()** được sử dụng để tạo một danh sách từ một mảng được chỉ định.

```
public class ArraysExample08 {
    public static void main(String[] args) {
        // An array of Integer
        Integer arr[] = { 4, 6, 1, 8, 3, 9, 7, 4, 2 };

        // Creates a wrapper list over arr[]
        List<Integer> list = Arrays.asList(arr);
        System.out.println(list); // [4, 6, 1, 8, 3, 9, 7, 4, 2]

        // list.add(10); // UnsupportedOperationException
        // list.remove(0); // UnsupportedOperationException

        list.set(0, 5); // overwrite the elements
        System.out.println(list); // [5, 6, 1, 8, 3, 9, 7, 4, 2]
    }
}
```

Lớp Arrays trong Java

Phương thức equals()

- Phương thức **equals()** được sử dụng để so sánh hai mảng có bằng nhau hay không.
- Phương pháp này lấy hai mảng làm tham số và trả về true nếu cả hai mảng có cùng một số phần tử và các cặp tương ứng của các phần tử của cả hai mảng đều bằng nhau.

```
public class ArraysExample09 {
    public static void main(String[] args) {
        String[] s1 = { "java", "j2ee", "struts", "hibernate" };
        String[] s2 = { "jsp", "spring", "jdbc", "hibernate" };
        String[] s3 = { "java", "j2ee", "struts", "hibernate" };
        String[] s4 = { "java", "struts", "j2ee", "hibernate" };

        System.out.println(Arrays.equals(s1, s2)); // Output : false
        System.out.println(Arrays.equals(s1, s3)); // Output : true
        System.out.println(Arrays.equals(s1, s4)); // Output : false
    }
}
```

Lớp Arrays trong Java

Phương thức deepEquals()

- Phương thức **deepEquals()** được sử dụng để so sánh hai **mảng hai chiều** có bằng nhau hay không, thay vì phương thức equals()

```
import java.util.Arrays;

public class ArraysExample10 {
    public static void main(String[] args) {
        String[][] s1 = { { "java", "swings", "j2ee" },
                          { "struts", "jsp", "hibernate" } };
        String[][] s2 = { { "java", "swings", "j2ee" },
                          { "struts", "jsp", "hibernate" } };

        System.out.println(Arrays.equals(s1, s2)); // Output : false
        System.out.println(Arrays.deepEquals(s1, s2)); // Output : true
    }
}
```

Lớp Collections trong Java

Sự khác nhau giữa Collection và Collections

- **Collection** là một interface cấp cao nhất nhất của **Collection Framework**.
- Trong khi đó, **Collections** là một lớp tiện ích.
- **Collections** bao gồm các phương thức static được sử dụng để thao tác trên các đối tượng của Collection (List, ArrayList, LinkedList, Map, Set,…)
- Interface **java.util.Collection** được định nghĩa như sau:

```
public interface Collection<E> extends Iterable<E>{

}
```

- Lớp **java.util.Collections** được định nghĩa như sau:

```
public class Collections {

}
```

Lớp Collections trong Java

Những đặc điểm quan trọng về lớp Collections

- Lớp **Collections** hỗ trợ các thuật toán đa hình (**polymorphic algorithms**) hoạt động trên các **Collection**.
- Lớp **Collections** ném một ngoại lệ **NullPointerException** nếu các **Collection** hoặc các đối tượng lớp cung cấp cho chúng là **null**.

Các thuộc tính của lớp Collections

- **static List EMPTY_LIST**: khởi tạo một danh sách trống, không thể thay đổi (immutable)
- **static Map EMPTY_MAP**: khởi tạo một map trống, không thể thay đổi (immutable)
- **static Set EMPTY_SET**: khởi tạo một tập hợp trống, không thể thay đổi (immutable)

Lớp Collections trong Java

Ví dụ sử dụng thuộc tính EMPTY và phương thức empty() để khởi tạo Collection rỗng

- Các thuộc tính **EMPTY_LIST**, **EMPTY_SET**, **EMPTY_MAP** và các phương thức **emptyList()**, **emptySet()**, **emptyMap()** được sử dụng để khởi tạo một Collection trống, không thể thay đổi (immutable).
- Nếu bạn cố tình thêm, xóa, thay đổi phần tử này, JVM sẽ ném ra ngoại lệ **UnsupportedOperationException**.
- Các thuộc tính và phương thức empty này thường được sử dụng để tránh lỗi **NullPointerException** khi khởi tạo một Collection null hoặc kết quả trả về của một phương thức là null.

Lớp Collections trong Java

Ví dụ sử dụng thuộc tính EMPTY và phương thức empty() để khởi tạo Collection rỗng

```java
public class CollectionsExample01 {
    public static void main(String a[]) {
        // Using empty fields
        List<Integer> list1 = Collections.EMPTY_LIST;
        Set<Integer> set1 = Collections.EMPTY_SET;
        Map<Integer, String> map1 = Collections.EMPTY_MAP;

        // Can't add element
        // list1.add(10); // throw UnsupportedOperationException
        // set1.add(10); // throw UnsupportedOperationException
        // map1.put(1, "one"); // throw UnsupportedOperationException

        // Using empty methods
        List<Integer> list2 = Collections.emptyList();
        Set<Integer> set2 = Collections.emptySet();
        Map<Integer, String> map2 = Collections.emptyMap();

        // Can't add element
        // list2.add(10); // throw UnsupportedOperationException
        // set2.add(10); // throw UnsupportedOperationException
        // map2.put(1, "one"); // throw UnsupportedOperationException

        // After init
        list2 = new ArrayList<>();
        list2.add(1); // run normally
    }
}
```

Lớp Collections trong Java

Ví dụ sử dụng phương thức addAll() để thêm tất cả các phần tử của một danh sách này vào một danh sách khác

```
public class CollectionsExample02 {
    public static void main(String a[]) {
        List<String> list = new ArrayList<String>();
        list.add("C");
        list.add("Core Java");
        list.add("Advance Java");
        System.out.println("Initial collection value:" + list);

        Collections.addAll(list, "Servlet", "JSP");
        System.out.println("After adding elements collection value:" + list);

        String[] strArr = { "C#", ".Net" };
        Collections.addAll(list, strArr);
        System.out.println("After adding array collection value:" + list);
    }
}
```

Lớp Collections trong Java

Sử dụng phương thức min(), max(), search(), sort(), reverse(), reverseOrder(), reverseOrder(Comparator) với kiểu dữ liệu cơ bản (String, Wrapper)

```
public static void main(String a[]) {
    List<Integer> list = new ArrayList<Integer>();
    list.add(46);
    list.add(67);
    list.add(24);
    list.add(16);
    list.add(8);
    list.add(12);
    System.out.println("Maximum value: " + Collections.max(list));
    System.out.println("Minimum value: " + Collections.min(list));
    System.out.println("Index of value 24 : " + Collections.binarySearch(list, 24));
    System.out.println("Index of value 10 : " + Collections.binarySearch(list, 10));

    Collections.sort(list);
    System.out.println("Sorted ASC: " + list);

    Collections.reverse(list);
    System.out.println("Sorted DESC: " + list);

    Comparator<Integer> compareDesc = Collections.reverseOrder();
    Collections.sort(list, compareDesc);
    System.out.println("Sorted ASC: " + list);

    Comparator<Integer> compareAsc = Collections.reverseOrder(compareDesc);
    Collections.sort(list, compareAsc);
    System.out.println("Sorted DESC: " + list);
}
```

Lớp Collections trong Java

Sử dụng phương thức min(), max(), search(), sort(), reverse(), reverseOrder(), reverseOrder(Comparator) với kiểu dữ liệu do người dùng tự định nghĩa (Object)

```java
public class Student {
    private int id;
    private String name;
    private int age;

    public Student(int id, String name, int age) {
        super();
        this.id = id;
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", name="
                        + name + ", age=" + age + "]";
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}
```

```java
public class StudentAgeComparator implements Comparator<Student> {

    @Override
    public int compare(Student o1, Student o2) {
        // sort student's age by ASC
        if (o1.getAge() < o2.getAge()) {
            return -1;
        } else if (o1.getAge() == o2.getAge()) {
            return 0;
        } else {
            return 1;
        }
    }
}
```

```java
public class CollectionsExample04 {
    public static void main(String a[]) {

    private static void print(List<Student> students) {
        for (Student student : students) {
            System.out.println(student);
        }
        System.out.println("---");
    }
}
```

Lớp Collections trong Java

Sử dụng phương thức min(), max(), search(), sort(), reverse(), reverseOrder(), reverseOrder(Comparator) với kiểu dữ liệu do người dùng tự định nghĩa (Object)

```
// Create list
List<Student> students = new ArrayList<>();
Student student1 = new Student(1, "myname1", 15);
Student student2 = new Student(2, "myname2", 20);
Student student3 = new Student(3, "myname3", 17);
Student student4 = new Student(4, "myname4", 10);
Student student5 = new Student(5, "myname5", 19);
Student student6 = new Student(6, "myname6", 19);
students.add(student3);
students.add(student1);
students.add(student2);
students.add(student5);
students.add(student4);

// Init comparator
StudentAgeComparator ageComparator = new StudentAgeComparator();

// Using comparator
System.out.println("Maximum value: " + Collections.max(students, ageComparator));
System.out.println("Minimum value: " + Collections.min(students, ageComparator));
System.out.println("Index of student1 : " + Collections.binarySearch(students, student1, ageComparator));
System.out.println("Index of student6 : " + Collections.binarySearch(students, student6, ageComparator));
System.out.println("---");
```

```
Collections.sort(students, ageComparator);
System.out.println("Sorted ASC: ");
print(students);

Collections.reverse(students);
System.out.println("Sorted DESC: ");
print(students);

Comparator<Student> compareDesc = Collections.reverseOrder(ageComparator);
Collections.sort(students, compareDesc);
System.out.println("Sorted DESC: ");
print(students);

Comparator<Student> compareAsc = Collections.reverseOrder(compareDesc);
Collections.sort(students, compareAsc);
System.out.println("Sorted ASC: ");
print(students);
```

Trong hàm main

Lớp Collections trong Java

*Sử dụng phương thức min(), max(), search(), sort(), reverse(), reverseOrder(), reverseOrder(**Comparable**) với kiểu dữ liệu do người dùng tự định nghĩa (Object)*

```
public class Student02 implements Comparable<Student02> {
    private int id;
    private String name;
    private int age;

    public Student02(int id, String name, int age) {

    @Override
    public int compareTo(Student02 student) {
        // sort student's age by ASC
        if (this.getAge() < student.getAge()) {
            return -1;
        } else if (this.getAge() == student.getAge()) {
            return 0;
        } else {
            return 1;
        }
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" +
                        name + ", age=" + age + "]";
    }

    public int getId() {

    public String getName() {

    public int getAge() {
}
```

Lớp Collections trong Java

Sử dụng phương thức min(), max(), search(), sort(), reverse(), reverseOrder(), reverseOrder(Comparable) với kiểu dữ liệu do người dùng tự định nghĩa (Object)

```
// Create list
List<Student02> students = new ArrayList<>();
Student02 student1 = new Student02(1, "myname1", 15);
Student02 Student02 = new Student02(2, "myname2", 20);
Student02 student3 = new Student02(3, "myname3", 17);
Student02 student4 = new Student02(4, "myname4", 10);
Student02 student5 = new Student02(5, "myname5", 19);
Student02 student6 = new Student02(6, "myname6", 19);
students.add(student3);
students.add(student1);
students.add(Student02);
students.add(student5);
students.add(student4);

// Không cần cung cấp bộ so sánh, bởi vì nó đã được cài đặt trong lớp Student02
System.out.println("Maximum value: " + Collections.max(students));
System.out.println("Minimum value: " + Collections.min(students));
System.out.println("Index of student1 : " + Collections.binarySearch(students, student1));
System.out.println("Index of student6 : " + Collections.binarySearch(students, student6));
System.out.println("---");
```

```
Collections.sort(students);
System.out.println("Sorted ASC: ");
print(students);

Collections.reverse(students);
System.out.println("Sorted DESC: ");
print(students);

Comparator<Student02> compareDesc = Collections.reverseOrder();
Collections.sort(students, compareDesc);
System.out.println("Sorted DESC: ");
print(students);

Comparator<Student02> compareAsc = Collections.reverseOrder(compareDesc);
Collections.sort(students, compareAsc);
System.out.println("Sorted ASC: ");
print(students);
```

Trong hàm main

Lớp Collections trong Java

Sử dụng phương thức frequency() để đếm số lần xuất hiện của phần tử

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class CollectionsExample06 {
    public static void main(String args[]) {
        List<Integer> myList = new ArrayList<>();
        myList.add(10);
        myList.add(20);
        myList.add(10);
        myList.add(20);
        myList.add(30);
        myList.add(10);
        System.out.println("Elements of ArrayList: " + myList);
        System.out.println("No. of times 10 exists: " + Collections.frequency(myList, 10));
        System.out.println("No. of times 20 exists: " + Collections.frequency(myList, 20));
        System.out.println("No. of times 30 exists: " + Collections.frequency(myList, 30));
    }
}
```

Lớp Collections trong Java

Sử dụng phương thức copy() để sao chép một list này sang một list khác

```
public static void main(String args[]) {
    List<String> firstList = new ArrayList<>();
    firstList.add("10");
    firstList.add("20");
    firstList.add("30");
    System.out.println("Elements firstList: " + firstList);

    List<String> secondList = new ArrayList<>();
    secondList.add("one");
    secondList.add("two");
    secondList.add("three");
    System.out.println("Elements secondList: " + secondList);

    Collections.copy(secondList, firstList);
    System.out.println("Elements of secondList after copying firstList: " + secondList);

    List<String> thirdList = new ArrayList<>();
    thirdList.add("one");
    thirdList.add("two");
    thirdList.add("three");
    thirdList.add("four");
    thirdList.add("five");
    System.out.println("\nElements thirdList: " + thirdList);

    Collections.copy(thirdList, firstList);
    System.out.println("Elements of thirdList after copying firstList: " + thirdList);
}
```

Lớp Collections trong Java

Sử dụng phương thức swap() để hoán đổi vị trí của 2 phần tử

```
public class CollectionsExample08 {
    public static void main(String args[]) {
        List<Integer> myList = new ArrayList<>();
        myList.add(50);
        myList.add(10);
        myList.add(20);
        myList.add(40);
        System.out.println("Elements before swap: " + myList);

        Collections.swap(myList, 0, 1);
        System.out.println("Elements after 0,1 swap: " + myList);

        Collections.swap(myList, 2, 3);
        System.out.println("Elements after 2,3 swap: " + myList);
    }
}
```

Lớp Collections trong Java

Sử dụng phương thức shuffle() để truy cập ngẫu nhiên các phần tử

```
public class CollectionsExample09 {
    public static void main(String args[]) {
        List<Integer> myList = new ArrayList<>();
        myList.add(30);
        myList.add(10);
        myList.add(20);
        myList.add(40);
        System.out.println("Elements before shuffle: " + myList);

        Collections.shuffle(myList);
        System.out.println("Elements after first shuffle: " + myList);

        Collections.shuffle(myList);
        System.out.println("Elements after second shuffle: " + myList);

        Collections.shuffle(myList);
        System.out.println("Elements after third shuffle: " + myList);
    }
}
```

Lớp Collections trong Java

Sử dụng phương thức rotate() để xoay các phần tử trong danh sách

```
public class CollectionsExample10 {
    public static void main(String args[]) {
        List<Integer> myList = new ArrayList<>();
        myList.add(10);
        myList.add(20);
        myList.add(30);
        myList.add(40);
        myList.add(50);
        myList.add(60);
        System.out.println("Elements myList before rotate: " + myList);

        Collections.rotate(myList, 3);
        System.out.println("Elements myList after rotate: " + myList);
    }
}
```

Lớp Collections trong Java

Sử dụng phương thức replaceAll() để tìm kiếm và thay thế các phần tử bằng một phần tử khác

```
public static void main(String args[]) {
    List<Integer> myList = new ArrayList<>();
    myList.add(10);
    myList.add(20);
    myList.add(30);
    myList.add(40);
    System.out.println("Elements myList before replacing: " + myList);

    boolean success = Collections.replaceAll(myList, 10, 100);
    System.out.println("Replace operation successful: " + success);
    System.out.println("Elements after replacing: " + myList);

    success = Collections.replaceAll(myList, 50, 200);
    System.out.println("Replace operation successful: " + success);
    System.out.println("Elements after replacing: " + myList);
}
```

Lớp Collections trong Java

Sử dụng phương thức fill() để thay thế tất cả các phần tử trong danh sách bằng một phần tử bất kỳ

```
public static void main(String args[]) {
    List<Integer> myList = new ArrayList<>();
    myList.add(10);
    myList.add(20);
    myList.add(30);
    myList.add(40);
    System.out.println("Elements before fill: " + myList);

    Collections.fill(myList, 0);
    System.out.println("Elements after fill: " + myList);

    List<String> namesList = new ArrayList<>();
    namesList.add("wellcome");
    namesList.add("to");
    namesList.add("javacoder");
    System.out.println("\nElements before fill: " + namesList);

    Collections.fill(namesList, null);
    System.out.println("Elements after fill: " + namesList);
}
```

Lớp Collections trong Java

Sử dụng phương thức disjoint() để kiểm tra hai danh sách có chứa bất kỳ phần tử nào giống nhau không

```
public static void main(String args[]) {
    List<Integer> list1 = new ArrayList<>();
    list1.add(10);
    list1.add(20);
    list1.add(30);

    List<Integer> list2 = new ArrayList<>();
    list2.add(60);
    list2.add(40);
    list2.add(20);

    List<Integer> list3 = new ArrayList<>();
    list3.add(60);
    list3.add(40);
    list3.add(50);

    System.out.println("Elements of list1: " + list1);
    System.out.println("Elements of list2: " + list2);
    System.out.println("Elements of list3: " + list3);

    boolean exists = Collections.disjoint(list1, list2);
    System.out.println("\nlist1 and list2 contains same elements: " + exists); // false

    exists = Collections.disjoint(list1, list3);
    System.out.println("list1 and list3 contains same elements: " + exists); // true
}
```

Lớp Collections trong Java

Sử dụng phương thức indexOfSubList() và lastIndexOfSubList() để tìm vị trí xuất hiện đầu tiên và cuối cùng của một danh sách này trong một danh sách khác

```
List<Integer> list1 = new ArrayList<>();
list1.add(10);
list1.add(20);
list1.add(30);
list1.add(40);
list1.add(100);
list1.add(20);
list1.add(30);
list1.add(400);

List<Integer> list2 = new ArrayList<>();
list2.add(20);
list2.add(30);

List<Integer> list3 = new ArrayList<>();
list3.add(20);
list3.add(40);
```

Trong hàm main

```
System.out.println("list1 elements: " + list1);
System.out.println("list2 elements: " + list2);
System.out.println("list3 elements: " + list3);

int num1 = Collections.indexOfSubList(list1, list2);
System.out.println("\nFirst index list2 in list1: " + num1);

int num2 = Collections.lastIndexOfSubList(list1, list2);
System.out.println("Last index list2 in list1: " + num2);

int num3 = Collections.lastIndexOfSubList(list1, list3);
System.out.println("\nFirst index num3 in list1: " + num3);

int num4 = Collections.lastIndexOfSubList(list1, list3);
System.out.println("Last index num3 in list1: " + num4);
```

Lớp Collections trong Java

Sử dụng phương thức unmodifiableCollection() để sửa đổi bất kỳ phần tử nào trong tập hợp, và các phần tử đó được khởi tạo bởi lớp unmodifiable

```
List<Integer> list1 = new ArrayList<>();
list1.add(10);
list1.add(20);
list1.add(30);
list1.add(40);

List<Integer> list2 = Collections.unmodifiableList(list1);
System.out.println("list2 elements: " + list2);

// list2.add(50); // throws UnsupportedOpearationException

System.out.println("list1 elements before adding 50: " + list1);
list1.add(50);
System.out.println("list1 elements after adding 50: " + list1);
```

Trong hàm main

```
Set<String> set1 = new HashSet<>();
set1.add("welcome");
set1.add("to");
set1.add("gpconder");
Set<String> set2 = Collections.unmodifiableSet(set1);
System.out.println("set2 elements: " + set2);

// set2.add("sir"); // throws UnsupportedOpearationException

Map<Integer, String> map1 = new HashMap<>();
map1.put(1, "one");
map1.put(2, "two");
map1.put(3, "three");
Map<Integer, String> map2 = Collections.unmodifiableMap(map1);
System.out.println("map2 elements: " + map2);

// map2.put(4, "four"); // throws UnsupportedOpearationException
```

Lớp Collections trong Java

Sử dụng phương thức synchronizedCollection() để sử dụng các phương thức của Collection trong môi trường đa luồng (multi-thread)

```
public static void main(String args[]) {
    List<Integer> list1 = new ArrayList<>();
    list1.add(10);
    list1.add(20);
    list1.add(30);
    list1.add(40);

    List<Integer> list2 = Collections.synchronizedList(list1);
    System.out.println("list2 elements: " + list2);

    HashSet<String> set1 = new HashSet<String>();
    set1.add("welcome");
    set1.add("to");
    set1.add("gpconder");
    Set<String> set2 = Collections.synchronizedSet(set1);
    System.out.println("set2 elements: " + set2);

    Map<Integer, String> map1 = new HashMap<>();
    map1.put(1, "one");
    map1.put(2, "two");
    map1.put(3, "three");
    Map<Integer, String> map2 = Collections.synchronizedMap(map1);
    System.out.println("map2 elements: " + map2);
}
```

Lớp Collections trong Java

Sử dụng phương thức checkedCollection() để sử dụng Type-safe Collection (tương tự Generic)

```
public static void main(String args[]) {
    List list1 = new ArrayList();
    list1.add(10);
    list1.add(20);
    list1.add(30);
    list1.add(40);

    List list2 = Collections.checkedList(list1, Integer.class);
    System.out.println("list2 elements: " + list2);

    list1.add("hello");
    // list2.add("hello"); // throws ClassCastException

    HashSet set1 = new HashSet();
    set1.add("welcome");
    set1.add("to");
    set1.add("gpconder");
    Set set2 = Collections.checkedSet(set1, String.class);
    System.out.println("set2 elements: " + set2);

    set1.add(10);
    // secondSet.add(10); // throws ClassCastException

    Map map1 = new HashMap();
    map1.put(1, "one");
    map1.put(2, "two");
    map1.put(3, "three");
    Map map2 = Collections.checkedMap(map1, Integer.class, String.class);
    System.out.println("map2 elements: " + map2);

    // map2.put("4", "four"); // throws ClassCastException
}
```

Lớp Collections trong Java

Sử dụng phương thức singletonList() để đảm bảo một đối tượng chỉ có một phần tử

```
public class CollectionsExample18 {
    public static void main(String args[]) {
        List<Integer> list = Collections.singletonList(new Integer(10));
        System.out.println("list elements: " + list);

        // list.add(20); // throws UnsupportedOperationException

        Set<String> set = Collections.singleton("Welcome to javacoder");
        System.out.println("set elements: " + set);

        // set.add("world"); // throws UnsupportedOperationException

        Map<Integer, String> map = Collections.singletonMap(1, "one");
        System.out.println("map elements: " + map);

        // map1.put(2, "two"); // throws UnsupportedOperationException
    }
}
```

Lớp Collections trong Java

Sử dụng phương thức list(Enumeration) để chuyển Enumeration sang ArrayList

```
public static void main(String args[]) {
    Vector<Integer> vect = new Vector<>();
    vect.addElement(10);
    vect.addElement(30);
    vect.add(50);
    vect.add(20);
    System.out.println("Elements of Vector: " + vect);

    Enumeration<Integer> e = vect.elements();
    ArrayList<Integer> myList = Collections.list(e);
    System.out.println("Elements of ArrayList: " + myList);
}
```

Lớp Collections trong Java

Sử dụng phương thức enumeration() để có thể duyệt các phần tử của Collection thông qua đối tượng Enumeration

```
public static void main(String args[]) {
    List<Integer> list1 = new ArrayList<>();
    list1.add(10);
    list1.add(20);
    list1.add(30);
    list1.add(40);
    System.out.println("list1 elements: " + list1);

    Enumeration<Integer> e = Collections.enumeration(list1);

    System.out.print("list1 elements using Enumeration: ");
    while (e.hasMoreElements()) {
        Object obj = e.nextElement();
        System.out.print(obj + " ");
    }
}
```

Tổng kết nội dung bài học

- Lớp Properties trong Java
- Lớp Arrays trong Java
- Lớp Collections trong Java