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!

Section 20.2

▼20.2.1

What is a data structure?

A data structure is a collection of data organized in some fashion.In object-oriented thinking, a data structure is an object that stores other objects, referred to as data or elements. So some people refer a data structure as a container object or a collection object. To define a data structure is essentially to declare a class.

Hide Answer

▼20.2.2

Describe the Java Collections Framework. List the interfaces, convenience abstract classes, and concrete classes under the Collection interface.

The Java Collections Framework defines the Java API for handling common data structures tasks in Java. It defines classes and interfaces for storing and manipulating data in sets, lists, and maps.

A convenience class is an abstract class that partially implements an interface. The Java Collections Framework defines interfaces, convenience abstract classes, and concrete classes.

Hide Answer

▼20.2.3

Can a collection object be cloned and serialized?

Yes. The concrete classes of Set, List, and Map implements the clone() method in the Cloneable interface.

Hide Answer

▼20.2.4

What method do you use to add all the elements from one collection to another collection?

addAll(Collection c).

Hide Answer

▼20.2.5

When should a method throw an UnsupportedOperationException?

If a method has no meaning in the subclass, you can implement it in the subclass to throw java.lang.UnsupportedOperationException, a subclass of RuntimeException. This is a good design that you can use in your project. If a method has no meaning in the subclass, you can implement it as follows:

```
public void someMethod() {
  throw new UnsupportedOperationException
    ("Method not supported");
}
```

Hide Answer

Section 20.3

▼20.3.1

How do you obtain an iterator from a collection object?

The Collection interface extends the Iterable interface. You can obtain an iterator from a collection using the iterator() method.

Hide Answer

▼20.3.2

What method do you use to obtain an element in the collection from an iterator?

Use the next() method.

Hide Answer

▼20.3.3

Can you use a foreach loop to traverse the elements in any instance of Collection?

Yes.

Hide Answer

▼20.3.4

When using a foreach loop to traverse all elements in a collection, do you need to use the next() or hasNext() methods in an iterator?

No. They are implicitly used in a foreach loop.

Hide Answer

Section 20.4

▼20.4.1

Can you use the forEach method on any instance of Collection? Where is the forEach method defined?

Yes. It is defined in the Iterable interface which is a super interface for Collection.

Hide Answer

▼20.4.2

Suppose each element in list is a StringBuilder, write a statement using a forEach method to change the first character to uppercase for each element in list.

```
list.forEach(e -> {
  if (((StringBuilder)e).length() > 0) {
    char ch = ((StringBuilder)e).charAt(0);
```

```
        if (Character.isLowerCase(ch)) {
          ((StringBuilder)e).setCharAt(0, Character.toUpperCase(ch));
        };
      };
    });
```

Hide Answer

Section 20.5

▼20.5.1

How do you add and remove elements from a list? How do you traverse a list in both directions?

Use the add or remove method to add or remove elements from a list. Use the listIterator() to obtain an iterator. This iterator allows you to traverse the list bi-directional.

Hide Answer

▼20.5.2

Suppose that list1 is a list that contains the strings red, yellow, and green, and that list2 is another list that contains the strings red, yellow, and blue. Answer the following questions:

a. What are list1 and list2 after executing list1.addAll(list2)?
b. What are list1 and list2 after executing list1.add(list2)?
c. What are list1 and list2 after executing list1.removeAll(list2)?
d. What are list1 and list2 after executing list1.remove(list2)?
e. What are list1 and list2 after executing list1.retainAll(list2)?
f. What is list1 after executing list1.clear()?

list2 is not changed by all these methods.
a. list1 is [red, yellow, green, red, yellow, blue]
b. list1 is [red, yellow, green, [red, yellow, blue]]
c. list1 is [green] What is list1 and list2 after executing list1.remove(list2);
d. list1 is [red, yellow, green]
e. list1 is [red, yellow] What is list1 after executing list1.clear();
f. list1 is empty

Hide Answer

▼20.5.3

What are the differences between ArrayList and LinkedList? Which list should you use to insert and delete elements at the beginning of a list?

ArrayList and LinkedList can be operated similarly. The critical differences between them are their internal implementation, which impacts the performance. ArrayList is efficient for retrieving elements, and for adding and removing elements from the end of the list. LinkedList is efficient for adding and removing elements anywhere in the list.

Hide Answer

▼20.5.4

Are all the methods in ArrayList also in LinkedList? What methods are in LinkedList but not in ArrayList?

All the methods in ArrayList are also in LinkedList except the trimToSize() method. The methods getFirst, getLast, addFirst, addLast are in LinkedList, but not in ArrayList.

Hide Answer

▼20.5.5

How do you create a list from an array of objects?

A simple way to create a list from an array of objects is to use

```
new ArrayList(Arrays.asList(arrayObject))
```

or

```
new LinkedList(Arrays.asList(arrayObject)).
```

Hide Answer

Section 20.6

▼20.6.1

What are the differences between the Comparable interface and the Comparator interface? In which package is Comparable, and in which package is Comparator?

The Comparable interface contains the compareTo method and Comparator interface contains the compare method and equals method. Normally, if the objects of a class have natural order (e.g., String, Date), let the class implement the Comparable interface. The Comparator interface is more flexible in the sense that it enables you to define a new class that contains the compare(Object, Object) method to compare two objects of other classes.

The Comparable interface is in the java.lang package, and the Comparator interface is in the java.util package.

Hide Answer

▼20.6.2

How do you define a class A that implements the Comparable interface? Are two instances of class A comparable? How do you define a class B that implements the Comparator interface and override the compare method to compare two objects of type B1? How do you invoke the sort method to sort a list of objects of the type B1 using a comparator?

How do you define a class A that implements the Comparable interface?

```
public class A implements Comparable<A> {
  public int compareTo(A o) {
    return an integer;
  }
}
```

Are two instances of class A comparable? Yes.How do you define a class B that implements the Comparator interface and override the compare method to compare two objects of type B1?

```
public class B implements Comparator<B1> {
  public int compare(B1 o1, B1 o2) {
    return an integer;
  }
}
```

How do you invoke the sort method to sort a list of objects of the type B1?

```
list.sort(new B());
```

To sort an array x of objects of the type B1, use

```
java.util.Arrays.sort(x, new B());
```

Hide Answer

▼20.6.3

Write a lambda expression to create a comparator that compares two Loan objects by their annualInterestRate. Create a comparator using the Comparator.comparing method to compare Loan objects on annualInterestRate. Create a comparator to compare Loan objects first on annualInterestRate then on loanAmount.

```
(e1, e2) -> e1.getAnnualInterestRate() < e2.getAnnualInterestRate() ?
-1 :
   e1.getAnnualInterestRate() == e2.getAnnualInterestRate() ? 0 : 1
Comparator.comparing(Loan::getAnnualInterestRate);
Comparator.comparing(Loan::getAnnualInterestRate)
  .thenComparing(Loan:getLoanAmount);
```

Hide Answer

▼20.6.4

Create a comparator using a lambda expression and using the Comparator.comparing method, respectively, to compare Collection objects on their size.

```
(e1, e2) -> e1.size() - e2.size()
Comparator.comparing(Collection::size)
```

Hide Answer

▼20.6.5

Write a statement that sorts an array named points of Point2D objects on their y values and then on their x values.

```
java.util.sort(points,
  Comparator.comparing(Point2D::x).thenComparing(Point2D::y));
```

Hide Answer

▼20.6.6

Write a statement that sorts an ArrayList of strings named list in increasing order of their last character.

```
list.sort((e1, e2) -> {
  if (e1.length() == 0)
    return -1;
```

```
  else (e2.length() == 0)
    return 1;
  else
    return charAt(e1.size() - 1) - charAt(e2.size() - 1);
}
```

Hide Answer

▼20.6.7

Write a statement that sorts a two-dimensional array of double[][] in increasing order of their second column. For example, if the array is double[][] x = {{3, 1}, {2, -1}, {2, 0}}, the sorted array will be {{2, -1}, {2, 0}, {3, 1}}.

```
java.util.Arrays.sort(x, (e1, e2) -> (int)(e1[1] - e2[1]));
```

Hide Answer

▼20.6.8

Write a statement that sorts a two-dimensional array of double[][] in increasing order of their second column as the primary order and the first column as the secondary order. For example, if the array is double[][] x = {{3, 1}, {2, -1}, {2, 0}, {1, -1}}, the sorted array will be {{1, -1}, {2, -1}, {2, 0}, {3, 1}}.

```
java.util.Arrays.sort(x, (e1, e2) -> {
  if (e1[1] - e2[1] != 0)
    return (int)(e1[1] - e2[1]);
  else
    return (int)(e1[0] - e2[0]);
  });
```

Hide Answer

Section 20.7

▼20.7.1

Are all the methods in the Collections class static?

Yes.

Hide Answer

▼20.7.2

Which of the following static methods in the Collections class are for lists, and which are for collections?

```
sort, binarySearch, reverse, shuffle, max, min, disjoint, frequency
```

The methods for lists are: sort, binarySearch, reverse, shuffle

The methods for collections are: max, min, disjoint, frequency

Note that all the methods for collections are also for lists, because lists are collections.

Hide Answer

▼20.7.3

Show the output of the following code:

```
import java.util.*;

public class Test {
```

```
  public static void main(String[] args) {
    List<String> list =
      Arrays.asList("yellow", "red", "green", "blue");
    Collections.reverse(list);
    System.out.println(list);

    List<String> list1 =
      Arrays.asList("yellow", "red", "green", "blue");
    List<String> list2 = Arrays.asList("white", "black");
    Collections.copy(list1, list2);
    System.out.println(list1);

    Collection<String> c1 = Arrays.asList("red", "cyan");
    Collection<String> c2 = Arrays.asList("red", "blue");
    Collection<String> c3 = Arrays.asList("pink", "tan");
    System.out.println(Collections.disjoint(c1, c2));
    System.out.println(Collections.disjoint(c1, c3));

    Collection<String> collection =
      Arrays.asList("red", "cyan", "red");
    System.out.println(Collections.frequency(collection, "red"));
  }
}
[blue, green, red, yellow]
[white, black, green, blue]
false
true
2
```

Hide Answer

▼20.7.4

Which method can you use to sort the elements in an ArrayList or a LinkedList? Which method can you use to sort an array of strings?

You can use Collections.sort(list) to sort an ArrayList or a LinkedList and use Arrays.sort(Object[]) to sort an array of strings. For example,

```
LinkedList<String> list = new LinkedList<>();
list.add("Java"); list.add("Python"); list.add("C++");
java.util.Collections.sort(list); // Sort the list

String[] languages = {"Java", "Python", "C++"};
java.util.Arrays.sort(languages); // Sort the array
```

Hide Answer

▼20.7.5

Which method can you use to perform binary search for elements in an ArrayList or a LinkedList? Which method can you use to perform binary search for an array of strings?

You can use Collections.binary(list, key) to perform binary search for an ArrayList or a LinkedList and use Arrays.binary(Object[], key) to sort an array of strings.

Hide Answer

▼20.7.6

Write a statement to find the largest element in an array of comparable objects.

```
Collections.max(Arrays.asList(arrayObject))
```

Hide Answer

Section 20.8

▼20.8.1

What is the return value from invoking pane.getChildren() for a pane?

The return value is an ObservableList<Node>, which is a subtype of List<Node>.

Hide Answer

▼20.8.2

How do you modify the code in the MutilpleBallApp program to remove the first ball in the list when the button is clicked?

Replace line 75 with the following code:

```
getChildren().remove(getChildren().size() - 1);
```

Hide Answer

▼20.8.3

How do you modify the code in the MutilpleBallApp program so that each ball will get a random radius between 10 and 20?

Change line 133 to

```
radius = Math.random*11 + 10;
```

Hide Answer

Section 20.9

▼20.9.1

How do you create an instance of Vector? How do you add or insert a new element into a vector? How do you remove an element from a vector? How do you find the size of a vector?

Vector is the same as ArrayList except that, except that Vector contains the synchronized methods for accessing and modifying the vector. Since Vector implements List, you can use the methods in List to add, remove elements from a vector, and use the size() method to find the size of a vector. To create a vector, use either its constructors.

Hide Answer

▼20.9.2

How do you create an instance of Stack? How do you add a new element to a stack? How do you remove an element from a stack? How do you find the size of a stack?

Stack is a subclass of Vector. The Stack class represents a last-in-first-out stack of objects. The elements are accessed only from the top of the stack. You can retrieve, insert, or remove an element from the top of the stack. To add a new element to a

stack, use the push method. To remove an element from the top of the stack, use the method pop. To find a stack size, use the size() method.

Hide Answer

▼20.9.3

Does Listing 20.1, TestCollection.java, compile and run if all the occurrences of ArrayList are replaced by LinkedList, Vector, or Stack?

Yes, because these classes are subtypes of the Collection interface.

Hide Answer

Section 20.10

▼20.10.1

Is java.util.Queue a subinterface of java.util.Collection, java.util.Set, or java.util.List? Does LinkedList implement Queue?

java.util.Queue is a subinterface of java.util.Collection, and LinkedList implements Queue.

Hide Answer

▼20.10.2

How do you create a priority queue for integers? By default, how are elements ordered in a priority queue? Is the element with the least value assigned the highest priority in a priority queue?

Use the constructors of PriorityQueue to create priority queues. By default, the elements in a priority queue are ordered in their natural order using the compareTo method in the Comparable interface. The element with the least value is assigned the highest priority in PriorityQueue.

Hide Answer

▼20.10.3

How do you create a priority queue that reverses the natural order of the elements?

```
new PriorityQueue(initialCapacity, Collections.reverseOrder()).
```

Hide Answer

Section 20.11

▼20.11.1

Can the EvaluateExpression program evaluate the following expressions "1+2", "1 + 2", "(1) + 2", "((1)) + 2", and "(1 + 2)"?

Yes.

Hide Answer

▼20.11.2

Show the change of the contents in the stacks when evaluating "3 + (4 + 5) * (3 + 5) + 4 * 5" using the EvaluateExpression program.

Omitted.

Hide Answer

▼20.11.3

If you enter an expression "4 + 5 5 5", the program will display 10. How do you fix this problem?

You can fix this problem by throwing an exception if operandStack is not empty after poping the result out of the operandStack stack.

Hide Answer