420-B31

# Lab 7 – Wildcards, Basic Collection and List ADT Answers

**Part A – Generic Type Questions**

1. What error do you get when you try to call **whatAreYou** ()?

**The error that we get is an Unresolved compilation problem**

1. Why?

**The reason is because we had a collection of objects for our method, but our collection has Animal, Dog, Shape.**

1. Which collection failed for **talkNow**().

**The object, shape, and dog.**

1. Why?

**This is because, the method is a type animal, even though a dog inherits from Animal, you need to change the method to take extend animal.**

1. Complete the following table with the classes Object, Animal, Dog, Shape.

|  |  |
| --- | --- |
| **Parameter** | **Collection<Class> it works for** |
| Collection<Object> coll | **Object** |
| Collection<?> coll | **Object, Animal, Dog, Shape** |
| Collection<Animal> coll | **Animal** |
| Collection<? extends Animal> coll | **Animal, Dog** |
| Collection<E> coll | **Any type** |

**Part C Question1 – ListIterator**

## Complete the table of how lists work. Refer to the notes from class for an example. Remember that the object state for a List requires the head, tail, size, the actual list and the index of each element in the list.

| Method | **Object State** | **Returned Value** |
| --- | --- | --- |
| List<String> list = new LinkedList<String>() | **Size = 0**  **Head = null**  **Tail = null** | **A LinkedList** |
| list.add("peach") | **Size = 1**  **Head = Peach**  **Tail = Peach**  **“Peach”** |  |
| list.add(0,"apple") | **Size = 2**  **Head = “Apple”**  **Tail = “Peach”**  **“Apple””Peach”** |  |
| list.contains("orange") |  | **False** |
| list.isEmpty() |  | **False** |
| list.add(1,"banana") | **Size = 3**  **Head = “Apple”**  **Tail = “Peach”**  **“Apple””Banana””Peach”** |  |
| List<String> l2 = new ArrayList<String>(list) | **Size = 0**  **Head = null**  **Tail = null** |  |
| l2.add(0,"grape") | **Size = 4**  **Head = “Grape”**  **Tail = “Peach”**  **Grape, Apple, Banana, Peach** |  |
| l2.add(2,"pear") | **Size = 5**  **Head = “Grape”**  **Tail = “Peach”**  **Grape, Apple, Pear, Banana, Peach** |  |
| l2.remove("apple") | **Size = 4**  **Head = “Grape”**  **Tail = “Peach”**  **Grape, Pear, Banana, Peach** | **True** |
| l2.remove("grape") | **Size = 3**  **Head = “Pear”**  **Tail = “Peach”**  **Pear, Banana, Peach** | **True** |
| ListIterator<String> iter = list.listIterator() | **Size = 3**  **Head = “Apple”**  **Tail = “Peach”**  **Triangle Apple Banana Peach** | **An iterator object** |
| iter.hasNext() | **Apple Triangle Banana Peach** | **True** |
| String s = iter.next() | **Apple Banana Triangle Peach** | **Pear** |
| iter.add("date") | **Apple Triangle Date Banana Peach** | **True** |
| iter.remove() | **Size = 3**  **Head = “Date”**  **Tail = “Peach”**  **Triangle Date Banana Peach** | **False (Generates Exception Error)** |
| iter.hasNext() |  | **True** |
| list.remove("peach") | **Size = 2**  **Head = “Date”**  **Tail = “Banana”**  **Triangle Date Banana** |  |
| s = iter.next() | **Date Triangle Banana** | **Date** |

**Part D Question1 – Test Cases for LinkedList.reverse()**

**Test case 1**: Test LinkedList.reverse() for an empty list

|  |  |  |  |
| --- | --- | --- | --- |
| Method | **Purpose** | **Object State** | **Expected Result** |
| Reverse() | Reverse the elements of a list | List1  LinkedLinked:  null | NoSuchElementException e |

**Test case 2**: Test LinkedList.reverse() with a list with more than 2 elements

|  |  |  |  |
| --- | --- | --- | --- |
| Method | **Purpose** | **Object State** | **Expected Result** |
|  | Reverse the elements of a list | List1: | Fin, r,a,c,e,c,a,r |
| List.add(“R”)  List.add(‘a’)  List.add(‘c’)  List.add(‘e’) | Adding elements to the list | List1:  “E” “C” “A”  “R” | “E” “C” “A”  “R” |
| List.get(0)  List.get(1)  List.get(2)  List.get(3) |  | List1:  “E” “C” “A”  “R” |  |
| Reverse() | Reversing the elements of the list | List1:  “R” “A” “C” “E” | “R” “A” “C” “E” |
| List.get(0)  List.get(1)  List.get(2)  List.get(3) |  |  | “R” “A” “C” “E” |