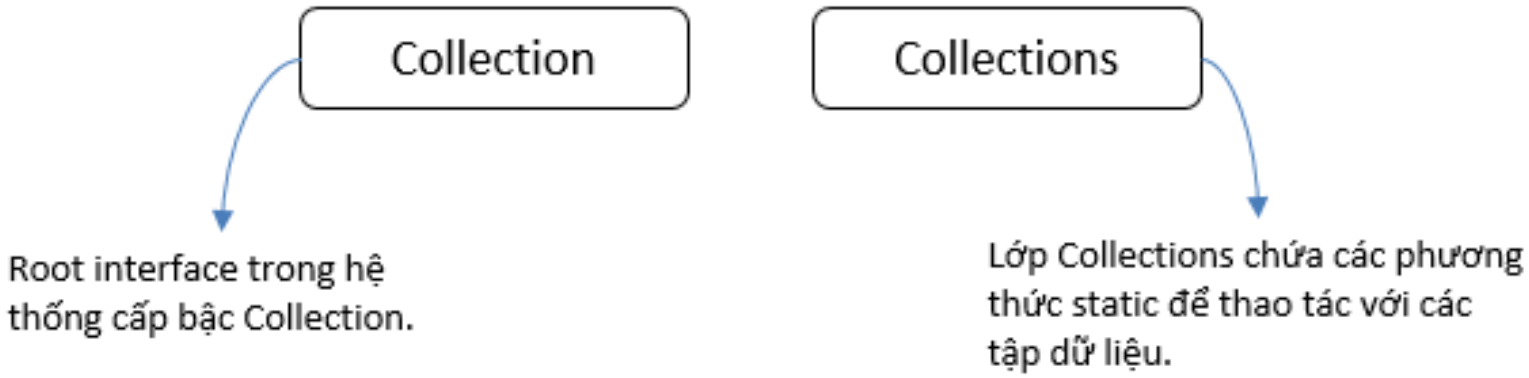
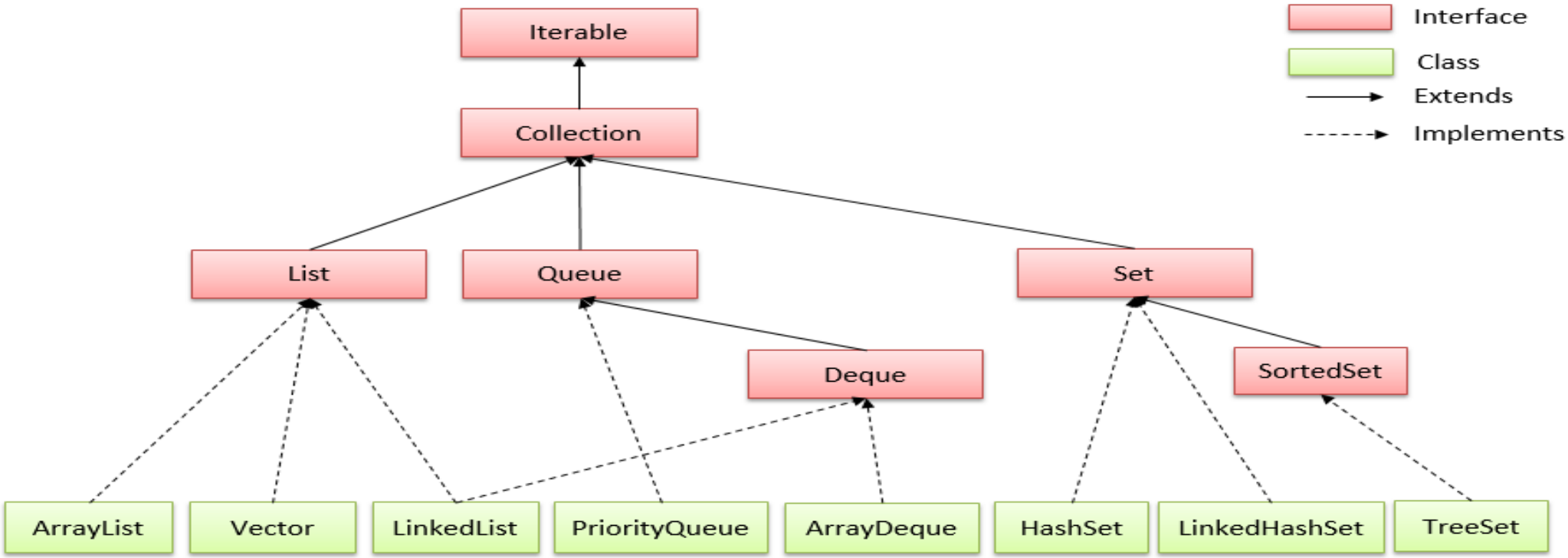
Bài 7
Lập trình với Collection

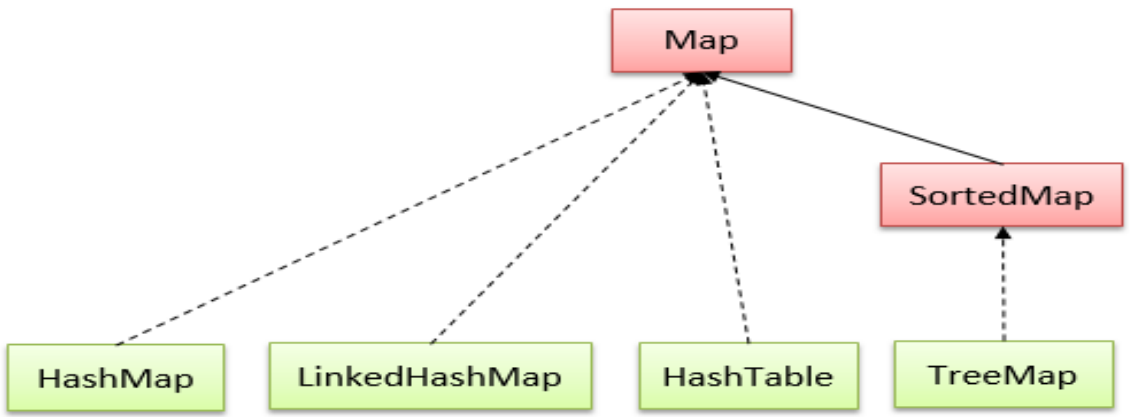
Collection vs Collections

- "Collection" và "Collections" trong java là hai khái niệm khác nhau.
- Collections trong java là một Framework cung cấp một kiến trúc để lưu trữ và thao tác tới nhóm các đối tượng. Tất cả các hoạt động mà bạn thực hiện trên một dữ liệu như tìm kiếm, phân loại, chèn, xóa,... có thể được thực hiện bởi Java Collections.
- Collection trong java là một root interface trong hệ thống cấp bậc Collection. Java Collection cung cấp nhiều interface (Set, List, Queue, Deque vv) và các lớp (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet vv).



Collections:





•**Set:** là một collection không thể chứa 2 giá trị trùng lặp. Set được sử dụng để biểu diễn các bộ, chẳng hạn như bộ tú lu khơ, thời khóa biểu của học sinh, các tiến trình đang chạy trên máy tính...
•**List:** là một collection có thứ tự (đôi khi còn được gọi là một chuỗi). List có thể chứa các phần tử trùng lặp. Thường có quyền kiểm soát chính xác vị trí các phần tử được chèn vào và có thể truy cập chúng bằng chỉ số (vị trí của chúng).
•**Queue (hàng đợi):** là một collection được sử dụng để chứa nhiều phần tử trước khi xử lý. Bên cạnh các thao tác cơ bản của collection, Queue cung cấp các thao tác bổ sung như chèn, lấy ra và kiểm tra. Queue có thể được sử dụng như là FIFO (first-in, first-out - vào trước, ra trước)
•**Deque:** là một collection được sử dụng để chứa nhiều phần tử trước khi xử lý. Ngoài các thao tác cơ bản của collection, một Deque cung cấp các thao tác bổ sung như chèn, lấy ra và kiểm tra. Deques có thể được sử dụng như là FIFO (first-in, first-out - vào trước, ra trước) và LIFO (last-in, first-out - vào sau, ra trước). Trong một Deque, tất cả các phần tử mới có thể được chèn vào, lấy ra và lấy ra ở cả hai đầu.
•**Map:** là một đối tượng ánh xạ mỗi key tương úng với một giá trị. Map không thể chứa giá trị trùng lặp. Mỗi key có thể ánh xạ đến nhiều nhất một giá trị.

Non-generic Collection và Generic Collection

Collection trong java là non-generic trước JDK 1.5. Từ JDK 1.5 là generic.

Non-generic Collection: (kiểu cũ)

```
ArrayList list = new ArrayList();
```

Generic Collection: (kiểu mới)

```
ArrayList<String> list = new ArrayList<String>();
```

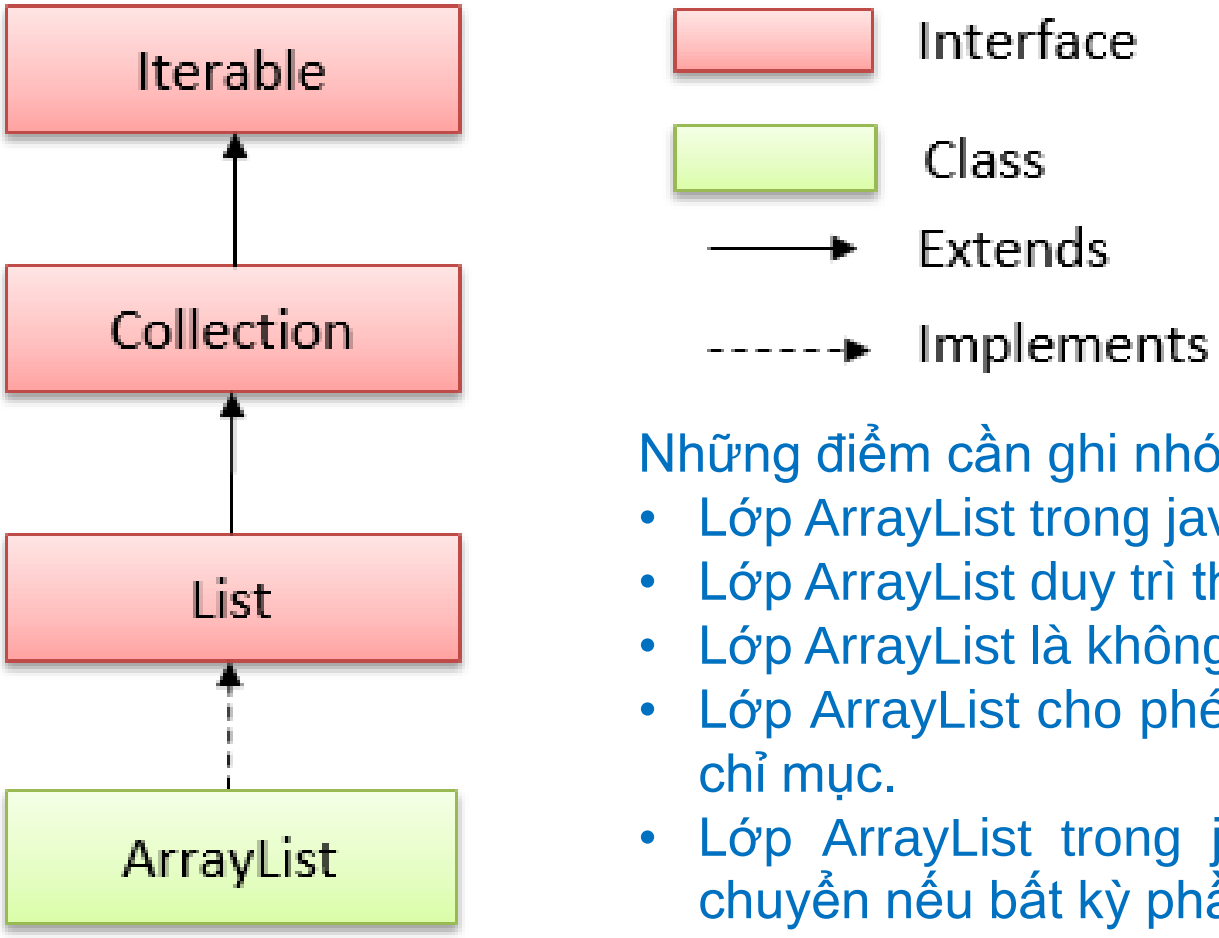
Duyệt các phần tử của collections:

Có 2 cách để duyệt các phần tử của collection trong java:

- Sử dụng Iterator interface.
- Sử dụng vòng lặp for-each.

```
import java.util.ArrayList;
import java.util.Iterator;
 public class ArrayListExample1 {
    public static void main(String args[]) {
        // Creating arraylist
        ArrayList<String> list = new ArrayList<String>();
        // Add objects to arraylist
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
        // Show list through Iterator
        Iterator<String> itr = list.iterator();
        while (itr.hasNext()) {
            System.out.print(itr.next() + ", ");
        }
    // Show list through for-each
        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) + ", ");
        } }}
```

1) Lớp ArrayList trong java



Những điểm cần ghi nhớ về ArrayList:

- Lớp ArrayList trong java có thể chứa các phần tử trùng lặp.
- Lớp ArrayList duy trì thứ tự của phần tử được thêm vào.
- Lớp ArrayList là không đồng bộ (non-synchronized).
- Lớp ArrayList cho phép truy cập ngẫu nhiên vì nó lưu dữ liệu theo chỉ mục.
- Lớp ArrayList trong java, thao tác chậm vì cần nhiều sự dịch chuyển nếu bất kỳ phần tử nào bị xoá khỏi danh sách.

Phương thức khởi tạo:

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

Phương thức của lớp ArrayList:

Phương thức	Mô tả
boolean add(Object o)	Nó được sử dụng để nối thêm phần tử được chỉ định vào cuối một danh sách.
void add(int index, Object element)	Nó được sử dụng để chèn phần tử element tại vị trí index vào danh sách.
boolean addAll(Collection c)	Nó được sử dụng để nối tất cả các phần tử trong collection c vào cuối của danh sách, theo thứ tự chúng được trả về bởi bộ lặp iterator.
boolean addAll(int index, Collection c)	Nó được sử dụng để chèn tất cả các phần tử trong collection c vào danh sách, bắt đầu từ vị trí index.
void retainAll(Collection c)	Nó được sử dụng để xóa những phần tử không thuộc collection c ra khỏi danh sách.

void removeAll(Collection c)	Nó được sử dụng để xóa những phần tử thuộc collection c ra khỏi danh sách.
int indexOf(Object o)	Nó được sử dụng để 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)	Nó được sử dụng để 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()	Nó được sử dụng để trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự.
Object[] toArray(Object[] a)	Nó được sử dụng để trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự.
Object clone()	Nó được sử dụng để trả về một bản sao của ArrayList.
void clear()	Nó được sử dụng để xóa tất cả các phần tử từ danh sách này.
void trimToSize()	Nó được sử dụng để cắt dung lượng của thể hiện ArrayList này là kích thước danh sách hiện tại.

Ví dụ ArrayList:

- **Khởi tạo một ArrayList:**

```
// import gói thư viện java.util.ArrayList
import java.util.ArrayList;
 public class KhoiTaoArrayList {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là listString
        // có kiểu là String
        ArrayList<String> listString = new ArrayList<String>();
    }
}
```

Nếu đã biết trước số lượng phần tử:

```
ArrayList<String> listString = new ArrayList<String>(20);
```

Hiển thị các phần tử có trong ArrayList:

```
import java.util.ArrayList;
 public class DuyetArrayList1 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
        // hiển thị các phần tử của list
        System.out.println("Các phần tử có trong list là: ");
        System.out.println(list);
    }
}
```

➢ Duyệt các phần tử của ArrayList - sử dụng vòng lặp for

```
import java.util.ArrayList;
 public class DuyetArrayList2 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
        // sử dụng vòng lặp for - hiển thị các phần tử của list
        System.out.println("Các phần tử có trong list là: ");
        for (int i = 0; i < list.size(); i++) {
            System.out.println(list.get(i));
        }
    }
}
```

Với kiểu for-each:

```
for (String str : list) {
            System.out.println(str);
        }
```

➢ Duyệt các phần tử của ArrayList - sử dụng Iterator:

```
import java.util.ArrayList;
import java.util.Iterator;
 public class DuyetArrayList4 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
        // sử dụng Iterator - hiển thị các phần tử của list
        Iterator<String> iterator = list.iterator();
        System.out.println("Các phần tử có trong list là: ");
        while (iterator.hasNext()) {
            System.out.println((String) iterator.next());
        }
    }
}
```

➢ Duyệt các phần tử của ArrayList - sử dụng ListIterator:

```
import java.util.ArrayList;
import java.util.ListIterator;
 public class DuyetArrayList5 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
        // sử dụng ListIterator - hiển thị các phần tử của list
        ListIterator<String> iterator = list.listIterator();
        System.out.println("Các phần tử có trong list là: ");
        while (iterator.hasNext()) {
            System.out.println((String) iterator.next());
        }
    }
}
```

➢ Các phương thức addAll(), removeAll(), retainAll() của lớp ArrayList

```
// khai báo 1 ArrayList có tên là list
    // có kiểu là String
    ArrayList<String> list = new ArrayList<String>();
    // Add objects to list
    list.add("Java");
    list.add("C++");
    list.add("PHP");
    list.add("Java");

    System.out.println("ví dụ sử dụng phương thức addAll()");
    System.out.println("--------------------------------");
    // thêm các phần tử của list vào listA
    ArrayList<String> listA = new ArrayList<String>();
    listA.addAll(list);
    System.out.print("listA:");
    showList(listA);
```

```
 System.out.println("\nví dụ sử dụng phương thức retainAll()");
        System.out.println("---------------------------------");
        // khởi tạo listB
        ArrayList<String> listB = new ArrayList<String>();
        listB.add("Java");
        // xóa những phần tử không thuộc listB khỏi listA
        listA.retainAll(listB);
        System.out.print("listA:");
        showList(listA);
        System.out.println("\nví dụ sử dụng phương thức removeAll()");
        System.out.println("---------------------------------");
        // xóa những phần tử thuộc listB  khỏi list
        list.removeAll(listB);
        System.out.print("list:");
        showList(list);
    }
    public static void showList(ArrayList<String> list) {
        // Show list through for-each
        for (String obj : list) {
            System.out.print("\t" + obj + ", ");
        }
        System.out.println();
    }
}
```

➢ Truy cập phần tử của ArrayList

```
import java.util.ArrayList;

public class TruyCapArrayList1 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");

        // truy cập phần tử có chỉ số 3 của list
        System.out.println(list.get(3));
    }
}
```

➢ Cập nhật giá trị của phần tử Arraylist

```
import java.util.ArrayList;

public class CapNhatArrayList1 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");

        System.out.println("list trước khi cập nhật: ");
        System.out.println(list);
        // cập nhật giá trị cho phần tử có chỉ số là 3 (Java)
        list.set(3, "Python");
        System.out.println("list trước khi cập nhật: ");
        System.out.println(list);
    }
}
```

➢ Xóa phần tử ArrayList

```
import java.util.ArrayList;

public class XoaArrayList1 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Python");

        System.out.println("Số phần tử của list ban đầu : " + list);
        System.out.println("Các phần tử của list ban đầu: " + list.size());
        // clear list
        list.clear();
        System.out.println("\nSố phần tử của list sau khi clear: " + list);
        System.out.println("Các phần tử của list sau khi clear: " + list.size());
    }
}
```

➢ Phương thức remove()

```
import java.util.ArrayList;
public class XoaArrayList1 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Python");
        System.out.println("Số phần tử của list ban đầu : " + list);
        System.out.println("Các phần tử của list ban đầu: " + list.size());
        // remove phần tử có chỉ số index = 1 khỏi list
        list.remove(1);
        System.out.println("\nSố phần tử của list sau khi remove phan tu co index = 1: "       + list);
        System.out.println("Các phần tử của list sau khi remove phan tu co index = 1: "    + list.size());
        // remove phần tử có chỉ số index = 1 khỏi list
        list.remove("PHP");
        System.out.println("\nSố phần tử của list sau khi remove phan tu \"PHP\":             + list);
        System.out.println("Các phần tử của list sau khi remove phan tu \"PHP\": "       + list.size());
```

➢ Tìm kiếm một phần tử ArrayList

```
import java.util.ArrayList;

public class TimKiemArrayList1 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Python");

        // kiểm tra xem PHP có tồn tại trong list hay không?
        System.out.println(list.contains("PHP"));
        // kiểm tra xem ANDROID có tồn tại trong list hay không?
        System.out.println(list.contains("ANDROID"));
    }
}
```

➢ Tìm kiếm vị trí xuất hiện đầu tiên của 1 phần tử trong ArrayList

```
import java.util.ArrayList;

public class TimKiemArrayList2 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Python");

        // kiểm tra xem Java có tồn tại trong list hay không?
        System.out.println(list.indexOf("Java"));
        // kiểm tra xem ANDROID có tồn tại trong list hay không?
        System.out.println(list.indexOf("ANDROID"));
    }
}
```

➢ Tìm kiếm vị trí xuất hiện cuối cùng của 1 phần tử trong List.

```
import java.util.ArrayList;

public class TimKiemArrayList3 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");

        // kiểm tra xem Java có tồn tại trong list hay không?
        System.out.println(list.lastIndexOf("Java"));
        // kiểm tra xem ANDROID có tồn tại trong list hay không?
        System.out.println(list.lastIndexOf("ANDROID"));
    }
}
```

➢ Chuyển ArrayList sang mảng (Array) trong Java

```
import java.util.ArrayList;
public class ConvertToArray {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
        // sử dụng phương thức toArray() chuyển list thành mảng
        // kết quả của phương thức này sẽ trả về mảng arr
        Object[] arr = list.toArray();
        // hiển thị các phần tử có trong mảng arr
        for (int i = 0; i < arr.length; i++) {
            System.out.println("Phần tử tại vị trí " + i + " "
                    + "trong arr là " + arr[i]);
        }
    }
}
```

➢ Tạo ArrayList có kiểu generic là String

```
import java.util.ArrayList;
import java.util.Iterator;
public class ArrayListExample1 {
    public static void main(String args[]) {
        // Creating arraylist
        ArrayList<String> list = new
ArrayList<String>();
        // Add objects to arraylist
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
```

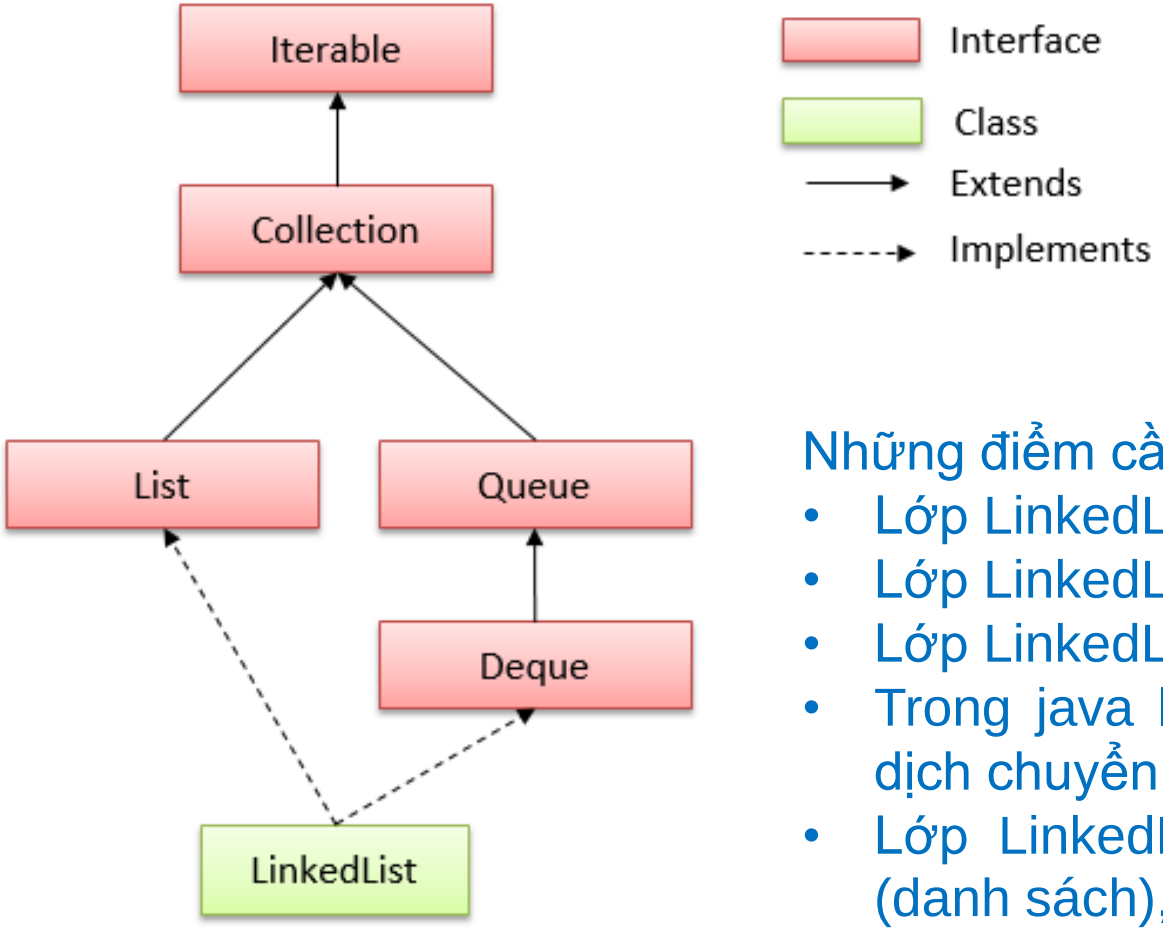
```
// Show list through Iterator
        Iterator<String> itr = list.iterator();
        while (itr.hasNext()) {
            System.out.print(itr.next() + ", ");
        }
        // Show list through for-each
        System.out.println();
        for (String obj : list) {
            System.out.print(obj + ", ");
        }
        // Show list through index
        System.out.println();
        int size = list.size();
        for (int i = 0; i < size; i++) {
            System.out.print(list.get(i) + ", ");
        }
    }
}
```

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

```
import java.util.ArrayList;

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

```
public class ArrayListExample2 {
    public static void main(String[] args) {
        // Create listStudent
        ArrayList<Student> listStudent = new ArrayList<Student>();
        // Create students
        Student student1 = new Student("Bac", 17);
        Student student2 = new Student("Nam", 20);
        Student student3 = new Student("Trung", 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());
        }
    }
}
```

2) Lớp LinkedList trong java



Những điểm cần ghi nhớ về lớp LinkedList:

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

Constructor của lớp LinkedList trong Java

Constructor	Mô tả
LinkedList()	Nó được sử dụng để xay dựng một danh sách trống.
LinkedList(Collection c)	Nó được sử dụng để 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.

Các phương thức của của lớp LinkedList trong Java

Phương thức	Mô tả
boolean add(Object o)	Nó được sử dụng để nối thêm phần tử được chỉ định vào cuối một danh sách.
void add(int index, Object element)	Nó được sử dụng để chèn các phần tử được chỉ định tại các chỉ số vị trí quy định trong một danh sách.
void addFirst(Object o)	Nó được sử dụng để chèn phần tử được chỉ định vào đầu danh sách.
void addLast(Object o)	Nó được sử dụng để chèn phần tử được chỉ định vào cuối danh sách.
int size()	Nó được sử dụng để trả lại số lượng các phần tử trong một danh sách
boolean contains(Object o)	Nó được sử dụng để trở về *true* nếu danh sách có chứa một phần tử được chỉ định.

boolean remove(Object o)	Nó được sử dụng để xóa phần tử được chỉ định đầu tiên trong một danh sách.
Object getFirst()	Nó được sử dụng để trả về phần tử đầu tiên trong một danh sách.
Object getLast()	Nó được sử dụng để trả lại phần tử cuối cùng trong một danh sách.
int indexOf(Object o)	Nó được sử dụng để 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)	Nó được sử dụng để 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.
boolean contains(element)	Kết quả trả về là true nếu tìm thấy element trong danh sách, ngược lại trả về false.

Ví dụ:

```
import java.util.LinkedList;
class Student {
    private String name;
    private int age;
    public Student(String name, int
age) {
        super();
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
```

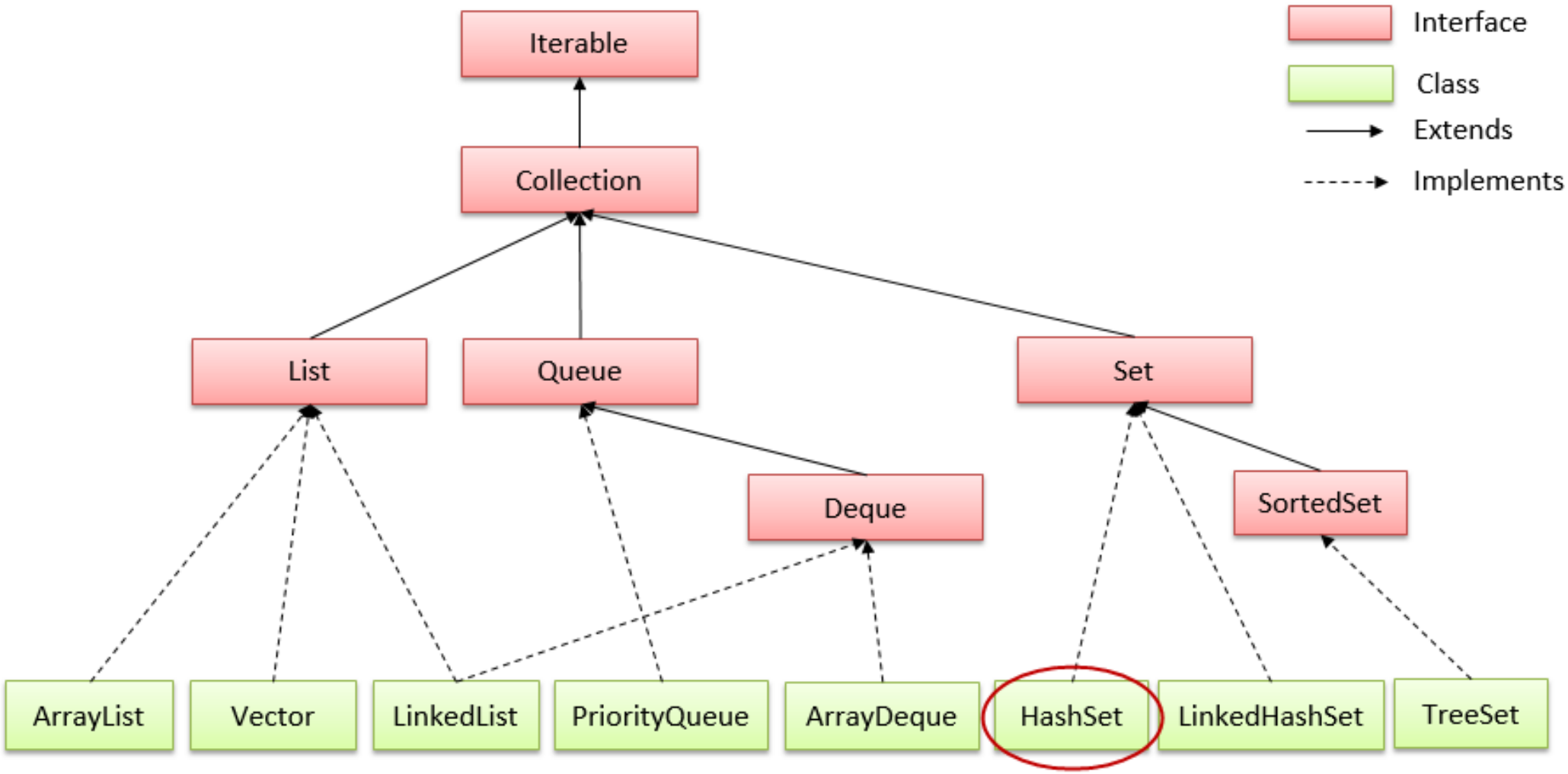
```
public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public String toString() {
        return "Student@[name=" + name + ",
age=" + age + "]";
    }
}
```

```
public class LinkedListExample2 {
    public static void main(String[] args) {
        // Create listStudent
        LinkedList<Student> listStudent = new LinkedList<Student>();
        // Create students
        Student student1 = new Student("Bac", 17);
        Student student2 = new Student("Nam", 20);
        Student student3 = new Student("Trung", 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());
        }
    }
}
```

Sự khác nhau giữa ArrayList và LinkedList

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 ArrayList trong java chỉ có thể **hoạt động như một list** vì nó chỉ implements giao tiếp List.	Lớp LinkedList trong java có thể **hoạt động như một list và queue(hàng đợi)** vì nó implements các giao tiếp 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.

3) Lớp HashSet trong Java



- Các điểm quan trọng về lớp HashSet trong java là:
  - HashSet chỉ chứa các phần tử duy nhất.
  - HashSet lưu trữ các phần tử bằng cách sử dụng một cơ chế được gọi là băm (hash table).
- Một hash table lưu giữ thông tin bởi sử dụng một kỹ thuật được gọi là hashing (băm):nội dung mang tính thông tin của một key được sử dụng để quyết định một value duy nhất, được gọi là hash code của nó.
- Hash code sau đó được sử dụng như là index, tại đó dữ liệu mà liên kết với key được lưu giữ. Phép biến đổi của key vào trong hash code của nó được thực hiện tự động.

```
HashSet<String> set =
        new HashSet<String>();

//Adding elements to HashSet

      set.add("RED");

      set.add("GREEN");

      set.add("BLUE");

      set.add("PINK");

//Removing "RED" from HashSet

      set.remove("RED");
```

```
public HashSet()
{
  map = new HashMap<String, Object>();
}

public boolean add("RED")
{
  return map.put("RED",PRESENT)==null;
}

public boolean add("GREEN")
{
  return map.put("GREEN",PRESENT)==null;
}

public boolean add("BLUE")
{
  return map.put("BLUE",PRESENT)==null;
}

public boolean add("PINK")
{
  return map.put("PINK",PRESENT)==null;
}

public boolean remove("RED")
{
  return map.remove("RED")==PRESENT;
}
```

Key	Value
RED	PRESENT
GREEN	PRESENT
BLUE	PRESENT
PINK	PRESENT

Internal HashMap object

Xóa phần tử có key là "RED" khỏi đối tượng HashMap

Trong đó, PRESENT là một constant được định nghĩa như sau:
private static final Object PRESENT = new OBJECT();

Constructor của lớp HashSet trong Java

Constructor	Mô tả
HashSet()	Nó được sử dụng để khởi tạo một HashSet trống có dung lượng ban đầu mặc định (16) và hệ số tải (0,75).
HashMapHashSet(Collection c)	Nó được sử dụng để xây dựng một HashSet chứa collection c được chỉ định.
HashSet(int initialCapacity)	Nó được sử dụng để xây dựng một HashSet trống với dung lượng ban đầu được chỉ định và hệ số tải mặc định là 0,75.
HashSet(int initialCapacity, float loadFactor)	Nó được sử dụng để xây dựng HashSet trống có dung lượng ban đầu được chỉ định và hệ số tải được chỉ định

Các phương thức của lớp HashSet trong java

Method	Description
boolean add(Object element)	Nó được sử dụng để chèn các phần tử vào HashSet.
boolean addAll(Collection c)	Nó được sử dụng để chèn tất cả các phần tử của c vào HashSet.
void clear()	Xóa tất cả các phần tử khỏi HashSet.
boolean contains(Object element)	Trả về true nếu tập hợp này chứa phần tử đã chỉ định.
boolean containsAll(Collection c)	Trả về true nếu HashSet chứa tất cả các phần tử của collection c đã chỉ định.
boolean equals(Object o)	So sánh các đối tượng được chỉ định với HashSet.
boolean isEmpty()	Trả về true nếu HashSet không chứa phần tử.
int hashCode()	Trả về giá trị mã băm

Iterator iterator()	Trả về một trình vòng lặp iterator để duyệt qua các phần tử của HashSet.
boolean remove(Object o)	Xóa phần tử đã chỉ định khỏi HashSet.
boolean removeAll(Collection c)	Xóa khỏi HashSet tất cả các phần tử của nó được chứa trong collection c đã chỉ định.
boolean retainAll(Collection c)	Chỉ giữ lại các phần tử trong HashSet được chứa trong collection c đã chỉ định.
int size()	Trả về số lượng các phần tử của HashSet.
Object[] toArray()	Trả về một mảng chứa tất cả các phần tử trong HashSet.
T[] toArray(T[] a)	Trả về một mảng chứa tất cả các phần tử trong HashSet, kiểu run-time của mảng trả về là kiểu đã chỉ định.

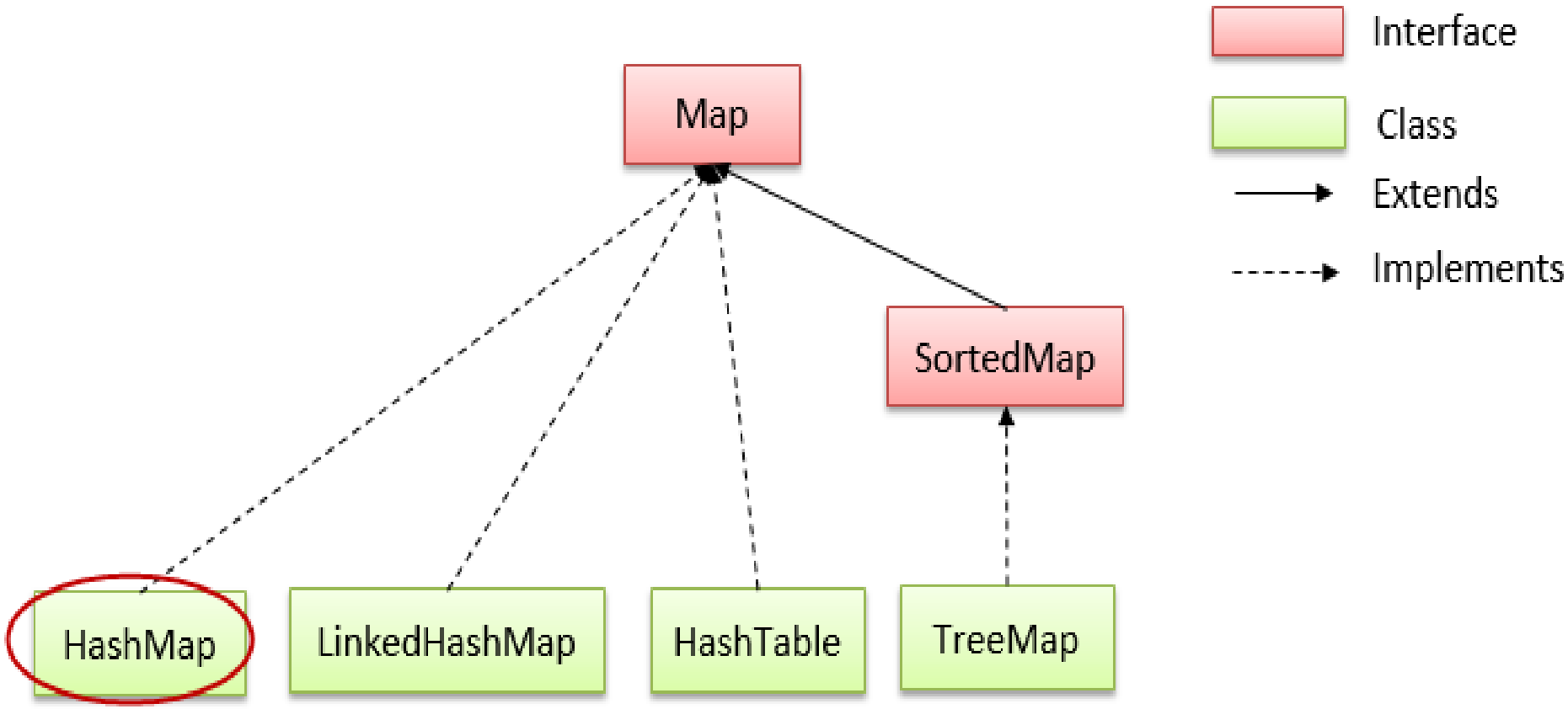
Ví dụ:

```
class Student implements {
    private String name;
    private int age;
    private String address;

    public Student() {
    }
    public Student(String name, int age, String address) {
        super();
        this.name = name;
        this.age = age;
        this.address = address;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
```

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

    public void setAge(int age) {
        this.age = age;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    @Override
    public String toString() {
        return "Student@name=" + name + ",age=" + age
+ ",address=" + address;
    }
}
```

```
import java.util.HashSet;

public class HashSetExam8 {
    public static void main(String[] args) {
        // khoi tao hashSet
        HashSet<Student> hashSet = new HashSet<Student>();
        // Tao cac doi tuong Student
        Student student1 = new Student("Cong", 17, "Hanoi");
        Student student2 = new Student("Dung", 16, "Haiphong");
        Student student3 = new Student("Ngon", 18, "Hanoi");
        Student student4 = new Student("Hanh", 19, "Danang");
        // them cac doi tuong Student vao hashSet
        hashSet.add(student1);
        hashSet.add(student2);
        hashSet.add(student3);
        hashSet.add(student4);
        hashSet.add(student1);
        // Hien thi hashSet
        for (Student student : hashSet) {
            System.out.println(student.toString());
        }
    }
}
```

4) Lớp HashMap trong Java



Những điểm quan trọng về lớp HashMap trong java là:

- HashMap lưu trữ dữ liệu dưới dạng cặp key và value.
- Nó chứa các key duy nhất.
- Nó có thể có 1 key là null và nhiều giá trị null.
- Nó duy trì các phần tử KHÔNG theo thứ tự.

K: kiểu key để lưu trữ.
V: kiểu giá trị được ánh xạ.

Constructor của lớp HashMap trong Java

Constructor	Mô tả
HashMap()	Nó được sử dụng để khởi tạo một HashMap trống.
HashMap(int capacity, float loadFactor)	Nó được sử dụng để xây dựng một HashMap trống với dung lượng (capacity) ban đầu được chỉ định và hệ số tải (loadFactor) được chỉ định.
HashMap(int capacity)	Nó được sử dụng để xây dựng một HashMap trống với dung lượng ban đầu được chỉ định và hệ số tải mặc định là 0,75.
HashMap(Map t)	Nó được sử dụng để xây dựng HashMap mới với một Map đã cho.

Phương thức của lớp HashMap trong Java

Phương thức	Mô tả
void clear()	Xóa tất cả các phần tử của HashMap.
Object clone()	Trả về một bản copy của HashMap.
boolean containsKey(Object key)	Trả về true nếu HashMap chứa một phần tử có key được chỉ định.
boolean containsValue(Object value)	Trả về true nếu HashMap chứa một phần tử có giá trị (value) được chỉ định.
Set entrySet()	Trả về Collection view các ánh xạ có trong HashMap.
Object get(Object key)	Trả về giá trị của key được chỉ định.
boolean isEmpty()	Trả về true nếu HashMap trống.
Set keySet()	Trả về một Set interface chứa tất cả các key của HashMap.
Object put(Object key, Object value)	Thêm một cặp key-value vào HashMap.
void putAll(Map t)	Sao chép các phần tử của Map được chỉ định vào HashMap.
Object remove(Object key)	Xóa một phần tử có key được chỉ định ra khỏi HashMap.
int size()	Trả về số phần tử của HashMap.
Collection values()	Trả về Collection của các giá trị có trong HashMap.

Ví dụ:

```
public class Student{
    private String name;
    private int age;
    private String address;

    public Student() {
    }

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

    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
    @Override
    public String toString() {
        return "Student@name=" + name + ",age=" + age +
",address=" + address;
    }
}
```

```
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
public class HashMapExample6 {
public static void main(String args[]) {
        // init hashMap
        Map<String, Student> hashMap = new HashMap<String, Student>();
        // add elements to hashMap
        hashMap.put("1", new Student("A", 12, "Hanoi"));
        hashMap.put(null, new Student("C", 13, "Hanoi"));
        hashMap.put("2", null);
        hashMap.put("4", new Student("D", 14, "Hanoi"));
        // show hashMap
        show(hashMap);
    }
public static void show(Map<String, Student> map) {
        Set<String> keySet = map.keySet();
        for (String key : keySet) {
            System.out.println(key + " " + map.get(key));
        }
    }
}
```

Bài tập:

Bài 1:

Sử dụng ArrayList kết hợp với các lớp ObjectInputStream/ ObjectOutputStream để xử lý và thao tác với tệp.

Bài 2:

Áp dụng bài 1 cho 3/16 bài thực hành đã cho từ chương trước.

Gợi ý bài 1:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;
public class WriteListEx1 {
    // Windows: C:/Data/test/flowers.data
    private static String file_path = "/Volumes/Data/test/flowers.data";
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        writeFile();
        readFile();
    }
```

```
private static void writeFile() throws IOException {
    ArrayList<String> flowers = new ArrayList<String>();
    flowers.add("Tulip");
    flowers.add("Daffodil");
    flowers.add("Poppy");
    flowers.add("Sunflower");
    flowers.add("Bluebell");
    File file = new File(file_path);
    file.getParentFile().mkdirs();
    OutputStream os = new FileOutputStream(file);
    ObjectOutputStream oos = new ObjectOutputStream(os);
    // Write a String
    oos.writeUTF("A list of flowers");
    // Write an Object
    oos.writeObject(flowers);
    oos.close();
  }
```

```
@SuppressWarnings("unchecked")
   private static void readFile() throws IOException, ClassNotFoundException {
      File file = new File(file_path);
      file.getParentFile().mkdirs();
      InputStream is = new FileInputStream(file);
      ObjectInputStream ois = new ObjectInputStream(is);
      // Read a String
      String info = ois.readUTF();
      // Read an Object
      List<String> flowers = (List<String>) ois.readObject();
      System.out.println(info);
      System.out.println();
      for (String s : flowers) {
         System.out.println(s);
      }
      ois.close();
   }
}
```

Gợi ý bài 2:

Bài tập quản lý sinh viên trong Java

Đề bài: Viết chương trình quản lý sinh viên. Mỗi đối tượng sinh viên có các thuộc tính sau: id, name, age, address và gpa (điểm trung bình). Yêu cầu: tạo ra một menu với các chức năng sau:

/**/

1. Add student.
2. Edit student by id.
3. Delete student by id.
4. Sort student by gpa.
5. Sort student by name.
6. Show student.
0. Exit.

/**/

Yêu cầu thêm: list sinh viên được lưu vào file "student.txt" hoặc cơ sở dữ liệu.

Cấu trúc của project:

- Cấu trúc của project được tạo trên Netbeans/Eclipse:

- Bài tập quản lý sinh viên trong java

Cấu trúc của project: Gồm các tệp

1. Tạo lớp Student.java
2. Tạo lớp StudentDao.java
3. Tạo lớp SortStudentByGPA.java
4. Tạo lớp SortStudentByName.java
5. Tạo lớp StudentManager.java
6. Tạo lớp Main.java

Trong đó:

- Lớp Student để lưu thông tin cho mỗi sinh viên.
- Lớp StudentDao để đọc và ghi sinh viên vào file.
- Lớp SortStudentByGPA được implements Comparator để sắp xếp sinh viên tăng dần theo điểm trung bình.
- Lớp SortStudentByName được implements Comparator để sắp xếp sinh viên tăng dần theo tên.
- Lớp StudentManager cung cấp các phương thức để quản lý sinh viên như thêm, sửa, xóa, sắp xếp và hiển thị sinh viên.
- Lớp Main chứa phương thức public static void main() để chạy ứng dụng và menu như yêu cầu của bài toán.

File: Student.java

```
package com.qlsv;
import java.io.Serializable;
public class Student implements Serializable {
    private int id;
    private String name;
    private byte age;
    private String address;
    /* điểm trung bình của sinh viên */
    private float gpa;
    public Student() {
    }
    public Student(int id, String name, byte age,
            String address, float gpa) {
        super();
        this.id = id;
        this.name = name;
        this.age = age;
        this.address = address;
        this.gpa = gpa;
    }
    public int getId() {
        return id;    }
    public void setId(int id) {
        this.id = id;    }
    public String getName() {
        return name;    }
    public void setName(String name) {
        this.name = name;    }
    public byte getAge() {
        return age;    }
    public void setAge(byte age) {
        this.age = age;    }
    public String getAddress() {
        return address;    }
    public void setAddress(String address) {
        this.address = address;    }
    public float getGpa() {
        return gpa;    }
    public void setGpa(float gpa) {
        this.gpa = gpa;    }
}
```

File: StudentDao.java

Tạo file "student.txt" tại thư mục gốc của dự án để lưu danh sách sinh viên.

1

```
package com.qlsv;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;
public class StudentDao {
    private static final String STUDENT_FILE_NAME =
"student.txt";
```

2

```
public void write(List<Student> studentList) {
        FileOutputStream fos = null;
        ObjectOutputStream oos = null;
        try {
            fos = new FileOutputStream(new
File(STUDENT_FILE_NAME));
            oos = new ObjectOutputStream(fos);
            oos.writeObject(studentList);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            closeStream(fos);
            closeStream(oos);
        }
    }
```

3

```
public List<Student> read() {
        List<Student> studentList = new ArrayList<>();
        FileInputStream fis = null;
        ObjectInputStream ois = null;
        try {
            fis = new FileInputStream(new
File(STUDENT_FILE_NAME));
            ois = new ObjectInputStream(fis);
            studentList = (List<Student>) ois.readObject();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } finally {
            closeStream(fis);
            closeStream(ois);
        }
        return studentList;
    }
```

4

```
private void closeStream(InputStream is) {
        if (is != null) {
            try {
                is.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
private void closeStream(OutputStream os) {
        if (os != null) {
            try {
                os.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

File: SortStudentByGPA.java

```
package com.qlsv;

import java.util.Comparator;

public class SortStudentByGPA implements Comparator<Student> {
    @Override
    public int compare(Student student1, Student student2) {
        if (student1.getGpa() > student2.getGpa()) {
            return 1;
        }
        return -1;
    }
}
```

File: SortStudentByName.java

```
package com.qlsv;

import java.util.Comparator;
public class SortStudentByName implements
Comparator<Student> {
    @Override
    public int compare(Student student1, Student student2) {
        return student1.getName().compareTo(student2.getName());
    }
}
```

File StudentManager.java

1

```
package com.qlsv;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
public class StudentManager {
    public static Scanner scanner = new
Scanner(System.in);
    private List<Student> studentList;
    private StudentDao studentDao;
public StudentManager() {
        studentDao = new StudentDao();
        studentList = studentDao.read();
    }
```

2

```
public void add() {
        int id = (studentList.size() > 0) ?
(studentList.size() + 1) : 1;
        System.out.println("student id = " + id);
        String name = inputName();
        byte age = inputAge();
        String address = inputAddress();
        float gpa = inputGpa();
        Student student = new Student(id, name,
age, address, gpa);
        studentList.add(student);
        studentDao.write(studentList);
    }
```

3

```
public void edit(int id) {
        boolean isExisted = false;
        int size = studentList.size();
        for (int i = 0; i < size; i++) {
            if (studentList.get(i).getId() == id) {
                isExisted = true;
                studentList.get(i).setName(inputName());
                studentList.get(i).setAge(inputAge());
                studentList.get(i).setAddress(inputAddress());
                studentList.get(i).setGpa(inputGpa());
                break;
            }
        }
        if (!isExisted) {
            System.out.printf("id = %d not existed.\n", id);
        } else {
            studentDao.write(studentList);
        }
    }
```

4

```
public void delete(int id) {
        Student student = null;
        int size = studentList.size();
        for (int i = 0; i < size; i++) {
            if (studentList.get(i).getId() == id) {
                student = studentList.get(i);
                break;
            }
        }
        if (student != null) {
            studentList.remove(student);
            studentDao.write(studentList);
        } else {
            System.out.printf("id = %d not
existed.\n", id);
        }
    }
```

5

```
public void sortStudentByName() {
        Collections.sort(studentList, new SortStudentByName());
    }
public void sortStudentByGPA() {
        Collections.sort(studentList, new SortStudentByGPA());
    }
public void show() {
        for (Student student : studentList) {
            System.out.format("%5d | ", student.getId());
            System.out.format("%20s | ", student.getName());
            System.out.format("%5d | ", student.getAge());
            System.out.format("%20s | ", student.getAddress());
            System.out.format("%10.1f%n", student.getGpa());
        }
    }
public int inputId() {
        System.out.print("Input student id: ");
        while (true) {
            try {
                int id = Integer.parseInt((scanner.nextLine()));
                return id;
            } catch (NumberFormatException ex) {
                System.out.print("invalid! Input student id again: ");
            }   }   }
```

6

```
private String inputName() {
        System.out.print("Input student name: ");
        return scanner.nextLine();
    }
private String inputAddress() {
        System.out.print("Input student address: ");
        return scanner.nextLine();
    }
private byte inputAge() {
        System.out.print("Input student age: ");
        while (true) {
            try {
                byte age =
Byte.parseByte((scanner.nextLine()));
                if (age < 0 && age > 100) {
                    throw new NumberFormatException();
                }
                return age;
            } catch (NumberFormatException ex) {
                System.out.print("invalid! Input student id
again: ");
            }
        }
    }
```

7

```
private float inputGpa() {
        System.out.print("Input student gpa: ");
        while (true) {
            try {
                float gpa = Float.parseFloat((scanner.nextLine()));
                if (gpa < 0.0 && gpa > 10.0) {
                    throw new NumberFormatException();
                }
                return gpa;
            } catch (NumberFormatException ex) {
                System.out.print("invalid! Input student age again: ");
            }
        }
    }
    // getter && setter
    public List<Student> getStudentList() {
        return studentList;
    }

    public void setStudentList(List<Student> studentList) {
        this.studentList = studentList;
    }
}
```

Lớp Main.java

```
package com.qlsv;
import java.util.Scanner;
 public class Main {
   public static Scanner scanner = new Scanner(System.in);
   public static void main(String[] args) {
      String choose = null;
      boolean exit = false;
      StudentManager studentManager =
                              new StudentManager();
      int studentId;
      // show menu
      showMenu();
      while (true) {
         choose = scanner.nextLine();
         switch (choose) {
         case "1":
            studentManager.add();
            break;
        case "2":
            studentId = studentManager.inputId();
            studentManager.edit(studentId);
            break;
```

```
case "3":
            studentId = studentManager.inputId();
            studentManager.delete(studentId);
            break;
         case "4":
            studentManager.sortStudentByGPA();
            break;
         case "5":
            studentManager.sortStudentByName();
            break;
         case "6":
            studentManager.show();
            break;
         case "0":
            System.out.println("exited!");
            exit = true;       break;
         default:
            System.out.println("invalid! please choose
action in below menu:");
            break;
         }
         if (exit) {              break;          }
             showMenu();
      }   }
```

```
public static void showMenu() {
        System.out.println("----------menu----------");
        System.out.println("1. Add student.");
        System.out.println("2. Edit student by id.");
        System.out.println("3. Delete student by id.");
        System.out.println("4. Sort student by gpa.");
        System.out.println("5. Sort student by name.");
        System.out.println("6. Show student.");
        System.out.println("0. exit.");
        System.out.println("------------------------");
        System.out.print("Please choose: ");
    }
}
```