420-B31

Lab 6 Answer Sheet

# Part A – Testing Review

Question 1 – BasicCollection Review

\*Recall for a collection that the Object State that is relevant is the size, the contents of the collection and the position of the iterator.

| **Method** | **New Object State** | **Returned Value** |
| --- | --- | --- |
| **Collection c = new BasicCollection<String>();** | c.size() = 0 | A basic collection of strings |
| **c.add("peach");** | c.size() = 1  “Peach” | True |
| **c.add("apple");** | c.size() = 2  “Peach””Apple” | True |
| **c.contains("orange");** |  | False |
| **c.isEmpty();** |  | False |
| **Collection c2 = new BasicCollection<String>();** | C2.size() = 0; | A basic collect of strings |
| **c2.add("peach");** | C2.size() = 1;  “Peach” | True |
| **c2.add("orange");** | C2.size() = 2  “Peach” “Orange” | True |
| **c2.containsAll(c);** |  | False |
| **c2.remove("apple");** | C2.size() = 2 | False |
| **c2.remove("peach");** | C2.size() = 1 | True |
| **Iterator<String> iter = c.iterator();** | Triangle”Peach””Apple” | Iterator |
| **iter.hasNext();** |  | True |
| **String s = iter.next();** | “Peach”Triangle”Apple” | “Peach” |
| **s = iter.next();** | “Peach””Apple”Triangle | “Apple” |
| **iter.remove();** | c.size() = 1  “Peach”Triangle | Void |
| **iter.hasNext();** |  | False |
| **c.remove("peach");** | c.size() = 0 | True |
| **s = iter.next();** | Triangle | Exception |

Question 2 – containsAll() test case

**Test Case 1: containsAll()** method – two empty collections

| **Operation** | **Purpose** | **Object**  **State** | **Expected**  **Result** |
| --- | --- | --- | --- |
| **Collection c1 = new BasicCollection<String>();** | To create an empty collection | c1.size = 0 | A BasicCollection object for Strings |
| **Collection c2 = new BasicCollection<String>();** | To create a second empty collection | C1.size = 0 | A BasicCollection object for Strings |
| **c1.containsAll(c2);** | To test if it c1 contains all the elements from c2 |  | true |

**Test Case 2: containsAll()** method – two identical non-empty collections

| **Operation** | **Purpose** | **Object**  **State** | **Expected**  **Result** |
| --- | --- | --- | --- |
| **Collection c1 = new BasicCollection<String>();** | To create an empty collection | c1.size = 0 | A BasicCollection object for Strings |
| **c1.add("A");** | To add to the collection | c1.size = 1  "A" | true |
| **c1.add("B");** | To add to the collection | c1.size = 2  "A" "B" | true |
| **c1.add("C");** | To add to the collection | c1.size = 3  "A" "B" "C" | true |
| **Collection c2 = new BasicCollection<String>();** | To create a second empty collection | c2.size = 0 | A BasicCollection object for Strings |
| **C2.add("A");** | To add to the collection | c1.size = 1  "A" | true |
| **C2.add("B");** | To add to the collection | c1.size = 2  "A" "B" | true |
| **C2.add("C");** | To add to the collection | c1.size = 3  "A" "B" "C” | true |
| **C1.containsAll(c2)** |  |  | true |

**Test Case 3: containsAll()** method –two non-empty collections – second collection a subset of the first

| **Operation** | **Purpose** | **Object**  **State** | **Expected**  **Result** |
| --- | --- | --- | --- |
| **Collection c1 = new BasicCollection<String>();** | To create an empty collection | c1.size = 0 | A BasicCollection object for Strings |
| **c1.add("A");** | To add to the collection | c1.size = 1  "A" | true |
| **c1.add("B");** | To add to the collection | c1.size = 2  "A" "B" | true |
| **c1.add("C");** | To add to the collection | c1.size = 3  "A" "B" "C" | true |
| **Collection c2 = new BasicCollection<String>();** | To create a second empty collection | C2.size = 0 | A BasicCollection object for Strings |
| **C2.add("A");** | To add to the collection | C2.size = 1  "A" | True |
| **C2.add("B");** | To add to the collection | C2.size = 2  "A" "B" | true |
| **C2.containsAll(c1)** |  |  | False |

**Test Case 4: containsAll()** method – two non-empty collections with the same number of elements, but different values

| **Operation** | **Purpose** | **Object**  **State** | **Expected**  **Result** |
| --- | --- | --- | --- |
| **Collection c1 = new BasicCollection<String>();** | To create an empty collection | c1.size = 0 | A BasicCollection object for Strings |
| **c1.add("A");** | To add to the collection | c1.size = 1  "A" | true |
| **c1.add("B");** | To add to the collection | c1.size = 2  "A" "B" | true |
| **c1.add("C");** | To add to the collection | c1.size = 3  "A" "B" "C" | true |
| **Collection c2 = new BasicCollection<String>();** | To create a second empty collection | C2.size = 0 | A BasicCollection object for Strings |
| **C2.add("D");** | To add to the collection | C2.size = 1  "D" | true |
| **C2.add("E");** | To add to the collection | C2.size = 2  "D" "E" | true |
| **C2.add("F");** | To add to the collection | C2.size = 3  “D” “E” “F” | true |
| **C2.containsAll(c1)** |  |  | false |

**Test Case 5: containsAll()** method – two non-empty collections with the first a subset of the second

| **Operation** | **Purpose** | **Object**  **State** | **Expected**  **Result** |
| --- | --- | --- | --- |
| **Collection c1 = new BasicCollection<String>();** | To create an empty collection | c1.size = 0 | A BasicCollection object for Strings |
| **c1.add("A");** | To add to the collection | c1.size = 1  "A" | true |
| **c1.add("B");** | To add to the collection | c1.size = 2  "A" "B" | true |
| **Collection c2 = new BasicCollection<String>();** | To create a second empty collection | C2.size() = 0 | A BasicCollection object for Strings |
| **C2.add("A")** | To add to the collection | C2.size = 1  "A" | True |
| **C2.add("B")** | To add to the collection | C2.size = 2  "A" "B" | True |
| **C2.add("C")** | To add to the collection | C2.size = 3  “A””B””C” | true |
| **C2.containsAll(c1)** |  |  | true |