# Thực hành 4

## 1. List

## 1.1. In các phần tử trong List

```
public static void main(String args[]) {
    List<String> list = new LinkedList<String>();// new ArrayList<String>();//
    list.add("Ravi");//Adding object in arraylist
    list.add("Vijay");
    list.add("Ravi");
    list.add("Ajay");
    System.out.println("Cach 1:");
    Iterator itr = list.iterator();
    while (itr.hasNext()) {
       System.out.print (itr.next()+"; ");
    }
    System.out.println("\nCach 2:");
    for (String st:list) {
       System.out.print (st+"; ");
    }
  }
```

## 1.2. Tìm kiếm sử dụng contains

### *1.2.1. Tìm kiếm sử dụng contains trên các phần tử có kiểu dữ liệu primitive hoặc là lớp có sẵn trong Java*

```
    public static void main(String args[]) {
        List<String> list1 = new LinkedList<String>();// new ArrayList<String>();//
        list1.add("Ravi");//Adding object in arraylist
        list1.add("Vijay");
        list1.add("Ravi");
```

```
        list1.add("Ajay");
        // Cho biết kết quả và giải thích ?
        boolean kq;
        kq = list1.contains(new String("Vijay"));
        kq = list1.contains("Vijay");
        List list2 = new ArrayList();//
        list2.add(1);
        list2.add(5);
        list2.add(9);
        list2.add(4);
        // Cho biết kết quả và giải thích ?
        kq = list1.contains(5);
    }
```

**Hãy cho biết kết quả và giải thích**

### *1.2.2. Tìm kiếm sử dụng contains trên các phần tử có kiểu dữ liệu là lớp ứng dụng (tự xây dựng)*

```
public class Employee    {
    private String firstName;
    private String lastName;
    private Date joinDate;
    public Employee (String firstName, String lastName, Date joinDate) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.joinDate = joinDate;
    }
```

**// các phương thức set, get, toString,** **equals**

```
public static void main(String[] args) throws ParseException {
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        List<Employee > listEmployees = new ArrayList<>();
```

```
        Employee  employee1 = new Employee ("Tom", "Eagar", dateFormat.parse("2007-12-03"));
        Employee  employee2 = new Employee ("Tom", "Smith", dateFormat.parse("2005-06-20"));
        Employee  employee3 = new Employee ("Bill", "Joy", dateFormat.parse("2009-01-31"));
        Employee  employee4 = new Employee ("Bill", "Gates", dateFormat.parse("2005-05-12"));
        Employee  employee5 = new Employee ("Alice", "Wden", dateFormat.parse("2009-01-22"));
        listEmployees.add(employee1);
        listEmployees.add(employee2);
        listEmployees.add(employee3);
        listEmployees.add(employee4);
        listEmployees.add(employee5);

         Employee  employee = new Employee ("Bill", "Joy"     , dateFormat.parse("2009-01-31"));
            Boolean b= listEmployees.contains(employee);
    }
```

**Hãy bổ sung các phương thức set, get, toString, equals cho lớp Employee để phương thức contains thực hiện tìm kiếm theo : joinDate, firstName và lastName.**

## 1.3. Sắp xếp và tìm kiếm trong List

### *1.3.1. Sắp xếp và tìm kiếm trong List trên các phần tử có kiểu dữ liệu primitive hoặc là lớp có sẵn trong Java*

```
    public static void main(String[] args) {
        List<String> names = Arrays.asList(
            "Tom", "Peter", "Alice", "Bob", "Sam",
            "Mary", "Jane", "Bill", "Tim", "Kevin");
        System.out.println("Before sorting: " + names);
        // Sắp xếp
        Collections.sort(names);
        System.out.println("After sorting: " + names);
        // Tìm kiếm
        int i= Collections.binarySearch(names, "Bob");
```

```
        if (i>=0) {
            System.out.println("Tìm thấy ở vị trí:"+i);
        } else
            System.out.println("Không tìm thấy Bob");

        List<Integer> numbers = Arrays.asList(8, 2, 5, 1, 3, 4, 9, 6, 7, 10);
        System.out.println("Before sorting: " + numbers);
        //Sắp xếp
        Collections.sort(numbers);
        System.out.println("After sorting: " + numbers);
    }
```

**Hãy cho biết kết quả và giải thích**

### *1.3.2. Sắp xếp và tìm kiếm trong List trên các phần tử có kiểu dữ liệu là lớp ứng dụng (tự xây dựng)*

```
public class Person    {
    private String firstName;
    private String lastName;
    private Date joinDate;
    public Person    (String firstName, String lastName, Date joinDate) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.joinDate = joinDate;
    }
  // các phương thức set, get, toString, ….
}
public class App {

    public static void main(String[] args) throws ParseException {
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
```

```
List<Person> list = new ArrayList<>();
Person employee1 = new Person("Tom", "Eagar", dateFormat.parse("2007-12-03"));
Person employee2 = new Person("Tom", "Smith", dateFormat.parse("2005-06-20"));
Person employee3 = new Person("Bill", "Joy", dateFormat.parse("2009-01-31"));
Person employee4 = new Person("Bill", "Gates", dateFormat.parse("2005-05-12"));
Person employee5 = new Person("Alice", "Wden", dateFormat.parse("2009-01-22"));

list.add(employee1);
list.add(employee2);
list.add(employee3);
list.add(employee4);
list.add(employee5);
System.out.println("Before sorting: ");
System.out.println(list);
//Yêu cầu 1:
//1.1: Sắp xếp theo joinDate
Collections.sort(list); //
System.out.println("After sorting1: ");
System.out.println(list);
Person employee = new Person("Bill", "Joy" , dateFormat.parse("2009-01-31"));
```

**//1.2: Thực hiện tìm kiếm**

**// Yêu cầu 2**

**// 2.1: Sắp xếp theo firstName và lastName**

```
Collections.sort(list , ??? );
System.out.println("After sorting2: ");
System.out.println(list);
```

**//2.2: Thực hiện tìm kiếm**

```
    }
  }
```

**Hoàn thiện 2 lớp Person và App để thực hiện được 2 yêu cầu ở trên.**

# 2. Set

## 2.1. In các phần tử trong Set

```
public static void main(String args[]) {
    Set<String> set = new TreeSet<String>();//new HashSet<String>()
    set.add("Ravi");
    set.add("Vijay");
    set.add("Ravi");
    set.add("Ajay");
    //Traversing elements
    System.out.println("Cách 1:");
    Iterator<String> itr = set.iterator();
    while (itr.hasNext()) {
       System.out.print (itr.next()+"; ");
    }
     System.out.println("\nCách 2:");
    for (String st:set) {
       System.out.print (st+": ");
    }
  }
```

## 2.2. Tìm kiếm sử dụng contains trên các phần tử có kiểu dữ liệu là lớp ứng dụng (tự xây dựng)

```
public class Employee   {
private String firstName;
private String lastName;
```

```
private Date joinDate;
public Employee (String firstName, String lastName, Date joinDate) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.joinDate = joinDate;
}
// các phương thức set, get, toString, hashCode,  equals

public static void main(String[] args) throws ParseException {
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        Set<Employee > setEmps = new HashSet<>();
        Employee  employee1 = new Employee ("Tom", "Eagar", dateFormat.parse("2007-12-03"));
        Employee  employee2 = new Employee ("Tom", "Smith", dateFormat.parse("2005-06-20"));
        Employee  employee3 = new Employee ("Bill", "Joy", dateFormat.parse("2009-01-31"));
        Employee  employee4 = new Employee ("Bill", "Gates", dateFormat.parse("2005-05-12"));
        Employee  employee5 = new Employee ("Alice", "Wden", dateFormat.parse("2009-01-22"));
        setEmps.add(employee1);
        setEmps.add(employee2);
        setEmps.add(employee3);
        setEmps.add(employee4);
        setEmps.add(employee5);

         Employee  employee = new Employee ("Bill", "Joy"     , dateFormat.parse("2009-01-31"));
         Boolean b= setEmps.contains(employee);
}
}
```

**Hãy bổ sung các phương thức set, get, toString, hashCode, equals cho lớp Employee để phương thức contains thực hiện tìm kiếm theo : joinDate, firstName và lastName.**

## 2.3. Tìm kiếm sử dụng contains trên các phần tử có kiểu dữ liệu là lớp ứng dụng (tự xây dựng)

```
class Student   {
   public String code;
   private String name;
   public Integer score;
   public Student(String code, String name, Integer score) {
      this.code = code;
      this.score = score;
      this.name = name;
   }
  // các phương thức set, get,. ..
// Comparable
   public static void main(String[] args) {
      Set<Student> ts = new TreeSet<>();
      ts.add(new Student("A06338", "AB", 7));
      ts.add(new Student("A07338", "ABC", 7));
      Student stX = new Student("A06338", "ABA", 9);
// Yêu cầu 1: Tìm kiếm theo code
      System.out.println(ts.contains(stX));
  //Yêu cầu 2: Tìm kiếm theo name
       Set<Student> ts1 = new TreeSet<>( new Comparator<Student>(){
            @Override
            public int compare(Student o1, Student o2) {
                 ???
            }}
```

```
        );
        ts1.addAll(ts);

        Student stY = new Student("A063138", "AB", 9);
        System.out.println(ts1.contains(stY));
    }
}
```

**Hãy hoàn thiện lớp Student để thực hiện 2 yêu cầu trên.**

# 3. Map

## 3.1. In các phần tử trong Map

```
public static void main(String args[]) {
    Map<Integer, String> map = new TreeMap<Integer, String>();;//new HashMap<Integer, String>();
    map.put(100, "Amit");
    map.put(101, "Vijay");
    map.put(102, "Rahul");
    //Elements can traverse in any order
    System.out.println("Cách 1");
    for (Map.Entry m : map.entrySet()) {
       System.out.print (m.getKey() + " " + m.getValue()+"; ");
    }

     System.out.println("\nCách 2");
    for (Integer it : map.keySet()) {
       System.out.print (it + " " + map.get(it)+"; ");
    }
  }
```

**3.2. Xây dựng ví dụ minh họa các phương thức: containsKey, containsValue, get trên HashMap với key, value là các lớp có sẵn trong Java.**

**3.3. Xây dựng ví dụ minh họa các phương thức: containsKey, containsValue, get trên HashMap với key, value là các lớp ứng dụng (tự xây dựng)**

**3.4. Xây dựng ví dụ minh họa các phương thức: containsKey, containsValue, get trên TreeMap với key, value là các lớp có sẵn trong Java.**

**3.5. Xây dựng ví dụ minh họa các phương thức: containsKey, containsValue, get trên TreeMap với key, value là các lớp ứng dụng (tự xây dựng)**