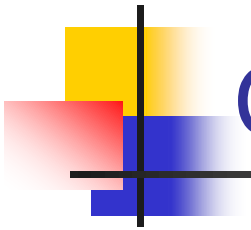
CÔNG NGHỆ JAVA

Trường: Đại học Giao thông vận Tải

Khoa: Công nghệ thông tin



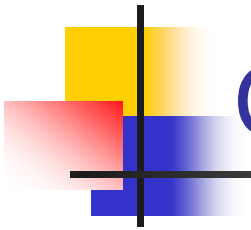
CÔNG NGHỆ JAVA

Chương 5:

Làm việc với Collection trong Java

Nội dung

- Nội dung
  - Tổng quan về Collection trong Java
    - Khái niệm chung về Collections
    - Kiến trúc Collections Framework trong Java
  - Một số Collection
    - List với ArrayList/Vector
    - Set với HashSet
    - Hashcode và equals
    - Queue với PriorityQueue
    - …
  - Làm việc với Map
    - Map
    - HashMap



CÔNG NGHỆ JAVA

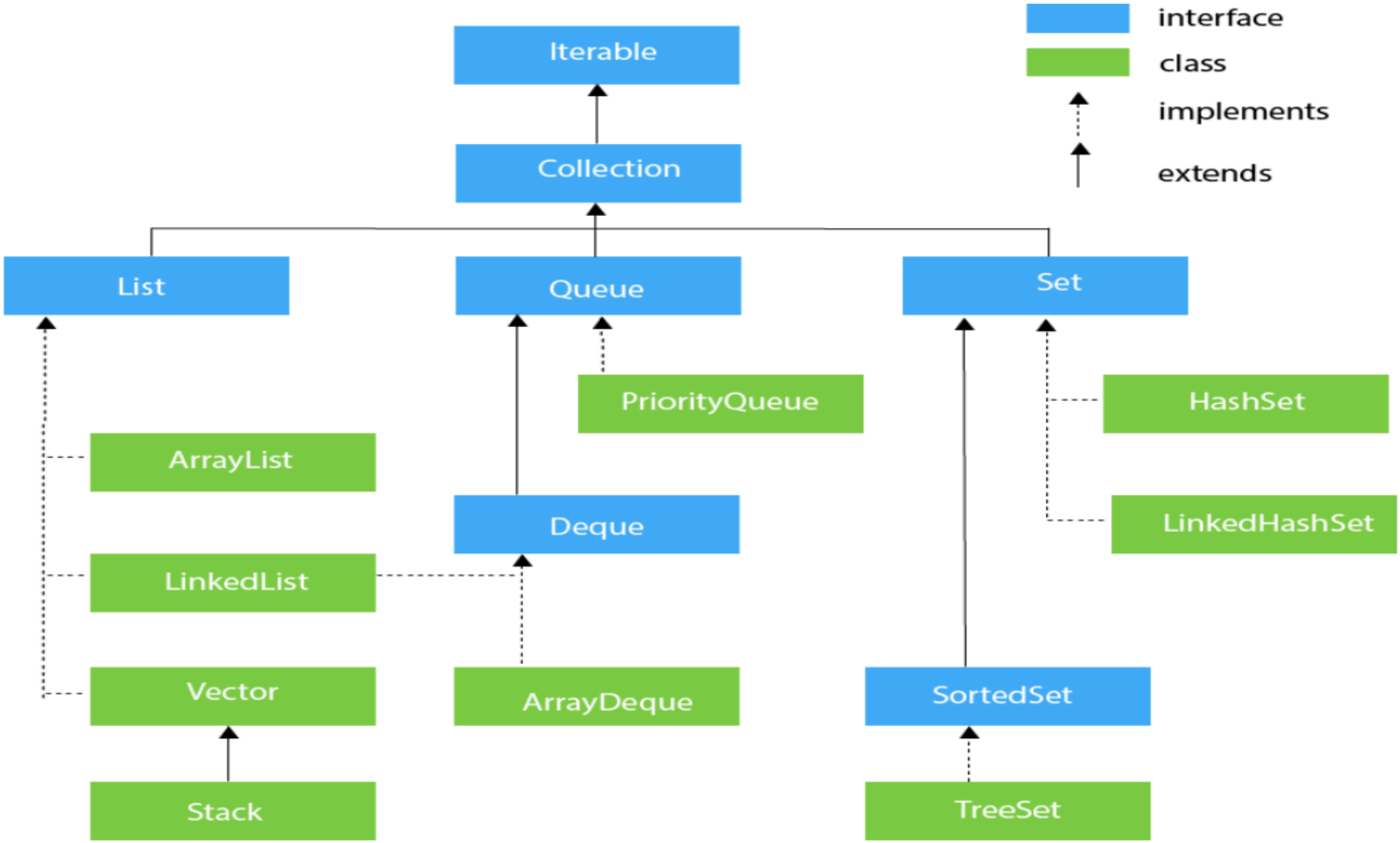
Chương 5:

1. Java Collections Framework

Khái niệm về Collection

- **Collections** trong Java là một Framework cung cấp một kiến trúc để lưu trữ và thao tác tới tập hợp các đối tượng trong cùng một họ dữ liệu.
- Java Collections Framework cung cấp các khung hoạt động cho phép chúng ta thực hiện các thao tác trên một họ đối tượng như tìm kiếm, sắp xếp, chèn, xóa, ...
- Toàn bộ kiến trúc của Java Collections Framework dựa trên *<<Interface>>* Collection để từ đó xây dựng ra các *<<Interface>>* và các *<<class>>* ứng với các cấu trúc dữ liệu khác nhau.
- Bắt đầu từ phiên bản Pre-JDK 1.2, được thiết kế chuẩn từ bản JDK 1.2 và được cập nhật với phiên bản JDK 1.5 hỗ trợ generic, Collections là một thư viện được xử dụng nhiều trong lập trình Java.

Kiến trúc của Collections



Interface – Iterable/Iterator

- Iterable interface

Phương thức	Mô tả
public void forEach(Consumer<? super T> action)	Cho phép duyệt tập các phần tử theo vòng lặp cho đến khi hết phần tử hoặc có lỗi.
public Iterator<T> iterator()	Cho phép chuyển đổi sang dạng Iterator để duyệt tập các phần tử.

- 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 tới phần tử tiếp theo.
public void remove()	Loại bỏ phần tử cuối được trả về bởi Iterator.

Interface – Collection

- Collection interface

Phương thức	Mô tả
public boolean add(Object element)	Thêm một phần tử vào collection.
public boolean addAll(Collection c)	Thêm các phần tử collection được chỉ định vào collection gọi phương thức này.
public boolean remove(Object element)	Xóa phần tử ra khỏi collection.
public boolean removeAll(Collection c)	Xóa tất cả các phần tử từ collection được chỉ định ra khỏi collection gọi phương thức này.
public boolean retainAll(Collection c)	Xóa tất cả các thành phần từ collection gọi phương thức này ngoại trừ collection được chỉ định.
public int size()	Trả về tổng số các phần tử trong collection.
public void clear()	Xóa tất cả các phần tử trong Collection, sau khi thực hiện phương thức này, Collection sẽ rỗng (empty)
public boolean contains(Object element)	Kiểm tra một phần tử có nằm trong Collection không
public boolean containsAll(Collection c)	Kiểm tra một Collection có chứa tất cả các phần tử của một Collection khác
public Iterator iterator()	Trả về một iterator.
public Object[] toArray()	Chuyển đổi collection thành mảng (array).
public boolean isEmpty()	Kiểm tra Collection có rỗng hay không.
public boolean equals(Object element)	So sánh 2 collection.
public int hashCode()	Trả về số hashcode của collection.

Cơ chế duyệt một Collection

- **Collection** trong Java cung cấp 3 cách để có thể duyệt các phần tử trong tập hợp:
  - Thông qua Iterator
  - forEach
  - for với biến tạm.

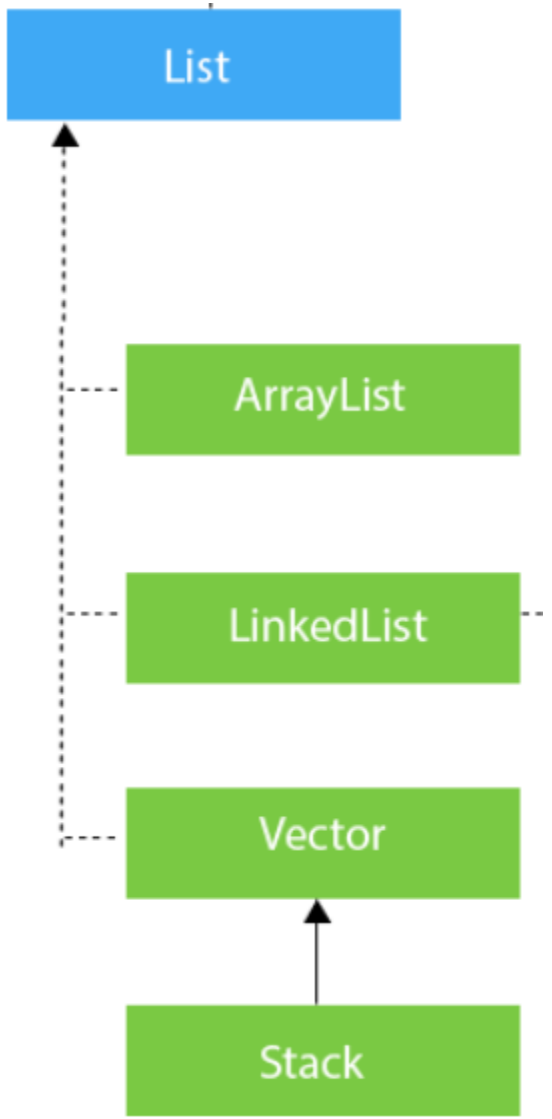
```
public class PerformListExample {
    public static void main(String[] args) {
        List<String> myList = new ArrayList<String>();
        myList.add("Hello");
        myList.add("How are you?");
        myList.add("I'm fine.");

        // Iterator
        Iterator<String> iter = myList.iterator();
        while(iter.hasNext()){
            System.out.println(iter.next());
        }
        // For each
        for (String value : myList) {
            System.out.println(value);
        }
        myList.forEach(value->System.out.println(value));
        // For
        for (int i=0; i<myList.size(); i++){
            System.out.println(myList.get(i));
        }
    }
}
```

Chương 5:

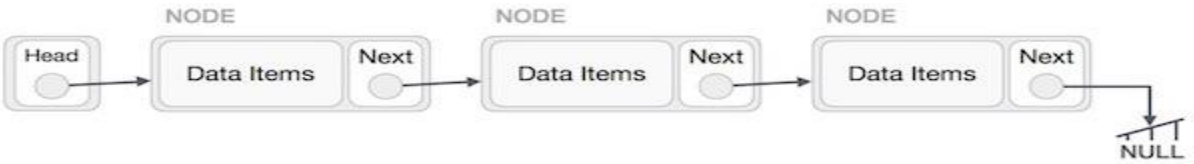
2. Một số Collection trong Java

List với ArrayList/LinkedList

- ***List***: là một interface cho phép chúng ta biểu diễn một tập các phần tử có thứ tự.
- List cho phép chứa các phần tử trùng nhau.
- List cho phép truy cập vào bất kỳ phần tử nào trong danh sách.
- ***ArrayList***: cơ chế biểu diễn danh sách liên kết đơn.
- ***LinkedList***: cơ chế biểu diễn danh sách liên kết kép.
- ***Stack***: cơ chế biểu diễn danh sách dạng LIFO – Last In First Out.

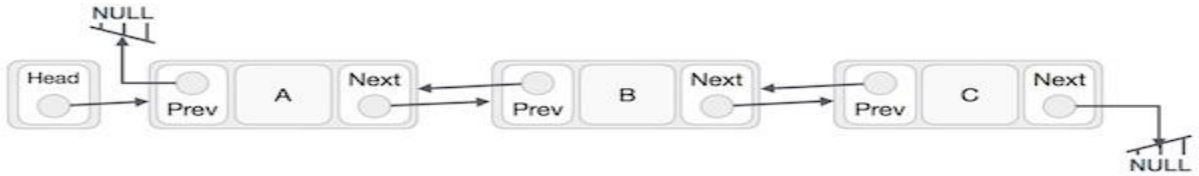


ArrayList




```
List<Point> myList = new ArrayList<Point>();
Point p1 = new Point(1, 2);
Point p2 = new Point(2, 3);
Point p3 = new Point(3, 4);

System.out.println("Add all elements in to ArrayList");
myList.add(p1);
myList.add(p2);
myList.add(p3);
for (Point point : myList) {
     System.out.println(point);
}

System.out.println("Add more element with index position");
Point p4 = new Point(4, 5);
myList.add(2, p4);
myList.forEach(point -> System.out.println(point));

System.out.println("Find an element - last position");
System.out.println(myList.lastIndexOf(p4));

System.out.println("Init an ArrayList via static list");
myList.clear();
Point[] staticList = {p1, p4, p3, p2, p1, p2};
myList.addAll(Arrays.asList(staticList));
myList.forEach(point -> System.out.println(point));

System.out.println("Sort");
myList.sort(new Point());
myList.forEach(point -> System.out.println(point));
```

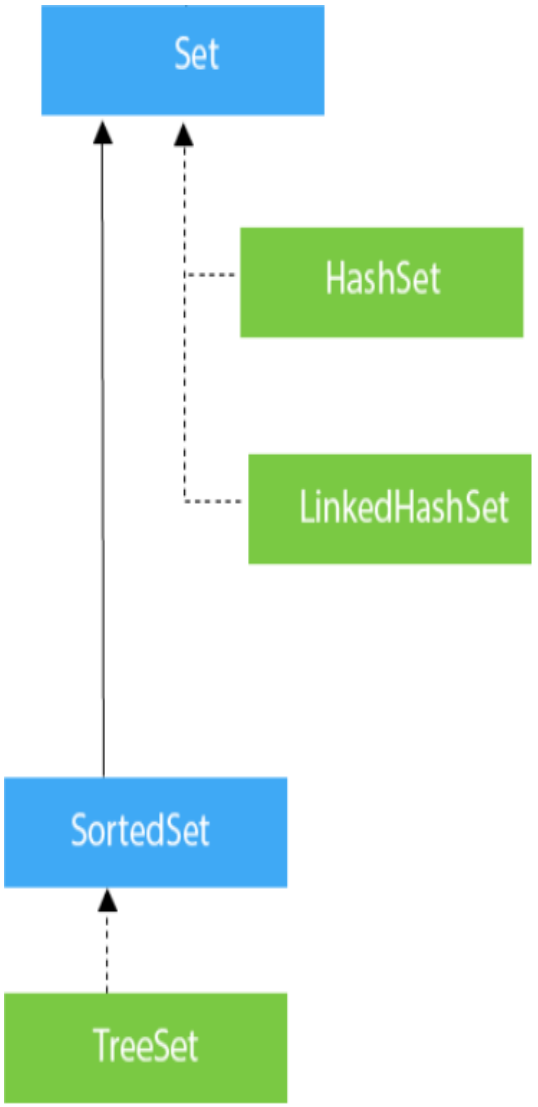
LinkedList




```
List<Point> myList = new LinkedList<Point>();
Point p1 = new Point(1, 2);
Point p2 = new Point(2, 3);
Point p3 = new Point(3, 4);

System.out.println("Add all elements in to LinkedList");
myList.add(p1);
myList.add(p2);
myList.add(p3);
for (Point point : myList) {
    System.out.println(point);
}

System.out.println("Add more element with index position");
Point p4 = new Point(4, 5);
myList.add(2, p4);
myList.forEach(point -> System.out.println(point));

System.out.println("Find an element - last position");
System.out.println(myList.lastIndexOf(p4));

System.out.println("Init an LinkedList via static list");
myList.clear();
Point[] staticList = {p1, p4, p3, p2, p1, p2};
myList.addAll(Arrays.asList(staticList));
myList.forEach(point -> System.out.println(point));

System.out.println("Sort");
myList.sort(new Point());
myList.forEach(point -> System.out.println(point));
```

Set với HashSet/TreeSet

- ***Set***: là một interface cho phép chúng ta biểu diễn một tập các phần tử không có thứ tự.
- Set không cho phép chứa các phần tử trùng nhau – dựa trên hàm băm (mỗi đối tượng đều có mã băm được định nghĩa bởi hashCode.
- Set cho phép thao tác các phần tử null trong danh sách.
- Chú ý: nếu muốn sắp xếp các phần tử theo thứ tự thì sử dụng ***TreeSet*** (dựa trên cơ chế hashCode và equals – cấu trúc cây nhị phân).

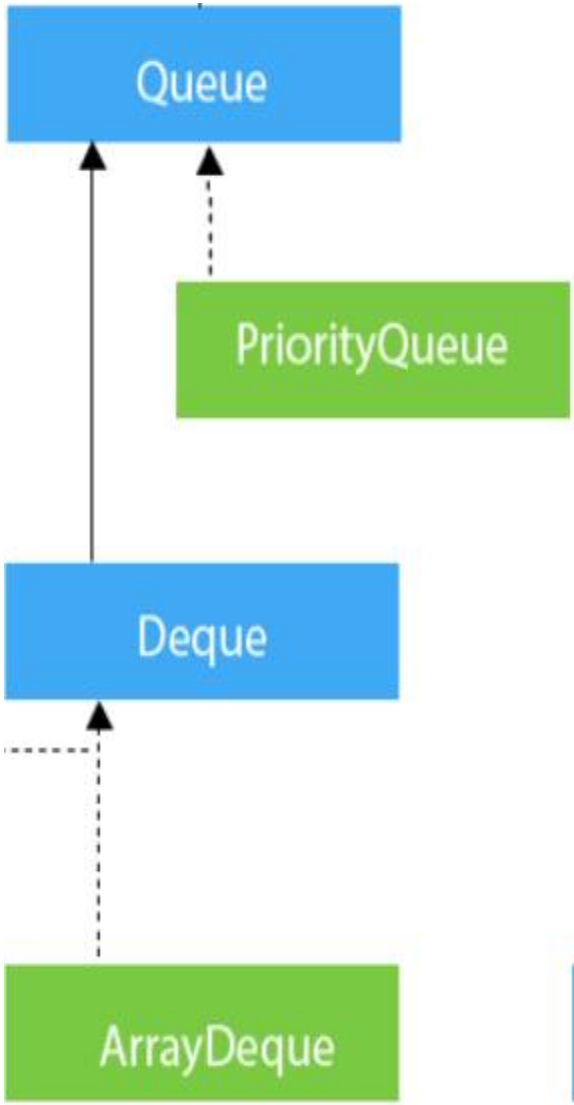


HashSet

```
Set<Point> mySet = new HashSet<Point>();
Point p1 = new Point(1, 2);
Point p2 = new Point(2, 3);
Point p3 = new Point(3, 4);
Point p4 = new Point(4, 5);

System.out.println("Add all elements in to HashSet");
mySet.add(p1);
mySet.add(p2);
mySet.add(p3);
Iterator<Point> iter = mySet.iterator();
while(iter.hasNext()){
    System.out.println(iter.next());
}

System.out.println("Find an element in HashSet");
System.out.println(mySet.contains(p2));

System.out.println("Init an HashSet via static list");
mySet.clear();
Point[] staticList = {p1, p4, p3, p2, p1, p2};
mySet.addAll(Arrays.asList(staticList));
mySet.forEach(point -> System.out.println(point));

System.out.println("Modify each element in HashSet");
p1.setX(15);
p1.setY(30);
mySet.forEach(point -> System.out.println(point));
```

Queue / Deque

- ***Queue*** và ***Deque***: là các interface cho phép chúng ta biểu diễn một tập các phần tử theo cơ chế FIFO và LIFO.
- Các phần tử trong ***Queue/Deque*** phải được so sánh với nhau do đó, các phần tử của nó cần cài đặt <<interface>> Comparable.
- ***Queue/Deque*** hỗ trợ các thao tác làm việc với FIFO/LIFO ➔ không nên truy cập theo chỉ số.
- Chú ý: không sử dụng Queue/Deque để sắp xếp các phần tử trong danh sách.



Hashcode và equals

- Khi sử dụng các Collection, để nhận được các hành vi mong muốn cần ghi đè các phương thức equals() và hashCode() trong các lớp của các phần tử được thêm vào Collection.
- Collection dựa trên bảng băm xác định một phần tử bằng cách gọi phương thức **hashCode()** và **equals()** của nó.
- Khi ghi đè các phương thức này chúng ta phải tuân theo các quy tắc sau:
  - Khi equals được ghi đè thì hashCode cũng phải được ghi đè
  - 2 đối tượng bằng nhau thì phải có trùng hash code
  - Hai đối tượng có hash code khác nhau thì phải khác nhau

Hashcode và equals

```
package list_demo;

import java.util.ArrayList;
import java.util.List;

public class Person {
	private String id;
	private String name;

	public Person(String id, String name) {
		super();
		this.id = id;
		this.name = name;
	}


	public static void main(String[] args) {
		List<Person> allPeople = new
		ArrayList<Person>();
		Person p1 = new Person("1", "An");

		allPeople.add(p1);

		Person p2 = new Person("1", "An");
		System.out.println(allPeople.contains(p2));
	}
}
```

```
false
```

- Nguyên nhân: Hàm contains duyệt qua từng phần tử trong list và so sách với tham số thông qua phương thức .equals()
- equals() được thừa kế từ lớp **Object** và mặc định sẽ so sánh tham chiếu

Hashcode và equals

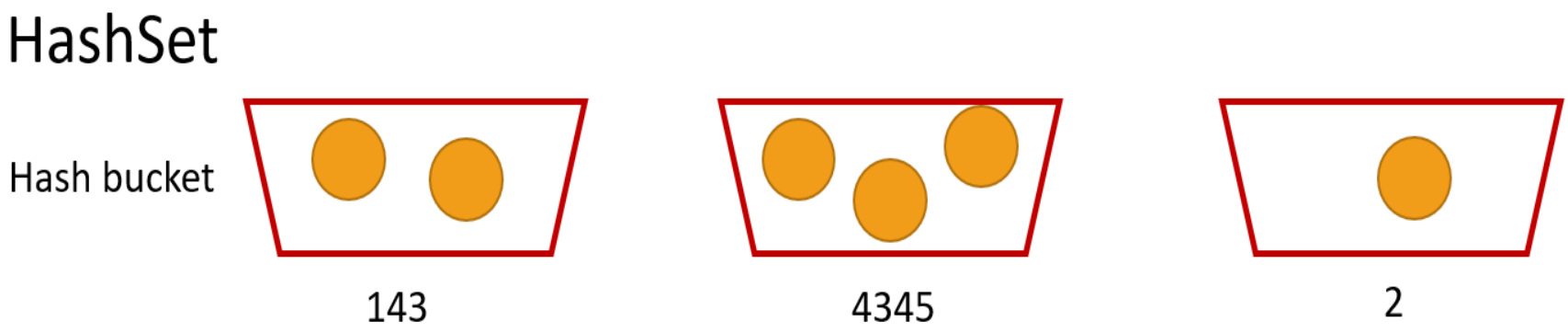
```
import java.util.ArrayList;
import java.util.List;

public class Person {
	private String id;
	private String name;

	public Person(String id, String name) {
		this.id = id;
		this.name = name;
	}

	@Override
	public boolean equals(Object obj) {
		if (obj instanceof Person) {
			Person another = (Person) obj;
			if (this.id.equals(another.id)) {
				return true;
			}
		}
		return false;
	}


	public static void main(String[] args) {
		List<Person> allPeople = new ArrayList<Person>();

		Person p1 = new Person("1", "An");
		allPeople.add(p1);

		Person p2 = new Person("1", "An");
		System.out.println(allPeople.contains(p2));
	}
}
```

```
true
```

- Nguyên nhân: **.equals()** được ghi đè

Hashcode và equals

- **hashCode()** trả về mã băm của đối tượng.
- Một số Collection dựa trên mã băm để lưu trữ các đối tượng theo nhóm. Mỗi nhóm chỉ chứa các đối tượng có mã băm giống nhau.
- Các bước để định vị đối tượng trong bảng băm:
  - Nhận giá trị mã băm của phần tử
  - Tìm nhóm tương ứng với mã băm
  - Trong nhóm tìm chính xác phần tử bằng so sánh **.equals()**

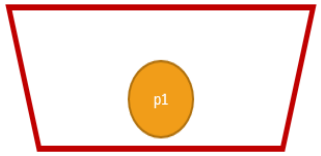


Hashcode và equals

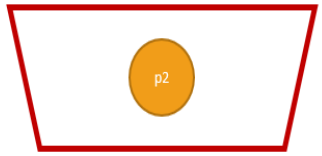
```java
import java.util.HashSet;
import java.util.Set;

public class Person {
	private String id;
	private String name;

	public Person(String id, String name) {
		this.id = id;
		this.name = name;
	}

	@Override
	public boolean equals(Object obj) {
		if (obj instanceof Person) {
			Person another = (Person) obj;
			if (this.id.equals(another.id)) {
				return true;
			}
		}
		return false;
	}


	public static void main(String[] args) {
		Set<Person> allPeople = new HashSet<Person>();

		Person p1 = new Person("1", "An");
		allPeople.add(p1);

		Person p2 = new Person("1", "An");
		allPeople.add(p2);

		System.out.println(allPeople.size());
	}
}
```

2

- Nguyên nhân: p1 và p2 có hash code khác nhau



366712642



1829164700

Hashcode và equals

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

public class Person {
	private String id;
	private String name;

	public Person(String id, String name) {
		this.id = id;
		this.name = name;
	}

	@Override
	public boolean equals(Object obj) {
		if (obj instanceof Person) {
			Person another = (Person) obj;
			if (this.id.equals(another.id)) {
				return true;
			}
		}
		return false;
	}

	public int hashCode() {
		return id.hashCode();
	}

	public static void main(String[] args) {
		Set<Person> allPeople = new HashSet<Person>();

		Person p1 = new Person("1", "An");
		allPeople.add(p1);

		Person p2 = new Person("1", "An");
		allPeople.add(p2);

		System.out.println(allPeople.size());
	}
}
```

```
1
```

- Nguyên nhân: p1 và p2 trùng hashcode và equal

Chương 5:

3. Map và làm việc với HashMap

Map - HashMap

- Map nhằm lưu trữ các cặp khóa và giá trị (***key, value***)
- HashMap lưu trữ các khóa theo hash set

```
package collection_demo;

import java.util.HashMap;
import java.util.Map;

public class MapDemo {

     public static void main(String[] args) {
          Map<String, Integer> map = new HashMap<String, Integer>();
          map.put("mystring1", 1);
          map.put("mystring2", 2);

          for(String key : map.keySet()) {
               System.out.println("key: " + key + "; value: " +
          map.get(key));
          }
     }

}
```

CÔNG NGHỆ JAVA