# Homework M3: Interfaces

**Due** Feb 21 at 11:59pm **Points** 100 **Questions** 45
**Available** until May 30 at 11:59pm **Time Limit** None

# Instructions

Review the **Homework FAQ** page for details about submitting homework. In this homework, you will:

- trace the use of bags (BagInterface) and lists (ListInterface and List)
- write code to use bags and lists from the client perspective
- write code to implement the Comparable interface

## Homework Files

Below is the driver/tester program. I strongly recommend using this to test your code before submitting. You can ignore the *test methods* at the end of the file.

**Important Note:** Be sure to use the correct files from the 5th edition (included in the zip file below). Do **not** use the interfaces from older editions- there are differences.

**HomeworkM3Files.zip**

## Notes

### Use the correct object type.

- The questions refer to BagInterface, ListInterface, and List.
- Read each question carefully to make sure you know which type of object is being used!
- Follow all specifications defined for those classes, including method names, return values, and indexing rules.
- You will lose points for syntax errors (including incorrect method names).

### Remember what it means to write at the client level.

- Writing code at the client level means you should write Java statements that only have access to the methods defined in the BagInterface, ListInterface, or List interfaces.
- You don't know anything about how these methods are implemented- you only know what is described and specified in the documentation of the class.
- In the driver program, I create objects using ArrayBag, LinkedBag, AList, LinkedList, and ArrayList.

This is only for testing purposes- you should **not** be referencing these classes for this homework. (We'll be learning about them soon!)

## Do not use other kinds of data structures.

- For full credit, do **not** use the toArray() method or additional data structures such as arrays, ArrayLists, etc.
- The purpose of this homework is to practice with BagInterface, ListInterface, and List, not to practice manipulating arrays or other data structures.
- When listed on individual questions, you can create temporary data structures of a specific type. You will then need to initialize as one of the concrete classes (e.g., BagInterface<Integer> temp = new ArrayBag<Integer>()). This **is** allowed.
- If not sure what you are allowed to use in an answer, post to discussion board.

## Account for special cases.

- Be sure to account for all situations so that your code will not crash.
- Unless explicitly stated otherwise, your code should run and produce an expected result for an empty and singleton bag/list.

This quiz was locked May 30 at 11:59pm.

## Attempt History

| | Attempt | Time | Score |
|---|---|---|---|
| **LATEST** | Attempt 1 | 5,304 minutes | 108 out of 100 |

Score for this quiz: **108** out of 100
Submitted Feb 18 at 1:22pm
This attempt took 5,304 minutes.

## Coding Questions

Use the Person class provided in **HomeworkM3Files**

(https://ccsf.instructure.com/courses/47904/files/7254226/download?download_frd=1) . Update the class so that it implements the Comparable interface.

1. Write the new class header for this class.
2. Write the complete compareTo method using the following logic:
   - Person objects are ordered by last name (ignoring case/capitalization), then by first name (ignoring case/capitalization), then by id.
   - In other words, if two people have the same last name, they should be ordered based on first name. If two people have the same last and first name, they should be ordered based on id.
   - Use alphabetic ordering and ignore capitalization (meaning that "cat" is treated the same as "CAT").

Note: If you run the driver for this question before you update the Person class header, the code will crash.

## Question 1

**2 / 2 pts**

Write the **complete** new class header for the Person class that implements the Comparable interface.

(Reminder: after the due date, I review all auto-graded answers to give points to correct answers that Canvas missed.)

**Correct!**

```
public class Person implements Comparable<Person>
```

**orrect Answers**

```
public class Person implements Comparable <Person>{
public class Person implements Comparable<Person>{
public class Person implements Comparable<Person> {
public class Person implements Comparable<Person>
public class Person implements Comparable <Person> {
public class Person implements Comparable <Person>
```

## Question 2

**10 / 10 pts**

Paste the complete compareTo method (including the method header) here.

Your Answer:

```
@Override
public int compareTo(Person otherPerson) {
    // comparing person's last name
    if (!this.lastName.toLowerCase().equals(otherPerson.getLastName().toLow
erCase())) {
        return this.lastName.toLowerCase().compareTo(otherPerson.getLastNam
e().toLowerCase());
    } else { // comparing person's first name
    if(!this.firstName.toLowerCase().equals(otherPerson.getFirstName().toLo
werCase())){
        return this.firstName.toLowerCase().compareTo(otherPerson.getFirstN
ame().toLowerCase());    // equals
    } else // comparing person id's
    if(this.id < otherPerson.getId()) {
        return -1;    // less than
    } else if (this.id > otherPerson.getId()) {
        return 1;    // greater than
    } else {
        return 0;    // equals
        }
    }
```

```
}
```

note that you can simplify the else code with return Integer.compare(this.id, otherPerson.id)

These questions ask you to write code at the client level. Read carefully to determine which ADT is required.

Also remember that when working from the client interface, you do not know what the implementing class is! You only know that an object is of type BagInterface or ListInterface. You don't know what the actual type is (e.g., it could be ArrayBag, LinkedBag, or some other implementing class).

### For Full Credit:

- Do **not** use the toArray method.
  - The purpose of the coding questions are to practice with the BagInterface, ListInterface, and List interfaces, not to work directly with arrays.
- Do not use a different data structure from the one in the question.
  - For example, in a coding question that uses BagInterface, do not use a List, ListInterface, ArrayList, array, etc..
  - Post any questions about what data structures or classes you should and should not use to the discussion board.
  - Again, the reason for this is that I want you to get practice with these specific interfaces.
- You will be graded both on syntax (the code compiles) **and** semantics (the code accomplishes the task).

Write a complete method from the client perspective that determines if a

bag contains any duplicate items.

For full credit, the parameter bag should have the same contents before and after the method executes.

The method header is:

```
public static boolean containsDuplicates(BagInterface<String> wordBag)
```

### For Full Credit:

- Do **not** use another data structure, such as ArrayList or an array.
- Do **not** invoke the toArray() method.
- Note: you **can** use local objects of type BagInterface (initialized as ArrayBag or LinkedBag).

## Question 3

**13 / 13 pts**

Paste the complete containsDuplicates method here.

Your Answer:

```
// Home work question 3
public static boolean containsDuplicates(BagInterface<String> wordBag) {
    // parameters
    boolean hasDuplicates = false;

    // create tempWordBag
    BagInterface<String> tempWordBag = new ArrayBag<String>();

    // checks if wordBag is empty
    if (!wordBag.isEmpty()) {
        while (!wordBag.isEmpty()) {
        // removes item from word bag
            String word = wordBag.remove();

            // checks
            if (wordBag.getFrequencyOf(word) >= 1) {
                hasDuplicates = true;
            }

            // add word into tempWordBag
            tempWordBag.add(word);
        }
        // empty tempWordBag into wordBag
        while (!tempWordBag.isEmpty()) {
            wordBag.add(tempWordBag.remove());
```

```
        }
    }
    return hasDuplicates;
}
```

Write a complete method from the client perspective to create a new **ListInterface** object that contains all Strings on a wordList that contain a given character.

The method header is:

```
public static ListInterface<String> createListInterface
ContainingChar(ListInterface<String> wordList, char tar
getChar)
```

### For Full Credit:

- The relative order of the names in the returned list should match the order of the names in the parameter list.
- The wordList parameter should be unchanged when the method completes.
- Consider case/capitalization. (In other words, 'c' is different from 'C'.)
- Do **not** invoke the toArray() method.
- Do **not** use another data structure, such as ArrayList or an array.
  - Note: You **can** use local objects of type ListInterface (initialized as AList or LList).

## Question 4

**10 / 10 pts**

Paste the complete createListInterfaceContainingChar method here.

Your Answer:

```
// method question 4
public static ListInterface<String> createListInterfaceContainingChar(ListI
nterface<String> wordList, char targetChar){
    // variables
```

```
    // new tempWordList
    ListInterface<String> tempWordList = new AList<String>();

    // loop through wordList
    for (int i = 1; i <= wordList.getLength(); i++) {

        // if targetChar in word, append word to tempWordList
        if(wordList.getEntry(i).indexOf(targetChar) != -1) {
            tempWordList.add(wordList.getEntry(i));
        }
    }
    // return
    return tempWordList;
}
```

Write a complete method from the client perspective to find the last position of an item in a **List**.

The method header is:

```
public static int lastPosition(List<String> wordList, S
tring target)
```

### For Full Credit:

- The wordList parameter should be unchanged when the method completes.
- If the item appears more than once, return the position of the **last** appearance.
- Decide what to do if the item is not on the list. Make sure your action is logical.
- Do not use the lastIndexOf method from the ArrayList class.
  - This question is essentially asking you to re-write this method for yourself.
  - You will receive 0 points if your method invokes lastIndexOf.

## Question 5

**10 / 10 pts**

Paste your complete lastPosition method here.

Your Answer:

```
// method question 5
public static int lastPosition(List<String> wordList, String target) {
    // parameters
    int targetIndex = -1;

    // create a reverseWordList
    List<String> reverseWordList = new ArrayList<String>();

    // for loop highest to lowest index
    for (int i = (wordList.size() - 1); i >= 0; i--) {
        if (wordList.get(i).equals(target)){
            targetIndex = i;
            break;
        }
    }
    return targetIndex;
}
```

Write a complete method from the client perspective to determine if a ListInterface and List of numbers have equivalent contents- meaning the same numbers in the same order.

The method header is:

```
public static boolean equivalentLists(ListInterface<Integ
er> numberListInterface, List<Integer> numberList)
```

## For Full Credit:

- Neither parameter list should be changed when the method completes.
- In order to be considered equivalent, the two lists should be the same size. Be sure to account for special sizes (e.g., 0 and 1).
- Do **not** convert either list to an array.
- Do **not** use another data structure, such as ArrayList, AList, or an array.

## Question 6

**15 / 15 pts**

Paste your complete equivalentLists method here.

Your Answer:

```
// method question 6
public static boolean equivalentLists(ListInterface<Integer> numberListInte
rface, List<Integer> numberList) {
    // parameters
    boolean equalBoolean = true;

    // check for equal length
    if((numberListInterface.getLength() != numberList.size())) {
        equalBoolean = false;
    }else {

        // loop for comparing items
        for (int i = 0; i < numberList.size(); i++) {
            // base one index
            int baseOneIndex = (i + 1);
            // check
            if (numberList.get(i) != numberListInterface.getEntry(baseOneIn
dex)) {
                equalBoolean = false;
            }
        }
    }
    return equalBoolean;
}
```

note that it's best to always use the equals method, not == or !=, to compare objects for logical equivalence; even for Integer objects! for Integers, you can also convert to int and then safely use ==, like this:
list.get(i).getValue()==listInterface.getEntry(i+1).getValue)

# Short Answer Questions: List

These next set of questions ask about objects of type List (in the Java standard library).

- Carefully review the **API page (https://docs.oracle.com/javase/10/docs/api/java/util/List.html)** to see how the List methods work.

- Remember that List objects start at position 0!
- Assume **all** methods in the List interface **are** implemented.
- Unless otherwise specified, assume a list is initially empty.

## Question 7

**1 / 1 pts**

wordList is type List<String>.

What is true about the code below?

```
wordList.add("sun");
wordList.add("tree");
wordList.add("bird");
wordList.add("ocean");
for(int i=0; i<wordList.size(); i++) {
     System.out.println(wordList.get(i));
}
```

- there will be a compiler error
- there will be a runtime error (an exception thrown)

**Correct!**

- (selected) there are no errors

## Question 8

**1 / 1 pts**

wordList is type List<String>.

What is printed?

```
wordList.add("apple");
wordList.add("banana");
wordList.add("cat");
```

```
wordList.add(2, "dog");
System.out.println(wordList.size());
```

- none of these is correct
- 5
- 3
- 0
- 1
- 2
- you cannot tell from the provided code

**Correct!**

- (selected) 4

## Question 9

**1 / 1 pts**

wordList is type List<String>.

What is printed?

```
wordList.add("apple");
wordList.add("banana");
wordList.add("cat");
wordList.add(2, "dog");
System.out.println(wordList.get(3));
```

- an exception will be thrown
- apple

- null
- you cannot tell from the provided code
- none of these is correct
- dog

**Correct!**

- cat
- banana

## Question 10

**1 / 1 pts**

wordList is type List<String>.

What is printed?

```
wordList.add("apple");
wordList.add("banana");
wordList.add("cat");
wordList.add("dog");
System.out.println(wordList.get(4));
```

- false
- true
- cat
- null
- banana
- apple

Correct!

- (•) an exception will be thrown
- ( ) dog
- ( ) you cannot tell from the provided code

## Question 11

**2 / 2 pts**

wordList is type List<String>.

What is the position of each word added to the list using the code below?

```
wordList.add("cat");
wordList.add("frog");
wordList.add(2, "pumpkin");
wordList.add(1, "house");
```

Correct! **cat** — 0

Correct! **frog** — 2

Correct! **pumpkin** — 3

Correct! **house** — 1

Other Incorrect Match Options:

- 4
- 5

## Question 12

1 / 1 pts

wordList is type List<String>.

What is printed?

```
wordList.add("house");
wordList.add("yard");
wordList.add("car");
wordList.add("street");
System.out.println(wordList.remove(2));
```

- you cannot tell from the provided code
- house
- none of these is correct
- an exception will be thrown
- yard
- street
- true

**Correct!**

- car
- false
- null

## Question 13

1 / 1 pts

wordList is type List<String>.

What is printed?

```
wordList.add("house");
wordList.add("yard");
wordList.add("car");
wordList.add("street");
System.out.println(wordList.remove(6));
```

- yard
- house

**Correct!**

- an exception will be thrown
- you cannot tell from the provided code
- street
- none of these is correct
- car
- true
- false
- null

## Question 14

**1 / 1 pts**

wordList is type List<String>.

What is printed?

```
wordList.add("car");
wordList.add("boat");
wordList.add("plane");
System.out.println(wordList.set(2, "train"));
```

- none of these is correct
- true
- false
- 2
- null

**Correct!**

- plane
- train
- you cannot tell from the provided code
- boat
- car
- an exception will be thrown

## Question 15

**1 / 1 pts**

wordList is type List<String>.

What is printed?

```
wordList.add("car");
wordList.add("boat");
wordList.add("plane");
```

```
System.out.println(wordList.set(6, "train"));
```

- car
- false
- none of these is correct
- null
- boat
- train
- plane
- true
- you cannot tell from the provided code

**Correct!**

- an exception will be thrown

## Question 16

**1 / 1 pts**

wordList is type List<String>.

True of False: When the code completes, the list is empty.

```
wordList.add("apple");
wordList.add("banana");
wordList.add("carrot");
... more adds (not shown)
wordList.add("watermelon");
for(int i=0; i<wordList.size(); i++) {
     wordList.remove(i);
}
```

- True

Correct!

- False

## Question 17

0 / 1 pts

wordList is type List<String>.

What is true about the code below?

```
wordList.add("sun");
wordList.add("tree");
wordList.add("bird");
wordList.add("ocean");
for(String word : wordList) {
     System.out.println(word);
}
```

ou Answered

- there will be a compiler error

- there will be a runtime error (an exception thrown)

orrect Answer

- there are no errors

# Short Answer Questions: BagInterface

These next set of questions ask about objects of type BagInterface.

- Refer to that **interface** (https://ccsf.instructure.com/courses/47904/files/7254571/download?download_frd=1) for details.

- Unless otherwise specified, assume a bag is initially empty.
- Remember that you do **not** know how a BagInterface is implemented- you only know that the object is type **BagInterface**.
  - You can only rely on how behavior is described in the interface. If the interface doesn't specify something, you cannot assume it about BagInterface objects.
  - You don't know what the actual type is. It could be ArrayBag, LinkedBag, or some other implementing class.
  - **Don't rely on running test code with ArrayBag or LinkedBag to test behavior that isn't defined in the interface.** Other implementing classes could show different behavior if the behavior isn't specifically described in the interface.

## Question 18

**1 / 1 pts**

wordBag is type BagInterface<String>.

What is printed?

```
wordBag.add("apple");
wordBag.add("banana");
wordBag.add("dog");
wordBag.add("cat");
System.out.println(wordBag.getCurrentSize());
```

**Correct!**

- (selected) 4
- none of these is correct
- 5
- 3
- you cannot tell from the provided code

## Question 19

1 / 1 pts

wordBag is type BagInterface<String>.

What is printed?

```
wordBag.add("apple");
wordBag.add("banana");
wordBag.add("apple");
wordBag.add("cat");
wordBag.remove("apple");
System.out.println(wordBag.getCurrentSize());
```

**Correct!**

- (selected) 3
- 4
- you cannot tell from the provided code
- none of these is correct
- 2
- 1
- 5

## Question 20

1 / 1 pts

wordBag is type BagInterface<String>.

What is printed?

```
wordBag.add("apple");
wordBag.add("frog");
```

```
wordBag.add("banana");
wordBag.add("frog");
wordBag.add("elephant");
System.out.println(wordBag.getFrequencyOf("frog"));
```

- you cannot tell from the provided code
- true
- 3
- null
- 5
- 1
- false

Correct!

- 2
- none of these is correct
- 4

## Question 21

1 / 1 pts

wordBag is type BagInterface<String>.

What is printed?

```
wordBag.add("apple");
wordBag.add("dog");
wordBag.add("cat");
wordBag.add("banana");
System.out.println(wordBag.contains("cat"));
```

- 3
- 4
- none of these is correct
- 1
- 0
- false
- 2

**Correct!**

- true
- you cannot tell from the provided code

## Question 22

**1 / 1 pts**

wordBag is type BagInterface<String>.

What is printed?

```
wordBag.add("apple");
wordBag.add("frog");
wordBag.add("banana");
wordBag.add("frog");
wordBag.add("elephant");
System.out.println(wordBag.getFrequencyOf("giraffe"));
```

Correct!

- (•) 0
- ( ) 1
- ( ) false
- ( ) null
- ( ) true
- ( ) -1
- ( ) none of these is correct
- ( ) you cannot tell from the provided code

## Question 23

1 / 1 pts

wordBag is type BagInterface<String>.

What is printed?

```
wordBag.add("apple");
wordBag.add("dog");
wordBag.add("cat");
wordBag.add("banana");
System.out.println(wordBag.remove());
```

- ( ) apple

Correct!

- (•) you cannot tell from the provided code
- ( ) an exception will be thrown
- ( ) banana

- dog
- cat
- none of these is correct
- null

## Question 24

**1 / 1 pts**

wordBag is type BagInterface<String>.

What is printed?

```
wordBag.add("apple");
wordBag.add("banana");
wordBag.add("cat");
wordBag.add("frog");
System.out.println(wordBag.remove("banana"));
```

- you cannot tell from the provided code
- 1
- null
- none of these is correct
- 3

**Correct!**

- true
- false
- banana

## Question 25

**1 / 1 pts**

wordBag is type BagInterface<String>.

What is printed?

```
wordBag.add("apple");
wordBag.add("banana");
wordBag.add("cat");
```

```
wordBag.add("frog");
System.out.println(wordBag.remove("melon"));
```

- none of these is correct
- 3
- null

Correct!

- (•) false
- 0
- true
- melon
- you cannot tell from the provided code

## Question 26

1 / 1 pts

wordBag is type BagInterface<String>.

What is printed?

```
wordBag.add("apple");
wordBag.add("banana");
wordBag.clear();
System.out.println(wordBag.remove());
```

- true

Correct!

- (•) null

- 0
- an exception will be thrown
- you cannot tell from the provided code
- none of these is correct
- false
- -1

## Question 27

**1 / 1 pts**

wordBag is type BagInterface<String>.

What is true about the code below?

```
wordBag.add("bird");
wordBag.add("tree");
wordBag.add("bird");
while(wordBag.contains("bird")) {
     wordBag.remove("bird");
}
```

- there is a compiler error

**Correct!**

- none of these is correct
- there will be a runtime error (an exception thrown)
- there will be an infinite loop

## Question 28

1 / 1 pts

wordBag is type BagInterface<String>.

What is true about the code below?

```
wordBag.add("bird");
wordBag.add("tree");
wordBag.add("dog");
while(!wordBag.isEmpty()) {
     wordBag.remove("cat");
}
```

- there will be a runtime error (an exception thrown)
- there is a compiler error

**Correct!**

- (selected) there will be an infinite loop
- none of these is correct

# Short Answer Questions: ListInterface

These next set of questions ask about objects of type ListInterface.

- Refer to that **interface** (https://ccsf.instructure.com/courses/47904/files/7254571/download?download_frd=1) for details.
- Remember that ListInterface objects start at position 1.
- Unless otherwise specified, assume a list is initially empty.

## Question 29

1 / 1 pts

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("apple");
wordList.add("banana");
wordList.add("cat");
wordList.add(2, "dog");
System.out.println(wordList.getLength());
```

- 0
- 2
- 4 **Correct!**
- none of these is correct

- 3
- 5
- you cannot tell from the provided code
- 1

## Question 30

**1 / 1 pts**

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("adriana");
wordList.add("bob");
wordList.add("frank");
wordList.add("edna");
System.out.println(wordList.getEntry(2));
```

- false

**Correct!**

- bob
- you cannot tell from the provided code
- null
- frank
- adriana
- an exception will be thrown
- true

- frank
- none of these is correct

## Question 31

**1 / 1 pts**

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("apple");
wordList.add("banana");
wordList.add("cat");
wordList.add(2, "dog");
System.out.println(wordList.getEntry(3));
```

- null
- you cannot tell from the provided code
- cat
- none of these is correct

**Correct!**

- banana
- apple
- dog
- an exception will be thrown
- true
- false

## Question 32

**1 / 1 pts**

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("adriana");
wordList.add("bob");
wordList.add("frank");
wordList.add("edna");
System.out.println(wordList.getEntry(0));
```

- bob
- edna
- frank
- null
- you cannot tell from the provided code
- adriana
- none of these is correct

**Correct!**

- an exception will be thrown

## Question 33

**2 / 2 pts**

wordList is type ListInterface<String>.

What is the position of each word added to the list using the code below?

```
wordList.add("cat");
wordList.add("frog");
wordList.add(2, "pumpkin");
wordList.add(1, "house");
```

| | | |
|---|---|---|
| Correct! | cat | 2 |
| Correct! | frog | 4 |
| Correct! | pumpkin | 3 |
| Correct! | house | 1 |

Other Incorrect Match Options:

- 0
- 5

## Question 34

**1 / 1 pts**

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("house");
wordList.add("yard");
wordList.add("car");
wordList.add("street");
System.out.println(wordList.remove(2));
```

- you cannot tell from the provided code
- null
- house
- an exception will be thrown

- false
- car

Correct!

- yard
- street
- true

## Question 35

1 / 1 pts

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("house");
wordList.add("yard");
wordList.add("car");
wordList.add("street");
System.out.println(wordList.remove(5));
```

- null
- you cannot tell from the provided code
- house
- yard

Correct!

- an exception will be thrown
- false
- true

- car
- street

## Question 36

**1 / 1 pts**

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("car");
wordList.add("boat");
wordList.add("plane");
wordList.add("car");
wordList.add("bike");
wordList.remove(2);
System.out.println(wordList.getLength());
```

- 0
- 3
- 2
- 1

**Correct!**

- 4
- you cannot tell from the provided code
- 5
- none of these is correct

## Question 37

1 / 1 pts

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("car");
wordList.add("boat");
wordList.add("plane");
System.out.println(wordList.replace(2, "train"));
```

- car
- plane
- null
- false
- train
- true
- you cannot tell from the provided code
- none of these is correct

**Correct!**

- boat
- 2

## Question 38

1 / 1 pts

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("car");
wordList.add("boat");
wordList.add("plane");
System.out.println(wordList.replace(4, "train"));
```

- none of these is correct
- true
- boat
- car
- train
- null

**Correct!**

- an exception will be thrown
- you cannot tell from the provided code
- plane
- false

## Question 39

**1 / 1 pts**

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("sun");
wordList.add("tree");
wordList.add("bird");
```

```
wordList.add("ocean");
System.out.println(wordList.contains("bird"));
```

- null
- false
- 4
- an exception will be thrown
- you cannot tell from the provided code

**Correct!**

- true
- 1
- none of these is correct
- 0
- 3
- 2

## Question 40

**1 / 1 pts**

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("sun");
wordList.add("tree");
wordList.add("bird");
wordList.add("ocean");
System.out.println(wordList.contains("snow"));
```

- none of these is correct
- true
- you cannot tell from the provided code

**Correct!**

- false
- -1
- 0
- an exception will be thrown
- null

## Question 41

**1 / 1 pts**

wordList is type ListInterface<String>.

What is printed?

```
wordList.add("car");
wordList.add("boat");
wordList.add("plane");
wordList.clear();
System.out.println(wordList.isEmpty());
```

- 2
- null
- 0
- false

- an exception will be thrown
- you cannot tell from the provided code
- none of these is correct

**Correct!**

- (selected) true
- 1
- 3

## Question 42

**1 / 1 pts**

wordList is type ListInterface<String>.

What is true about the code below?

```
wordList.add("sun");
wordList.add("tree");
wordList.add("bird");
wordList.add("ocean");
for(int i=0; i<wordList.getLength(); i++) {
     System.out.println(wordList.getEntry(i));
}
```

**Correct!**

- (selected) there will be a runtime error (an exception thrown)
- there are no errors
- there will be a compiler error

## Question 43

**1 / 1 pts**

wordList is type ListInterface<String>.

What is true about the code below?

```
wordList.add("sun");
wordList.add("tree");
wordList.add("bird");
wordList.add("ocean");
for(String word : wordList) {
     System.out.println(word);
}
```

- there are no errors
- there will be a runtime error (an exception thrown)

**Correct!**

- there will be a compiler error

## Question 44

**0 / 1 pts**

wordList is type ListInterface<String>.

True or False: When the code completes, the list is empty.

```
wordList.add("apple");
wordList.add("banana");
wordList.add("carrot");
... more adds (not shown)
wordList.add("watermelon");
for(int i=1; i<=wordList.getLength(); i++) {
     wordList.remove(i);
}
```

ou Answered

- (•) True

rrect Answer

- ( ) False

# Optional Extra Credit (10 points)

Find the maximum value in a **List** and move that value to the front of the list.

The method header is:

```
public static void prioritizeMaximumValue(List<Integer>
numberList)
```

## Notes:

- If there is more than one copy of the maximum, you can move any one of those copies to the front.
  - Because there is more than one right answer here, the driver program does not include tests for this. I suggest adding your own tests.
- You are not swapping values- but performing a ***shift*** by moving the maximum value to the front of the list.

### For Full Credit:

- Take advantage of the methods provided in List rather than rewriting code to do the same task.
- Your code should not crash with empty lists or one-element lists.
- Do **not** use another data structure, such as ArrayList or an array.

## Question 45

**10 / 0 pts**

Paste your complete prioritizeMaximumValue method here.

Your Answer:

```
// extra credit problem
public static void prioritizeMaximumValue(List<Integer> numberList) {
    // check for list not equal to 0
    if (numberList.size() > 0) {

        // parameters
        int maxValue = numberList.get(0);
        System.out.println("maxvalue: " + maxValue);

        // loop for getting maxValue in list
        for (int i = 0; i < numberList.size(); i++) {
            // replace max value if bigger
            if(maxValue < numberList.get(i)) {
                maxValue = numberList.get(i);
            }
        }
        // add maxValue to front of List
        numberList.remove(numberList.indexOf(maxValue));
        numberList.add(0, maxValue);
    }
}
```

Quiz Score: **108** out of 100