Kiểm tra 15 phút
Yêu cầu không sửa hàm main()

- Tạo lớp Employee: String name, double salary
- Tạo lớp Seller: double revenue; cho Seller kế thừa Employee

Công thức tính: salary= salary+ 5%*revenue (revenue< 4000)

salary= salary+ 10%*revenue (revenue>=4000)

-> Nhiệm vụ:

- Hãy thêm các hàm cần thiết để thực hàm main
- Được phép thêm, sửa code để code hết lỗi và chạy được đúng

Comparator

- Khi so sánh kiểu String -> Sử dụng comparator()
- Khi so sánh kiểu Number -> Sử dụng toán từ “>, <, >=, <=, ==”

*Mẹo: Chỉ nên dùng một loại toán tử để tránh gây nhầm lẫn và nhớ theo ví dụ sau:

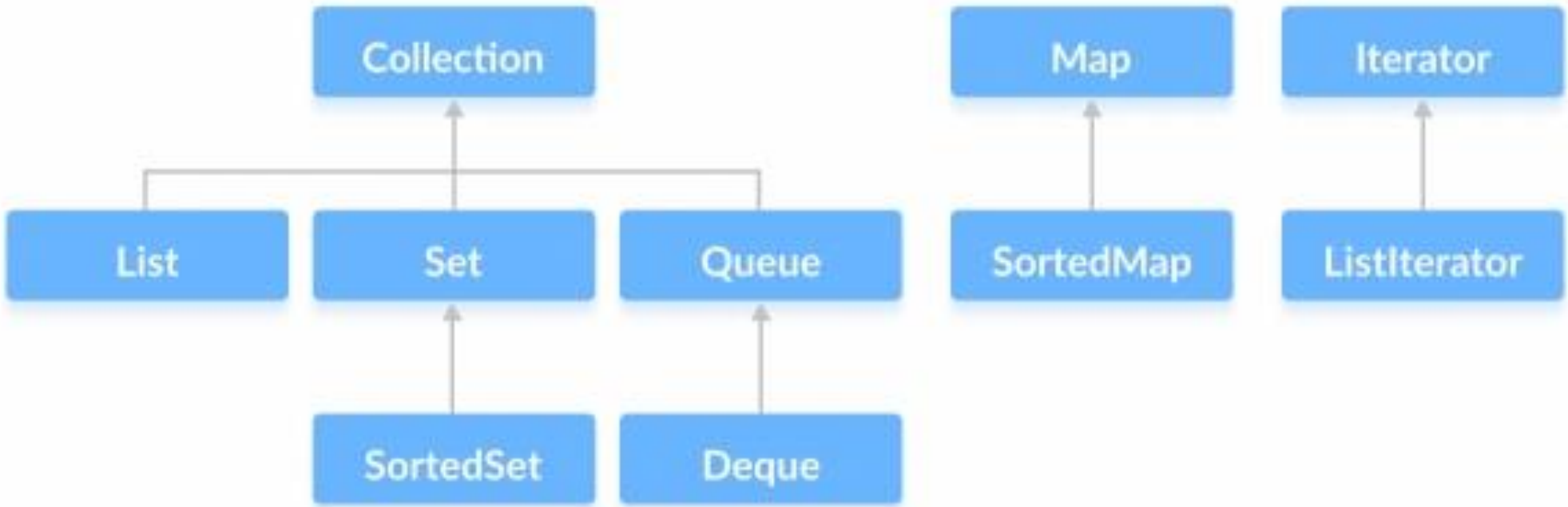
- Return o1.getAge()-o2.getAge() -> tăng dần
- Return o2.getAge()-o1.getAge() -> giảm dần

```
List<Student> listStudents = new ArrayList<Student>();
    // add students to list
    listStudents.add(new Student(1, “Nam", 29, "Hanoi"));
    listStudents.add(new Student(2, “Lan", 24, “HCM"));
    listStudents.add(new Student(3, "Phu", 10, "Hanoi"));

    // sort list student by it's name ASC
    System.out.println("sort list student by it's name ASC: ");
    Collections.sort(listStudents, new Comparator<Student>() {
      @Override
      public int compare(Student o1, Student o2) {
        return o1.getName().compareTo(o2.getName());
      }
    } );
    // show list students
    for (Student student : listStudents) {
      System.out.println(student.toString());
    }

    // sort list student by it's age ASC
    System.out.println("sort list student by it's age ASC: ");
    Collections.sort(listStudents, new Comparator<Student>() {
      @Override
      public int compare(Student o1, Student o2) {
        return o1.getAge() > o2.getAge() ? 1 : -1;
      }
    } );
    // show list students
    for (Student student : listStudents) {
      System.out.println(student.toString());
    }
```

Java Collections Framework



Khái niệm

Collections

- Collections được dùng để lưu trữ, truy xuất, tương tác với dữ liệu (thêm, sửa, xóa)

Framwork

- Framework là một tập hợp các thư viện (Library) đã được đóng gói để hỗ trợ phát triển ứng dụng dựa trên Framework đó. Đồng thời, Framework cũng cung cấp các nguyên tắc, cấu trúc của ứng dụng mà chúng ta phải tuân theo.

Iterator

- Iterator là một Interface cung cấp một số các phương thức để duyệt (lặp) qua các phần tử của bất kỳ tập hợp nào.
- Iterator còn có khả năng xóa phần tử của một tập hợp trong quá trình lặp.

- Các phương thức trong Iterator:

Phương thức	Mô tả
hasNext()	Trả về true nếu iterator còn phần tử kế tiếp.
next()	Trả về phần tử hiện tại và di chuyển con trỏ trỏ tới phần tử tiếp theo.
remove()	Loại bỏ phần tử (hiếm khi dùng).

```java
public static void main(String[] args) {
// List is a sub-interface of Collection.
List<String> classes = new ArrayList<String>();
classes.add("12A");
classes.add("12A1");
classes.add("12B");
classes.add("12B1");
classes.add("12C");

Iterator<String> iterator = classes.iterator();
while (iterator.hasNext()) {
   String class = iterator.next();
   System.out.println(class);
  }
}
```

```
12A
12A1
12B
12B1
12C
```

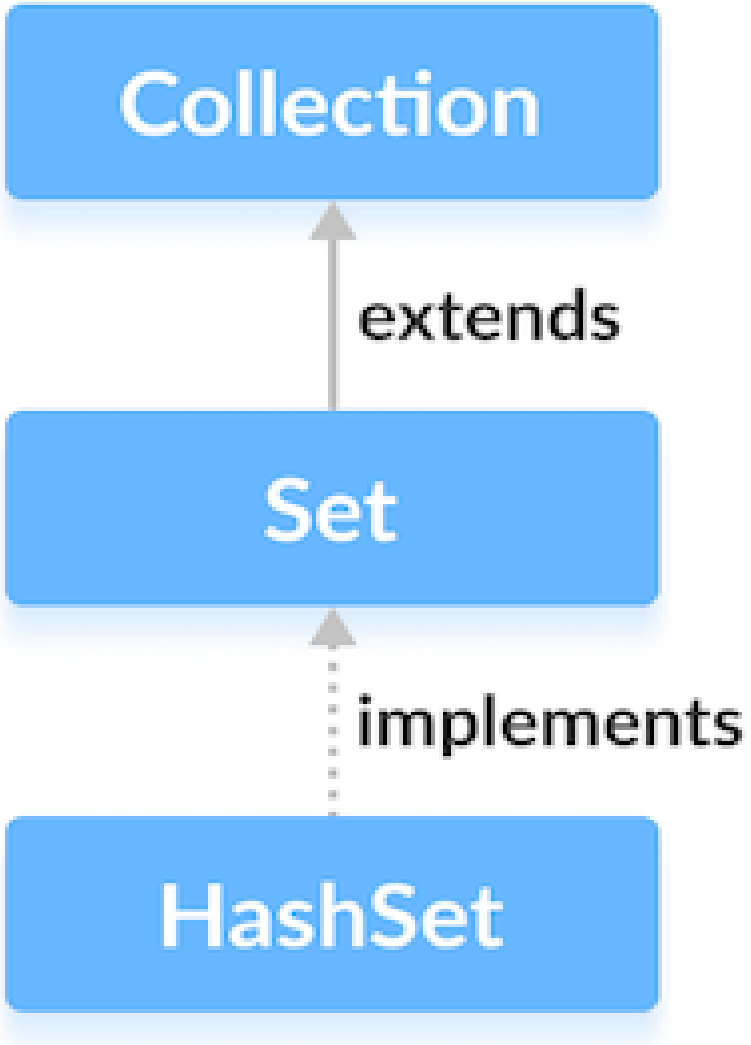
```java
public static void main(String[] args) {
 // List is a sub-interface of Collection.
List<Integer> years = new ArrayList<Integer>();
years.add(1998);
years.add(1995);
years.add(2000);
years.add(2006);
years.add(2021);

Iterator<Integer> iterator = years.iterator();
while (iterator.hasNext()) {
    Integer currentYear = iterator.next();
    if(currentYear % 2 == 0) {
       iterator.remove(); // Remove current element
    }
}
// After remove all even numbers:
for (Integer year : years) {
    System.out.println(year);
}
```

```
1995
2021
```

Set

- **Set** là một interface kế thừa interface **Collection** trong java. Set không thể chứa các phần tử trùng lặp.



HashSet

- **HashSet** là một class được sử dụng rộng rãi trong Set interface. HashSet được dùng để tạo collection sử dụng hash table (bảng băm) để lưu trữ.
- HashSet cho phép lưu phần tử 'NULL'

Khai báo Set:

```
Set<String> setA = new HashSet<String>();
Set<String> setA = new HashSet<>();
```

Các phương thức trong Set

add(Object element)	Thêm 1 phần tử cho Set
add(Object o)	Thêm một phần tử
addAll(Collection c)	Chèn tất cả các phần tử của collection khác vào Set
clear()	Xóa tất cả các phần tử khỏi set.
contains(Object element)	Trả về true nếu tập hợp này chứa phần tử đã chỉ định
isEmpty()	Kiểm tra có phần tử hay k
removeAll(Collection c)	Xóa khỏi set tất cả các phần tử của nó được chứa trong collection c đã chỉ định.

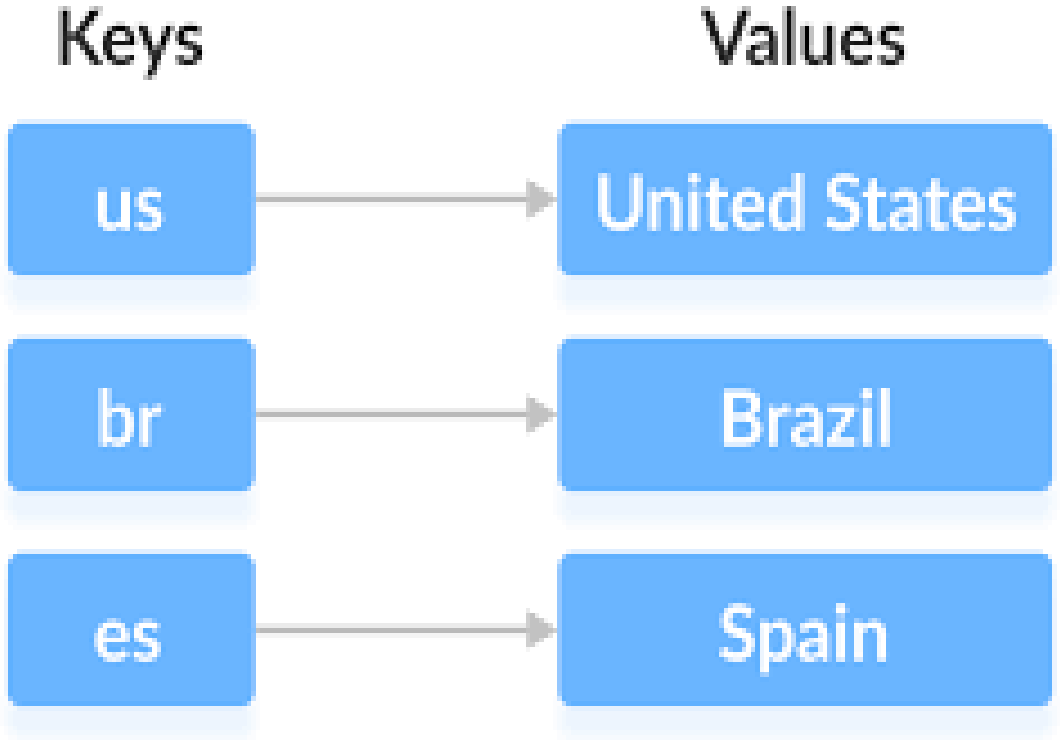
Ví dụ:

```
public static void main(String[] args) {
        Set<String> setA = new HashSet<String>();
        Set<String> setB = new HashSet<String>();
        setB.add("Java");
        setB.add("Python");
        setB.add("C++");
        setA.add("PHP");
        // Thêm các phần tử setB khác vào setA trong Java
        setA.addAll(setB);

        System.out.println("Số phần tử của setA: " + setA.size());
        System.out.println("Các phần tử của setA: " + setA);
        System.out.println("setA có chứa Java không? " + setA.contains("Java"));
        System.out.println("setA có chứa C++ không? " + setA.contains("C++"));
    }
```

Map

- Trong java, Map là một interface được định nghĩa để lưu trữ và truy xuất dữ liệu theo cặp key-value. Mỗi cặp key-value gọi là một entry.
- Map trong java chỉ chứa các giá trị key duy nhất (không trùng lặp).
- Nếu lưu key trùng nhau thì sẽ bị ghi đè giá trị.



Các phương thức trong Map

Method	Description
put(Object key, Object value)	Thêm một cặp Key-Value
remove(Object key)	Sử dụng để xóa một mục nhập của key được chỉ định.
get(Object key)	sử dụng để trả lại giá trị cho khoá được chỉ định.
containsKey(Object key)	Kiểm tra key đó có tồn tại hay k
keySet()	Trả về đối tượng Set có chứa tất cả các keys.

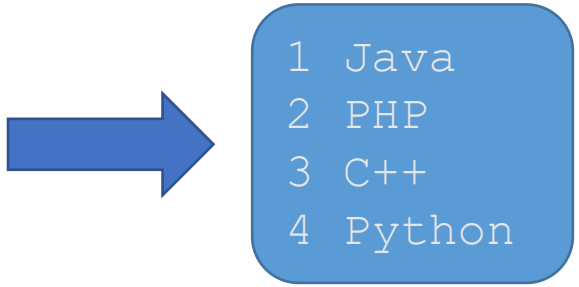
```java
public static void main(String args[]) {
    // init map
    Map<Integer, String> map = new HashMap<Integer, String>();
    map.put(100, "A");
    map.put(101, "B");
    map.put(102, "C");

    // Kiểm tra có key nào = 100 hay không
    if(map.containsKey(100)) {
      System.out.println(map.get(100));
    }

    // show map
    Set<Integer> set = map.keySet();
    for (Integer key : set) {
       System.out.println(key + " " + map.get(key));
    }
  }
```

Map.Entry để truy cập các phần tử của Map

```
public static void main(String args[]) {
    // init map
    Map<Integer, String> map = new HashMap<Integer, String>();
    // add elements to map
    map.put(1, "Java");
    map.put(3, "C++");
    map.put(2, "PHP");
    map.put(4, "Python");
    // show map
    for (Map.Entry<Integer, String> entry : map.entrySet()) {
       System.out.println(entry.getKey() + " " + entry.getValue());
    }
}
```

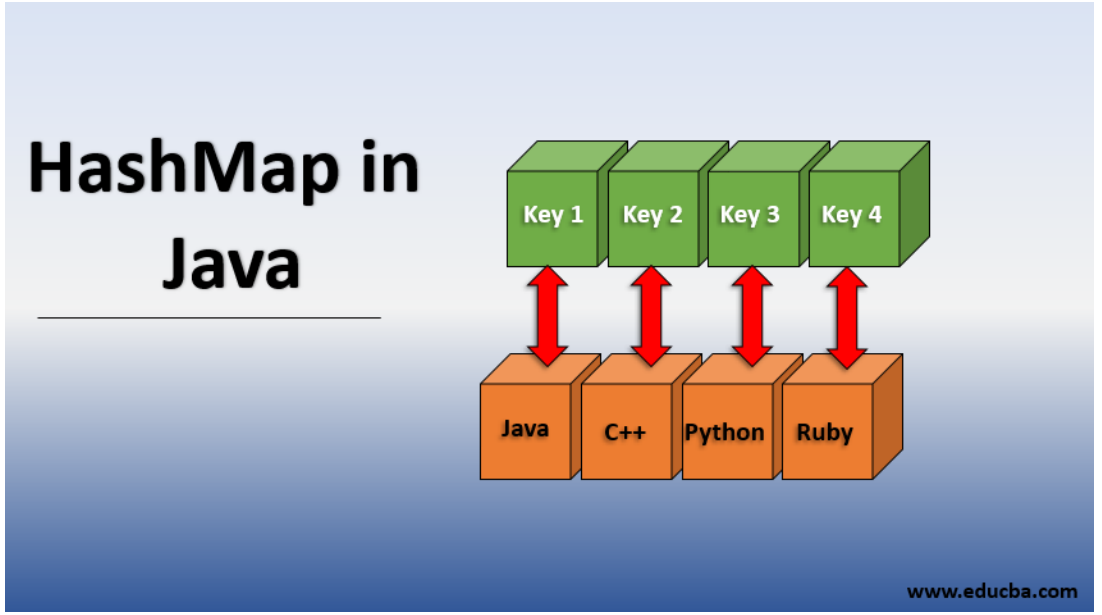


Các phương thức trong Map.Entry

Method	Mô tả
getKey()	Trả về giá trị của key
getValue()	Trả về giá trị của value

HashMap

- HashMap là một phần trong java collections.
- Nó cung cấp các phương thức cơ bản của Map interface.
- Được sử dụng để lưu trữ dữ liệu theo cặp `key-value`.
- Để truy cập một giá trị, ta phải biết `key` của nó.



Đặc điểm của HashMap

- **HashMap** không thể chứa key trùng lặp (mỗi giá trị key là duy nhất)
- Cho phép lưu giá trị ‘null’ (kể cả key hay value)
- Khai báo:

  + HashMap<K, V> hashMap = new HashMap<>();

  + Map<K, V> hashMap = new HashMap<>();

Bài tập 1

DO NOT EDIT ANY CODE in main.
You only need to complete the code in class MyCourse and Course.
We had provided you: Interface – **ICourse** which will declare some operations for a Course – DO NOT EDIT
1. Design and code a class named **Course** that holds information about a Course.
Information of a Course includes:
- A double value holding fee of a Course
- A string describing name of a Course
Include the following functions in your design:
- Constructors to set values for instance variables
- Add needed operations to the class so that the main function can be run and complete the function
2. Design and code a class named **MyCourse** which will implement interface ICourse and complete 2 methods which were declared in ICourse:
- *void f1(List<Course> a, int st)* – Sort the list of courses "a" ascending by course fee if st = 1, otherwise sort the list of courses "a" descending by course name. *The comparison must ignores the case during comparison.*
- *int f2(List<Course> a, double fee)* - count and return numbers of courses in the list “a” which are in the list “a” and has course fee greater than or equals given fee.

Bài tập 2

Phân tích và thiết kế một chương trình quản lý sách trong Techmaster.

Chương trình thực hiện các nhiệm vụ sau:

- Nhiệm vụ 1 - Quản lý sách: Cho phép thêm, xóa, sửa đổi, thông tin về các sách trong Techmaster. Một quyển gồm: Id, name, chủ đề, tác giả, số lượng còn trong thư viện.

- Nhiệm vụ 2 - Quản lý người đọc: Cho phép thêm, xóa, sửa đổi, tìm kiếm thông tin về người đọc trong thư viện. Người đọc gồm: Id, name, số điện thoại, địa chỉ.

- Nhiệm vụ 3 - Quản lý mượn trả sách: Cho phép tạo phiếu mượn sách, và cho phép người đọc trả sách. Thông tin về phiếu mượn sách bao gồm: người mượn, sách được mượn, ngày mượn, ngày bạn phải trả.