University of Bahrain

College of Information Technology

Department of Computer Science

First Semester, 2018-2019

**ITCS214 / ITCS215 / ITCS216 (Data Structures)**

**Tutorial 2**

**Question1 (A)** Write a generic class called **KWArrayList**having following data fields (private):

private static final int INITIAL\_CAPACITY = 10; // The default initial capacity

private E[] theData; // The underlying data array

private int size ; // The current size

private int capacity ; // The current capacity \*/

This class will have the following methods:

|  |  |
| --- | --- |
| **Method** | **Behavior** |
| public KWArrayList ( ) | Default Constructor with capacity = INITIAL\_CAPACITY |
| public KWArrayList (int cap ) | Constructor with capacity = cap |
| **public int size()** | Returns current size |
| **public boolean contains(E obj)** | Checks whether the given object obj is present in the list. If it is there then it returns true else it returns false. |
| **public void clear()** | Remove all the elements of the array list and make it empty |
| public boolean isEmpty() | Checks whether list is empty or not |
| public boolean add ( E anEntry) | Adds object anEntry at the end of the list and returns true. |
| public void add (int index, E anEntry) | Adds object anEntry in the list at the location given by index |
| public E get (int index) | Returns the element of the list at position given by index |
| public E set (int index, E newValue) | Updates the element at position index by newValue and returns the old value |
| public E remove (int index) | Removes the element at position index and returns the element being removed |
| public boolean remove (E obj) | Removes the first occurrence of the object obj from the list, if present and returns true, else returns false. |
| private void reallocate () | Private method to expand the array by allocation a new array of double the previous capacity. Called if the list becomes full |
| public int indexOf(E obj) | Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. |
| public String toString() | Returns the String equivalent of the list object |

**(B)** Write a class called ArrayListTest having only main method to test all functionalities.

**Question 2** Using the JCF **List** interface and **ArrayList** class of Java, write a java program having only main method. Create an object **list** of type **ArrayList** to store Strings. The user can enter as many Strings as the user wants to add to the **list** object. When the user wants to end, the user presses *ctrl z.* Use method **hasNext** of class **Scanner** to write the loop; display the contents of the **list** of strings with their indices.

Next, the user will enter two elements that should be swapped. If both elements are found in the list, swap these two elements in the **list** and output modified **list**, else output an error message.

**Example:**

The list contents:

1. Ali
2. Yousif
3. Mohamed
4. Sarah
5. Manal

Enter elements to be swapped: Yousif Manal

The list contents:

1. Ali
2. Manal
3. Mohamed
4. Sarah
5. Yousif

**Question 3** Write following methods to be included in class KWArrayList of Question 1:

1. **removeAll**: Removes all occurrences of the object *obj* from the list, if present and returns true, else returns false.

Method heading: **public boolean removeAll(E obj);**

1. **equals**: Returns true if “this list” and parameter list contains the same elements in the same order, else returns false.

Method heading: **public boolean equals (KWArrayList<E> list);**

1. **Copy constructor:** creates a copy of the list object of type KWArrayList

Method heading: **public KWArrayList ( KWArrayList<E> list);**

**Solution (A):**

public boolean removeAll(E obj)

{

boolean found = false;

int index = 0;

while(index < size)

{

// searching obj

if (obj.equals(theData[index]))

{

found = true; // obj found

// shifting elements in the array to delete the element at index.

// Can also be done by calling remove(index);

for (int i = index + 1; i < size; i++)

theData[i-1] = theData[i];

theData[size - 1] = null;

size--;

}

else

Index++;

}

return found;

}

**Solution (B):**

public boolean equals(KWArrayList<E> list)

{

if(this.size != list.size)

return false;

int index;

for(index = 0; index < size; index++)

{

if(!(this.theData[index]).equals(list.theData[index])))

return false;

}

return true;

}

**Solution (C):**

// Copy constructor is used to create a new object as a copy of an existing object of the

// same type.

// Parameter list is the original object and copy is made in “this object”.

public KWArrayList( KWArrayList<E> list)

{

this.capacity = list.capacity;

this.size = list.size;

// Allocating memory space for the array of the copy object

this.theData = (E[]) new Object[capacity];

// copying the elements of the array

for(int i = 0; i < size; i++)

this.theData[i] = list.theData[i];

}

**Question 4** Write following methods to be included in a class called **ArrayListEx.** Assume that class **ArrayList** is available for use.

1. **oddEven** that accepts three objects, **list1**, **list2** and **list3** of type **ArrayList** as parameters*.* Assume that **list1** is a list of integers and **list2** and **list3** are initially empty. The purpose of the method is to insert all odd integers of **list1** in**list2** and all even integers of **list1** in **list3**.

public class ArrayListEx

{

public static void oddEven(ArrayList<Integer> list1,

ArrayList<Integer> list2, ArrayList<Integer> list3)

{

1. **CompareSum** that compares the sum of the elements of the first half of the list list1, passed as parameter, with the sum of the elements of the second half of the list; if they are equal, returns true, otherwise, it returns false. Assume that the list contains integers and size of the list is even.

Method heading:

public static boolean CompareSum(ArrayList<Integer> list1);

**(C)** **deleteValues** that accepts two parameters, namely, **list1** of type **ArrayList** and **item** of type **int**. The method deletes all values from **list1**, which are less than or equal to **item**, starting from first occurrence of **item** in **list1**. If **list1** is empty or **item** is not found in **list1** then the method will not delete any values from **list1** and returns **false**, else it returns **true**.

**Example**: If **list1** before calling method **deleteValues** is:

3 10 4 5 3 2 8 4 6 1

And **item** is **4**, then **list1** after method call will be:

3 10 5 8 6

(Note that values deleted are underlined in the original list).

Method heading:

public static boolean deleteValues(ArrayList <Integer> list1,

int item);

**Solution (A) and (C):**

public class ArrayListEx

{

// Solution (A)

public static void oddEven(ArrayList<Integer> list1,

ArrayList<Integer> list2, ArrayList<Integer> list3)

{

int length = list1.size();

for(int i = 0; i < length; i++)

{

int item = list1.get(i);

if (item%2 == 1) // item is odd

list2.add(item);

else

list3.add(item);

}

}

// Solution (C)

public static boolean deleteValues(ArrayList <Integer>

list1, int item)

{

if(list1.isEmpty()

return false;

int loc = list1.indexOf(item); //search item in list1

if (loc == -1) // item not found

return false;

while (loc < list1.size())

{

int value = list1.get(loc);

if (value <= item)

list1.remove(item);

else

loc++;

}

return true;

}

**Question 5** Write a method called **merge** in a class called **MergeLists**. The method accepts three **ArrayList** objects **list1, list2** and **list3** of type **ArrayList** as parameters. Assume that **list1** and **list2** are sorted in ascending order. The **list3** is to be generated by merging the **list1** and **list2** by the method, in such a way that it is also sorted in ascending order.

Note that the function returns *false*, if both lists **list1** and **list2** are empty, else returns *true* after merging.

Example:

list1: 5 10 15 20 25 30 35 40

list2: 2 3 12 22 32 42 52

list3: 2 3 5 10 12 15 20 22 25 30 32 35 40 42 52

(generated by the function)

public class MergeLists

{

public static<E> boolean merge(ArrayList<E> list1,

ArrayList<E> list2, ArrayList<E> list3)

{

**Solution:**

public class MergeLists

{

public static<E> boolean merge(ArrayList<E> list1,

ArrayList<E> list2, ArrayList<E> list3)

{

if(list1.isEmpty() && list2.isEmpty())

{

System.out.println(“Both lists are empty”);

return false;

}

int n1 = list1.size(), n2 = list2.size();

int i = 0, j = 0, k;

E item1, item2;

while (i < n1 && j < n2)

{

item1 = list1.get(i);

item2 = list1.get(j);

if((Comparable)item1.compareTo((Comparable)item2) < 0)

{

list3.add(item1);

i++;

}

else

{

list3.add(item2);

j++;

}

} // end while

for(k = i; k < n1; k++)

list3.add(list1.get(k));

for(k = j; k < n2; k++)

list3.add(list2.get(k));

return true;

}

}

**Question 6 (A)** Write a method called **listChange** to be included in the **KWArrayList** class of Question 1 that accepts one parameter **list1** of type **KWArrayList** class. The method checks, if **list1** is sorted in ascending order. If so, it will delete the first and last elements of "this" list, otherwise, it will copy the first element from **list1** and add it to be the last element in "this" list. The method will return false, if **list1** or "this" list has less than two elements, otherwise it will return true.

Method heading:

**public boolean listChange(KWArrayList list1);**

**(B)** Write a class called **Test** that has the main method to test the **listChange** method of part(A).

**Solution:**

public boolean listChange(KWArrayList list1)

{

if (list1.size < 2 || this.size < 2)

return false;

boolean flag = true;

for (int i = 0; i < size-1 && flag; i++)

if(((Comparable)list1.theData[i]).compareTo

((Comparable)list1.theData[i+1]) > 0)

{

flag = false; // Not sorted in ascending order

break;

}

if (flag)

{

remove(0);

remove(size-1);

return true;

}

else

{

theData[size] = list1.theData[0];

size++;

return true;

}

}

public class Test

{

public static void main(String[] args)

{

KWArrayList <Integer> list1 =

new KWArrayList<Integer>();

list1.add(5);

list1.add(10);

list1.add(30);

list1.add(50);

KWArrayList <Integer> list2 =

new KWArrayList<Integer>();

list2.add(24);

list2.add(55);

list2.add(26);

list2.listChange(list1);

// only 55 should be printed.

for (int i = 0; i < list2.size(); i++)

System.out.println("Element: " + list2.get(i));

}

}