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 | BasicCollection of string objects |
| **c.add("peach");** | c.size = 1  “peach” | True |
| **c.add("apple");** | c.size = 2  “peach”, “apple” | True |
| **c.contains("orange");** | No change | False |
| **c.isEmpty();** | No change | False |
| **Collection c2 = new BasicCollection<String>();** | C2.size = 0 | BasicCollection of string objects |
| **c2.add("peach");** | C2.size = 1  “peach” | True |
| **c2.add("orange");** | C2.size = 2  “peach”, “orange” | True |
| **c2.containsAll(c);** | No change | False |
| **c2.remove("apple");** | No change | False |
| **c2.remove("peach");** | C2.size = 1  “orange” | True |
| **Iterator<String> iter = c.iterator();** | c.size = 2  ^ “peach”, “apple” | An iterator object |
| **iter.hasNext();** | No change | True |
| **String s = iter.next();** | “peach” ^ “apple” | “peach” |
| **s = iter.next();** | “peach” “apple” ^ | “apple” |
| **iter.remove();** | c.size = 1  “peach” ^ | True |
| **iter.hasNext();** | No change | False |
| **c.remove("peach");** | c.size = 0 | True |
| **s = iter.next();** | No change | 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.containsAll(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 | 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 |
| **C1.containsAll(c2);** | Test the containsAll method with two identical lists |  | 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 = 0  “A” “B” | True |
| **C1.containsAll(c2);** | Test the two collections where c2 is a subset of c1 |  | True |

**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 |
| **C1.containsAll(c2);** | Test the containsAll method for two lists of the same size with different contents |  | 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 |
| **C1.containsAll(c2);** | Test the containsAll method for two collections, the first one being a subset of the second |  | False |