CÔNG NGHỆ JAVA

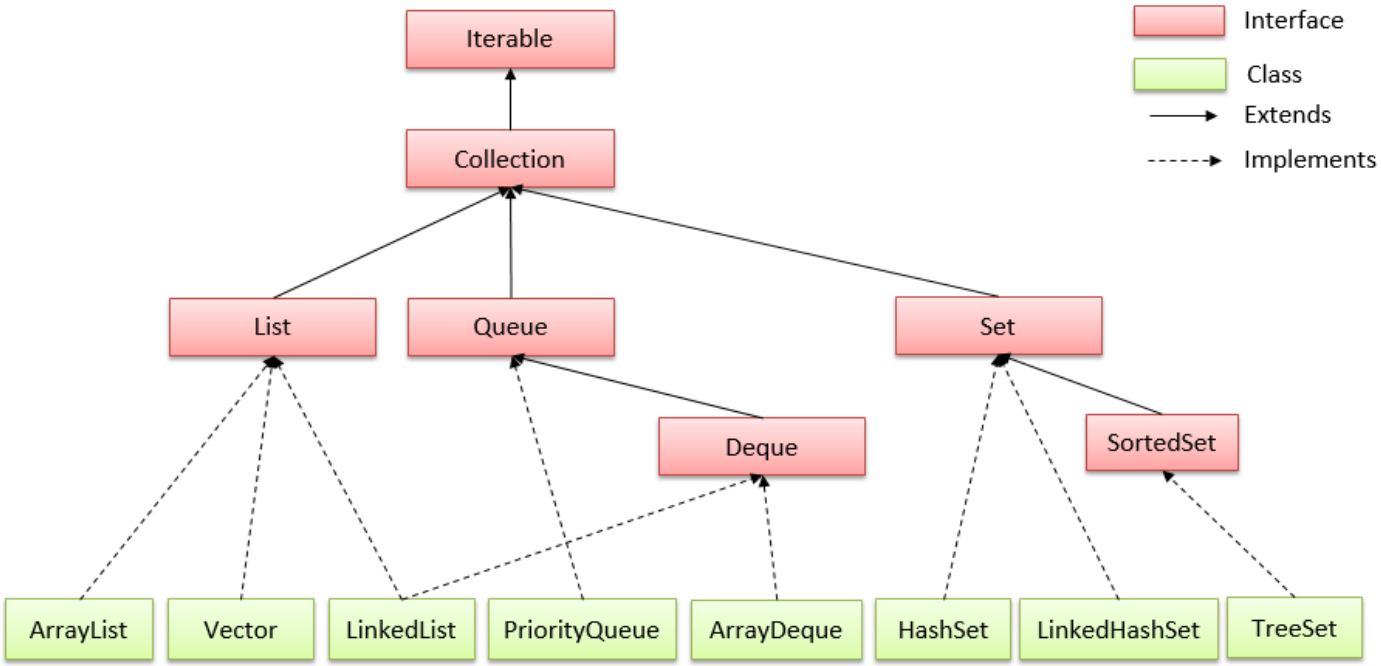
Collection

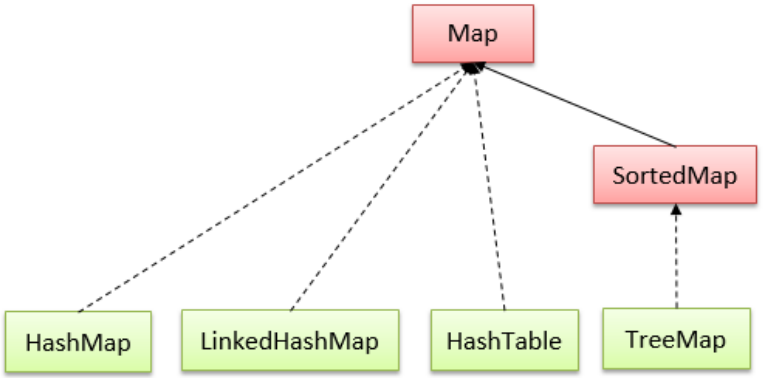
Nguyễn Hữu Thể

Collection

- Collection là một đối tượng mà nó nhóm các đối tượng khác thành phần tử và cung cấp các phương thức cơ bản để thêm, xóa, lấy, duyệt các phần tử…
  - Phần tử của Collection không được phép là kiểu nguyên thủy
- Collections Framework thống nhất cách thức sử dụng các collection, gồm 3 thành phần chính:
  - Interface
  - Lớp triển khai
  - Thuật toán
- Sử dụng đối tượng Iterator để duyệt qua tất cả các phần tử của collection.
- Ưu điểm: tiện dụng, hiệu năng cao

Hệ thống cấp bậc Collection





Collection

- **Iterable** interface
  - Chứa dữ liệu thành viên Iterator interface
- **Iterator** interface
  - Cung cấp phương tiện để lặp đi lặp lại các thành phần từ đầu đến cuối của một collection.
- Các phương thức của **Iterator** interface

Phương thức	Mô tả
public boolean hasNext()	Trả về true nếu iterator còn phần tử kế tiếp phần tử đang duyệt.
public object next()	Trả về phần tử hiện tại và di chuyển con trỏ trỏ tới phần tử tiếp theo.
public void remove()	Loại bỏ phần tử cuối được trả về bởi Iterator. Nó hiếm khi được sử dụng.

Các phương thức của Interface Collection

Phương thức	Mô tả
public boolean add(Object element)	Chèn một phần tử vào collection.
public boolean addAll(Collection c)	Chèn các phần tử collection được chỉ định vào collection gọi phương thức này.
public boolean remove(Object element)	Xóa phần tử từ collection.
public boolean removeAll(Collection c)	Xóa tất cả các phần tử của collection được chỉ định từ collection gọi phương thức này.
public boolean retainAll(Collection c)	Xóa tất cả các thành phần từ collection
public int size()	Trả lại tổng số các phần tử trong collection.
public void clear()	Loại bỏ tổng số của phần tử khỏi collection.
public boolean contains(Object element)	Tìm kiếm phần tử.
public boolean containsAll(Collection c)	Tìm kiếm collection được chỉ định trong collection.
public Iterator iterator()	Trả về một iterator.
public Object[] toArray()	Chuyển đổi collection thành mảng (array).
public boolean isEmpty()	Kiểm tra nếu collection trống.
public boolean equals(Object element)	So sánh 2 collection.
public int hashCode()	Trả về số hashcode của collection.

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

public class CollectionExample {
    public static void main(String[] args) {
        List<String> arrayList = new ArrayList<String>();
        arrayList.add("Python");
        arrayList.add("Java");
        arrayList.add("C++");
        System.out.println("Các phần tử của ArrayList: " + arrayList);

        List<String> linkedList = new LinkedList<String>();
        linkedList.add("Python");
        linkedList.add("Java");
        linkedList.add("C++");
        System.out.println("Các phần tử của LinkedList: " + linkedList);
    }
}
```

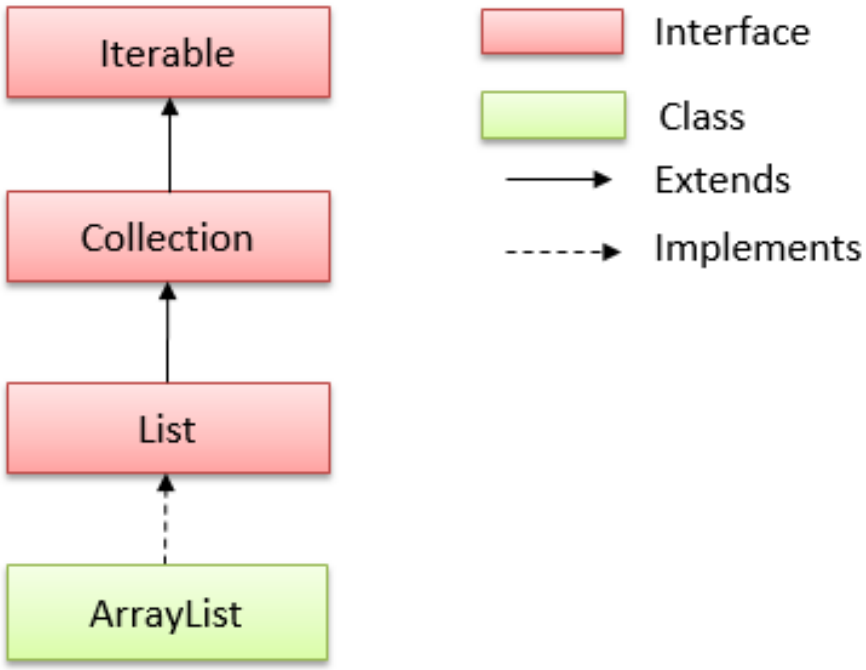
Các phần tử của ArrayList: [Python, Java, C++]
Các phần tử của LinkedList: [Python, Java, C++]

```java
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

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

        // new TreeSet() sẽ sắp xếp các phần tử
        Set<String> hashSet = new HashSet<String>();
        hashSet.add("Python");
        hashSet.add("Java");
        hashSet.add("Java");
        hashSet.add("C++");
        System.out.println("Các phần tử của Set: " + hashSet);

        // new TreeMap() sẽ sắp xếp các phần tử dự vào key của chúng
        Map<String, String> hashMap = new HashMap<String, String>();
        hashMap.put("Language2", "Java");
        hashMap.put("Language1", ".Net");
        System.out.println("Các phần tử của Map: " + hashMap);
    }
}
```

Các phần tử của Set: [Java, C++, Python]
Các phần tử của Map: {Language1=.Net, Language2=Java}

ArrayList

– **Lớp ArrayList:** mảng động.
  - Kế thừa lớp AbstractList và impliments interface List.



- ArrayList có thể chứa các phần tử trùng lặp.
- ArrayList duy trì thứ tự của phần tử được thêm vào.
- ArrayList là không đồng bộ (non-synchronized).
- ArrayList cho phép truy cập ngẫu nhiên vì nó lưu dữ liệu theo chỉ mục.

ArrayList - constructor

Constructor	Mô tả
ArrayList()	Khởi tạo một danh sách mảng trống.
ArrayList(Collection c)	Xây dựng một danh sách mảng được khởi tạo với các phần tử của collection c.
ArrayList(int capacity)	Xây dựng một danh sách mảng mà có dung lượng ban đầu được chỉ định.

ArrayList - phương thức

Phương thức	Mô tả
boolean add(Object o)	Nối thêm phần tử được chỉ định vào cuối một danh sách.
void add(int index, Object element)	Chèn các phần tử được chỉ định tại các chỉ số vị trí
boolean addAll(Collection c)	Nối tất cả các phần tử trong collection được chỉ định vào cuối của danh sách này, theo thứ tự chúng được trả về bởi bộ lặp iterator.
boolean addAll(int index, Collection c)	Chèn tất cả các phần tử trong collection được chỉ định vào danh sách này, bắt đầu từ vị trí đã chỉ định.
void retainAll(Collection c)	Xóa những phần tử không thuộc collection c và không thuộc list hiện tại
void removeAll(Collection c)	Xóa những phần tử thuộc collection c và thuộc list hiện tại khỏi list hiện tại.
int indexOf(Object o)	Trả về chỉ mục trong danh sách với sự xuất hiện đầu tiên của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa phần tử này.
int lastIndexOf(Object o)	Trả về chỉ mục trong danh sách với sự xuất hiện cuối cùng của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa phần tử này.
Object[] toArray()	Trả về một mảng chứa tất cả các phần tử trong danh sách theo thứ tự.
Object[] toArray(Object[] a)	Trả về một mảng chứa tất cả các phần tử trong danh sách theo thứ tự.
Object clone()	Trả về một bản sao của ArrayList.
void clear()	Xóa tất cả các phần tử từ danh sách này.
void trimToSize()	Cắt dung lượng của ArrayList là kích thước danh sách hiện tại.

Ví dụ: ArrayList có kiểu generic là String

```
import java.util.ArrayList;
import java.util.Iterator;

public class ArrayListExample1 {
	public static void main(String args[]) {
		ArrayList<String> list = new ArrayList<String>();
		list.add("Java");
		list.add("C++");
		list.add("PHP");
		// Show list through Iterator
		Iterator<String> itr = list.iterator();
		while (itr.hasNext()) {
			System.out.print(itr.next() + ", ");
		}
		// Show list through for-each
		System.out.println();
		for (String obj : list) {
			System.out.print(obj + ", ");
		}
		// Show list through index
		System.out.println();
		int size = list.size();
		for (int i = 0; i < size; i++) {
			System.out.print(list.get(i) + ", ");
		}
	}
}
```

Output:

```
Java, C++, PHP,
Java, C++, PHP,
Java, C++, PHP,
```

VD: ArrayList có kiểu generic là đối tượng do người dùng định nghĩa

```java
class Student {
	private String name;
	private int age;
	public Student(String name, int age) {
		super();
		this.name = name;
		this.age = age;
	}
	@Override
	public String toString() {
		return "Student@[name=" +
		name + ", age=" + age + "]";
	}
}
```

```java
import java.util.ArrayList;

public class ArrayListExample3 {
	public static void main(String[] args) {
		// Create listStudent
		ArrayList<Student> listStudent = new
		ArrayList<Student>();

		// Create students
		Student student1 = new Student("A", 18);
		Student student2 = new Student("B", 20);
		Student student3 = new Student("C", 19);

		// Add objects to listStudent
		listStudent.add(student1);
		listStudent.add(student2);
		listStudent.add(student3);

		// Show listStudent
		for (Student student : listStudent) {
			System.out.println(student.toString()
			);
		}
	}
}
```

Output:

```
Student@[name=A, age=18]
Student@[name=B, age=20]
Student@[name=C, age=19]
```

VD: addAll(), removeAll(), retainAll()

```java
import java.util.ArrayList;
public class ArrayListExample4 {
    public static void main(String[] args) {
        ArrayList<String> list = new
        ArrayList<String>();
        list.add("Java");
        list.add("C++");
        list.add("Python");
        // addAll()
        ArrayList<String> listA = new
        ArrayList<String>();
        listA.addAll(list);
        System.out.print("listA:");
        showList(listA);
        // retainAll() listA
        ArrayList<String> listB = new
        ArrayList<String>();
        listB.add("Java");
        listA.retainAll(listB);
        System.out.print("listA:");
        showList(listA);
        // removeAll() listA
        list.removeAll(listA);
        System.out.print("listA:");
        showList(list);
    }
```

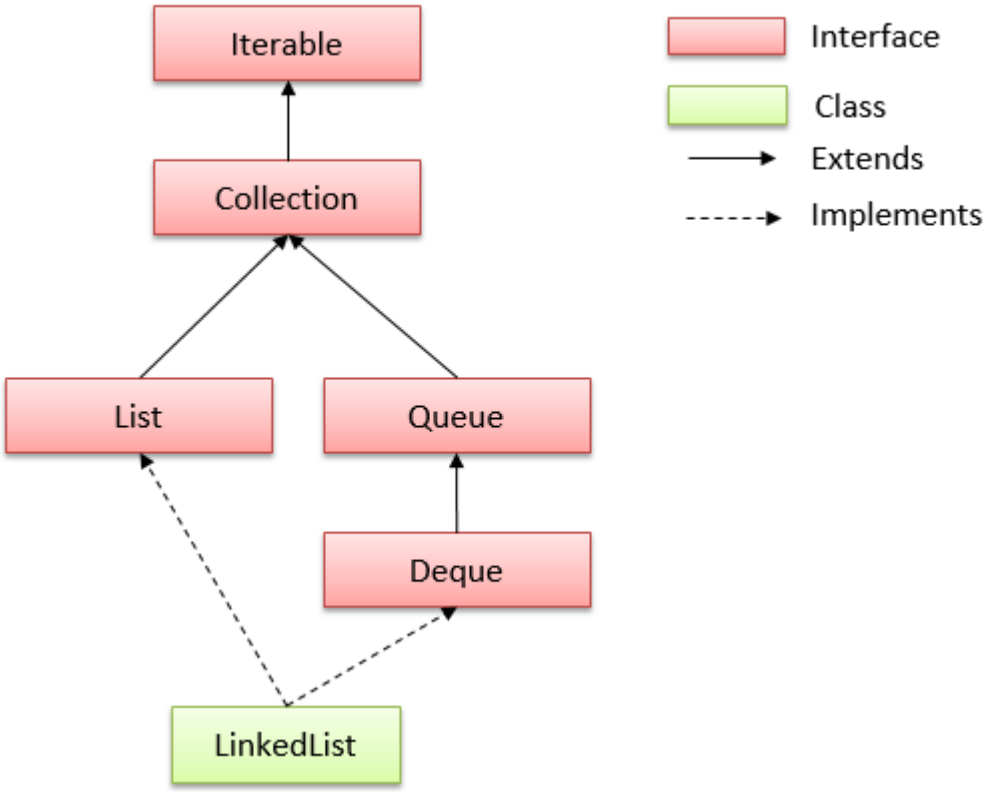
```java
    public static void
    showList(ArrayList<String> list) {
        for (String obj : list) {
            System.out.print("\t" + obj + ", ");
        }
        System.out.println();
    }
}
```

Output:

```
listA:Java, C++, Python,
listA:Java,
listA:C++, Python,
```

LinkedList

– Lớp LinkedList sử dụng cấu trúc danh sách liên kết Doubly (Doubly Linked List) để lưu trữ các phần tử.



- LinkedList có thể chứa các phần tử trùng lặp.
- LinkedList duy trì thứ tự của phần tử được thêm vào.
- LinkedList là không đồng bộ (non-synchronized).
- Lớp LinkList, thao tác nhanh vì không cần phải dịch chuyển nếu bất kỳ phần tử nào bị xoá khỏi danh sách.
- Lớp ArrayList có thể được sử dụng như list (danh sách), stack (ngăn xếp) hoặc queue (hàng đợi).

LinkedList - constructor

Constructor	Mô tả
LinkedList()	Xây dựng một danh sách trống.
LinkedList(Collection c)	Xây dựng một danh sách chứa các phần tử của collection được chỉ định, theo thứ tự chúng được trả về bởi iterator của collection.

LinkedList - constructor

Phương thức	Mô tả
boolean add(Object o)	Nối thêm phần tử được chỉ định vào cuối một danh sách.
void add(int index, Object element)	Chèn các phần tử được chỉ định tại các chỉ số vị trí quy định trong một danh sách.
void addFirst(Object o)	Chèn phần tử được chỉ định vào đầu danh sách.
void addLast(Object o)	Chèn phần tử được chỉ định vào cuối danh sách.
int size()	Trả lại số lượng các phần tử trong một danh sách
boolean contains(Object o)	Trả về *true* nếu danh sách có chứa một phần tử được chỉ định.
boolean remove(Object o)	Xóa phần tử được chỉ định đầu tiên trong một danh sách.
Object getFirst()	Trả về phần tử đầu tiên trong một danh sách.
Object getLast()	Trả lại phần tử cuối cùng trong một danh sách.
int indexOf(Object o)	Trả về chỉ mục trong một danh sách với sự xuất hiện đầu tiên của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa bất kỳ phần tử nào.
int lastIndexOf(Object o)	Trả lại chỉ mục trong danh sách với sự xuất hiện cuối cùng của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa bất kỳ phần tử nào.

LinkedList - constructor

```java
import java.util.Iterator;
import java.util.LinkedList;

public class LinkedListExample1 {
	public static void main(String[] args) {
		LinkedList<String> linkedList = new
		LinkedList<String>();

		linkedList.add("Java");
		linkedList.add("C++");
		linkedList.add("PHP");
		linkedList.add("Java");

		// Show list through Iterator
		Iterator<String> itr =
		linkedList.iterator();
		while (itr.hasNext()) {
			System.out.print(itr.next() + ", ");
		}
```

```java
		// add first
		linkedList.addFirst("Python");

		// Show list through for-each
		System.out.println();
		for (String obj : linkedList) {
			System.out.print(obj + ", ");
		}

		// add last
		linkedList.addLast("Ruby");

		// Show list through index
		System.out.println();
		int size = linkedList.size();
		for (int i = 0; i < size; i++) {
			System.out.print(linkedList.get(i) + ", ");
		}
	}
}
```

Output:

Java, C++, PHP, Java,
Python, Java, C++, PHP, Java,
Python, Java, C++, PHP, Java, Ruby,

ArrayList và LinkedList

- Giống nhau: implements từ List, duy trì thứ tự, non-synchronized
- Khác nhau:

ArrayList	LinkedList
1) ArrayList nội bộ sử dụng **mảng động** để lưu trữ các phần tử.	LinkedList nội bộ sử dụng **danh sách liên kết doubly** để lưu trữ các phần tử.
2) Thao tác với ArrayList là **chậm** bởi vì nó sử dụng nội bộ mảng. Nếu bất kỳ phần tử nào được xoá khỏi mảng, tất cả các bit được chuyển trong bộ nhớ.	Thao tác với LinkedList là **nhanh hơn** so với ArrayList bởi vì nó sử dụng danh sách liên kết doubly do đó không cần chuyển đổi bit nào trong bộ nhớ.
3) Lớp ArrayListchỉ có thể **hoạt động như một list** vì nó chỉ implements interface List.	Lớp LinkedList có thể **hoạt động như một list và queue(hàng đợi)** vì nó implements các interface List và Deque.
4) ArrayList là **tốt hơn trong việc lưu trữ và truy cập** dữ liệu.	LinkedList là **tốt hơn trong việc thao tác** dữ liệu.

TÀI LIỆU THAM KHẢO

- www.tutorialspoint.com/java
- viettuts/java-collection