Due to the print book page limit, we cannot inlcude all good CheckPoint questions in the physical book. The CheckPoint on this Website may contain extra questions not printed in the book. The questions in some sections may have been reordered as a result. Nevertheless, it is easy to find the CheckPoint questions in the book on this Website. Please send suggestions and errata to Dr. Liang at y.daniel.liang@gmail.com. Indicate the book, edition, and question number in your email. Thanks!

Chapter 21 Check Point Questions

Section 21.2

▼21.2.1
How do you create an instance of Set? How do you insert a new element in a set? How do you remove an element from a set? How do you find the size of a set?

The Set is an interface. To create an instance of Set, you need to use HashSet or TreeSet. To insert an element to a set, use the add method. To remove an element from a set, use the remove method. To find the size of a set, use the size() method.

Hide Answer

▼21.2.2
If two objects o1 and o2 are equal, what is o1.equals(o2) and o1.hashCode() == o2.hashCode()?

If two objects o1 and o2 are equal, o1.equals(o2) is true and o1.hashCode() == o1.hashCode() is true.

Hide Answer

▼21.2.3
What are the differences between HashSet, LinkedHashSet, and TreeSet?

HashSet is unsorted, but TreeSet is sorted. HashSet is more efficient than TreeSet if you don't want the elements in a set to be sorted. If you create a TreeSet using its default constructor, the compareTo method is used to compare the elements in the set, assuming that the class of the elements implements the Comparable interface. To use a comparator, you have to use the constructor TreeSet(Comparator comparator) to create a sorted set that uses the compare method in the comparator to order the elements in the set. A runtime error would occur if you add an element that cannot be compared with the existing elements in the tree set?

Hide Answer

▼21.2.4
How do you traverse the elements in a set?

To traverse a set, use the iterator method to obtain an iterator. You can then traverse the set through the iterator. Using an iterator, you can traverse only sequentially from the beginning to the end.

Hide Answer

▼21.2.5
How do you sort the elements in a set using the compareTo method in the Comparable interface? How do you sort the elements in a set using the Comparator interface? What would happen if you added an element that could not be compared with the existing elements in a tree set? How does the TreeeSet test if two elements are equal?

To sort the elements in a set using the Comparable interface, there are two ways: (1) create a TreeSet

using new TreeSet(), (2) create a TreeSet from a set using new TreeSet(set). To sort the elements in a set using the Comparator interface, create a TreeSet using new TreeSet(Comparator), then add the elements to the tree set.
How does the TreeeSet test if two elements are equal? two elements are considered equal in a TreeSet if e1.compare(e2) is 0 using the comparator. If two elements are equal, only one can be stored in a TreeSet. Note that two elements e1 and e2 in a HashSet and LinkedHashSet are considered equal, if e1.equals(e2) is true and their hashCode() are the same.

Hide Answer

▼21.2.6

Suppose that set1 is a set that contains the strings red, yellow, and green, and that set2 is another set that contains the strings red, yellow, and blue. Answer the following questions:
(a) What are in set1 and set2 after executing set1.addAll(set2)?
(b) What are in set1 and set2 after executing set1.add(set2)?
(c) What are in set1 and set2 after executing set1.removeAll(set2)?
(d) What are in set1 and set2 after executing set1.remove(set2)?
(e) What are in set1 and set2 after executing set1.retainAll(set2)?
(f) What is in set1 after executing set1.clear()?

set2 is not changed by all these methods. (If you use HashSet, the order of the elements in the set is unpredicatable.)
What is set1 and set2 after executing set?;
set1 is [green, red, blue, yellow]
What is set1 and set2 after executing set1.add(set2)?
set1 is [green, red, yellow, [red, blue, yellow]] Note set2 is added as a whole element into set1.
What is set1 and set2 after executing set1.removeAll(set2)?
set1 is [green]
What is set1 and set2 after executing set1.remove(set2)?
set1 is [green, red, yellow]
What is set1 and set2 after executing set1.retainAll(set2)?
set1 is [red, yellow] What is set1 after executing set1.clear();

Hide Answer

▼21.2.7

Show the output of the following code:

```java
import java.util.*;

public class Test {
  public static void main(String[] args) {
    LinkedHashSet<String> set1 = new LinkedHashSet<>();
    set1.add("New York");
    LinkedHashSet<String> set2 = set1;
    LinkedHashSet<String> set3 =
      (LinkedHashSet<String>)(set1.clone());
    set1.add("Atlanta");
    System.out.println("set1 is " + set1);
    System.out.println("set2 is " + set2);
    System.out.println("set3 is " + set3);
    set1.forEach(e -> System.out.print(e + " "));
  }
}
```

set1 is [New York, Atlanta]
set2 is [New York, Atlanta]
set3 is [New York]
New York Atlanta

Hide Answer

▼21.2.8
Show the output of the following code:

```
Set<String> set = new LinkedHashSet<>();
set.add("ABC");
set.add("ABD");
System.out.println(set);
```

[ABC, ABD]

Hide Answer

▼21.2.9
What will the output be if lines 6-7 in Listing 21.5 is replaced by the following code:

```
Set<GeometricObject> set = new HashSet<>();
```

It will display the area of four geometric objects in random order. Note that the eqauls method is not overidden in the Circle class. Therefore, both new Circle(40) in lines 9-10 are stored in the hash set.

Hide Answer

▼21.2.10
Show the output of the following code:

```
Set<String> set = new TreeSet<>(
  Comparator.comparing(String::length));
set.add("ABC");
set.add("ABD");
System.out.println(set);
```

[ABC]

Hide Answer

▼21.2.11
(This is not in the printed book) Show the output of the following code:

```
import java.util.*;
import java.io.*;

public class Test {
  public static void main(String[] args) throws Exception {
    ObjectOutputStream output = new ObjectOutputStream(
      new FileOutputStream("c:\\test.dat"));
    LinkedHashSet<String> set1 = new LinkedHashSet<>();
    set1.add("New York");
    LinkedHashSet<String> set2 =
      (LinkedHashSet<String>)set1.clone();
    set1.add("Atlanta");
```

```
      output.writeObject(set1);
      output.writeObject(set2);
      output.close();

      ObjectInputStream input = new ObjectInputStream(
        new FileInputStream("c:\\test.dat"));
      set1 = (LinkedHashSet<String>)input.readObject();
      set2 = (LinkedHashSet<String>)input.readObject();
      System.out.println(set1);
      System.out.println(set2);
      input.close();
    }
  }
```

[New York, Atlanta]
[New York]

Hide Answer

Section 21.3

▼21.3.1
Suppose you need to write a program that stores unordered, non-duplicate elements, what data structure should you use?

Suppose you need to write a program that stores non-duplicate elements, you should use a HashSet.

Hide Answer

▼21.3.2
Suppose you need to write a program that stores non-duplicate elements in the order of insertion, what data structure should you use?

Use a LinkedHashSet.

Hide Answer

▼21.3.3
Suppose you need to write a program that stores non-duplicate elements in increasing order of the element values, what data structure should you use?

Use a TreeSet.

Hide Answer

▼21.3.4
Suppose you need to write a program that stores a fixed number of the elements (possibly duplicates), what data structure should you use?

Use an array.

Hide Answer

▼21.3.5
Suppose you need to write a program that stores the elements in a list with frequent operations to append and delete elements at the end of the list, what data structure should you use?

Use an ArrayList.

Hide Answer

▼21.3.6
Suppose you need to write a program that stores the elements in a list with frequent operations to insert and delete elements at the beginning of the list, what data structure should you use?

Use a LinkedList.

Hide Answer

Section 21.4

▼21.4.1
Will the CountKeywords program work if lines 33-34 are changed to

```
Set<String> keywordSet =
  new LinkedHashSet<>(Arrays.asList(keywordString));
```

Yes.

Hide Answer

▼21.4.2
Will the CountKeywords program work if lines 33-34 are changed to

```
List<String> keywordSet =
  new ArrayList<String>(Arrays.asList(keywordString));
```

Yes.

Hide Answer

Section 21.5

▼21.5.1
How do you create an instance of Map? How do you add an entry to a map consisting of a key and a value? How do you remove an entry from a map? How do you find the size of a map? How do you traverse entries in a map?

Map is an interface. To create an instance of Map, you need to use the HashMap class or the TreeMap class. The HashMap and TreeMap have various constructors that you can use to create a HashMap or a TreeMap. You can use the put method to add an entry to a map, and the remove method to remove an entry with the specified key from the map. Use the size method to find the size of a map.

Hide Answer

▼21.5.2
Describe and compare HashMap, LinkedHashMap, and TreeMap.

The HashMap, LinkedHashMap, and TreeMap classes are three concrete implementations of the Map interface. The HashMap class is efficient for locating a value, inserting a mapping, and deleting a mapping. The entries in a HashMap are not ordered, but the entries in a LinkedHashMap can be retrieved in the order in which they were inserted into the map (known as the insertion order), or the order in which they were last accessed, from least recently accessed to most recently (access order). The TreeMap class, implementing SortedMap, is efficient for traversing the keys in a sorted order.

Hide Answer

▼21.5.3

Show the printout of the following code:

```
import java.util.*;
public class Test {
  public static void main(String[] args) {
    Map<Integer, String> map = new LinkedHashMap<>();
    map.put("123", "John Smith");
    map.put("111", "George Smith");
    map.put("123", "Steve Yao");
    map.put("222", "Steve Yao");
    System.out.println("(1)  " + map);
    System.out.println("(2)  " + new TreeMap<, String>(map));

    map.forEach((k, v) -> {
      if (k.equals("123")) System.out.println(v);});
  }
}
```

(1) {123=Steve Yao, 111=George Smith, 222=Steve Yao}
(2) {111=George Smith, 123=Steve Yao, 222=Steve Yao}

Hide Answer

Section 21.6

▼21.6.1

Will the CountOccurrenceOfWords program work if line 10 is changed to

```
Map<String, int> map = new TreeMap<>();
```

No. A concrete type must of an object type.

Hide Answer

▼21.6.2

Will the CountOccurrenceOfWords program work if line 17 is changed to

```
if (map.get(key) == null) {
```

Yes. It will work.

Hide Answer

▼21.6.3

Will the CountOccurrenceOfWords program work if lines 32-33 are changed to

```
for (String key: map)
  System.out.println(key + "\t" + map.getValue(key));
```

No. You cannot iterate on keys in a map in Java.

Hide Answer

▼21.6.4

Replace the code in lines 17-24 in one line of code using a conditional expression.

map.put(key, map.containsKey(key) ? map.get(key) + 1 : 1);

Hide Answer

Section 21.7

▼21.7.1

What is wrong in the following code?

```
Set<String> set = Collections.singleton("Chicago");
set.add("Dallas");
```

The singleton set is immutable. You cannot add an element into the singleton set.

Hide Answer

▼21.7.2

What happens when you run the following code?

```
List<String> list = Collections.unmodifiableList(
  Arrays.asList("Chicago", "Boston"));
list.remove("Dallas");
```

The list created from the unmodifiableList method is immutable is an unmodifiable view of the list. You cannot modify the list through this view.

Hide Answer