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 method whatAreYou(Collection<Object>) in the type B31\_L07\_Generics is not applicable for the arguments (Collection<Animal>)

The method whatAreYou(Collection<Object>) in the type B31\_L07\_Generics is not applicable for the arguments (Collection<Dog>)

The method whatAreYou(Collection<Object>) in the type B31\_L07\_Generics is not applicable for the arguments (Collection<Shape>)

1. Why?

We get these errors because the whatAreYou method expects only an Object, not an Animal.

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

Object, dog and Shape

1. Why?

Because the method looks for a parameter specifically of type animal and nothing else.

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

|  |  |
| --- | --- |
| **Parameter** | **Collection<Class> it works for** |
| Collection<Object> coll | None |
| Collection<?> coll | All |
| Collection<Animal> coll | Animal |
| Collection<? extends Animal> coll | Animal, Dog |
| Collection<E> coll | Animal, dog |

**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** |  |
| list.add("peach") | **Size = 1**  **“peach”** | **True** |
| list.add(0,"apple") | **Size = 2**  **“apple”, “peach”** | **True** |
| list.contains("orange") | **No change** | **False** |
| list.isEmpty() | **No change** | **False** |
| list.add(1,"banana") | **Size = 3**  **“apple”, “banana”, “peach”** | **True** |
| List<String> l2 = new ArrayList<String>(list) | **Size = 0** | **New ArrayList** |
| l2.add(0,"grape") | **Size = 1**  **“grape”** | **True** |
| l2.add(2,"pear") | **Size = 3**  **“grape”, null, “pear”** | **True** |
| l2.remove("apple") | **No change** | **False** |
| l2.remove("grape") | **Size = 2**  **Null, “pear”** | **True** |
| ListIterator<String> iter = list.listIterator() | **^ Apple, banana, peach** |  |
| iter.hasNext() | **No change** | **True** |
| String s = iter.next() | **Apple, ^, banana, peach** | **Apple** |
| iter.add("date") | **Size = 4**  **Apple, date, ^, banana, peach** | **True** |
| iter.remove() | **Size = 3**  **Apple, date, ^, banana, peach** | **IllegalStateException** |
| iter.hasNext() | **No change** | **True** |
| list.remove("peach") | **Apple, date, ^, banana** | **True** |
| s = iter.next() | **No change** | **True** |

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| Method | **Purpose** | **Object State** | **Expected Result** |
|  |  |  |  |

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

|  |  |  |  |
| --- | --- | --- | --- |
| Method | **Purpose** | **Object State** | **Expected Result** |
|  |  |  |  |