

CSE 12 – Basic Data Structures and Object-Oriented Design

Lecture 4

Greg Miranda & Paul Cao, Winter 2021

Announcements

- Quiz 4 due Wednesday @ 8am
- PA1 due Wednesday @ 11:59pm

Topics

- Lecture 4 Exercises
- Implement ArrayList Insert/Remove

```
public interface StringList {  
  
    /* Add an element at the end of the list */  
    void add(String s);  
  
    /* Get the element at the given index */  
    String get(int index);  
  
    /* Get the number of elements in the list */  
    int size();  
  
    /* Add an element at the specified index */  
    void insert(int index, String s);  
  
    /* Remove the element at the specified index */  
    void remove(int index);  
  
}
```

During the pre-lecture recording, we commented out insert and remove method. Why?

- A. We didn't plan to implement them at that time and commenting out them will make our code cleaner
- B. We didn't plan to implement them and commenting them out will avoid a compiler error
- C. We were overloading those two methods
- D. None of the above

In the `ArrayStringList` class, we have the following fields

```
String[] elements;
```

```
int size;
```

What's the point of having `size` as instance variable as the array elements already has size?

- A. It is duplicate information for ease of use
- B. It avoid calling `element.length` to save time
- C. `size` indicates how full the array is
- D. More than one of the above is correct

In the ArrayList class, we have a private helper method expandCapacity

```
private void expandCapacity() {  
    int currentCapacity = this.elements.length;  
    if(this.size < currentCapacity) { return; }  
  
    String[] expanded = new String[currentCapacity * 2];  
  
    for(int i = 0; i < this.size; i += 1) {  
        expanded[i] = this.elements[i];  
    }  
  
    this.elements = expanded;  
}
```

If I have a foo function inside the ArrayList class and have the following code what will be printed out? Assume that the array starts empty and has a capacity of 2.

```
public void foo(){  
    String[] tmp = elements;  
    add("a"); add("b"); add("c");  
    expandCapacity();  
    System.out.println(tmp == elements);  
}
```

- A. true
- B. false
- C. there will be a compiler error
- D. there will be a runtime error

In the ArrayList class, we have a private helper method expandCapacity

```
private void expandCapacity() {  
    int currentCapacity = this.elements.length;  
    if(this.size < currentCapacity) { return; }  
  
    String[] expanded = new String[currentCapacity * 2];  
  
    for(int i = 0; i < this.size; i += 1) {  
        expanded[i] = this.elements[i];  
    }  
  
    this.elements = expanded;  
}
```

When do I need to call this expandCapacity function?

- A. Inside the constructors
- B. Inside the insert method
- C. Inside the remove method
- D. Inside the get method
- E. Inside the add method

```
public void testAdd() {  
    StringList slist = new ArrayList();  
    slist.add("paul");  
    slist.add("greg");  
  
    assertEquals("paul", slist.get(0));  
    assertEquals("greg", slist.get(1));  
}
```

In our tester for add, we wrote the code for inserting two elements and test if we added properly. Can I write my tester as

```
assertEquals(slist.get(0), "paul");  
assertEquals(slist.get(1), "greg");
```

- A. Yes they are basically the same as what we wrote in pre-lecture video
- B. No you can't switch the order as it will generate the wrong test result
- C. No you can't switch the order as it makes the interpretation of the test result inaccurate

StringList Interface

```
public interface StringList {  
  
    /* Add an element at the end of the list */  
    void add(String s);  
  
    /* Get the element at the given index */  
    String get(int index);  
  
    /* Get the number of elements in the list */  
    int size();  
  
    /* Add an element at the specified index */  
    void insert(int index, String s);  
  
    /* Remove the element at the specified index */  
    void remove(int index);  
  
}
```

ArrayList Insert

```
/* Add an element at the specified index */  
void insert(int index, String s);
```

- Write a test case for the ArrayList insert method
- Implement the ArrayList insert method

ArrayList Remove

```
/* Remove the element at the specified index */  
void remove(int index);
```

- Write a test case for the ArrayList remove method
- Implement the ArrayList remove method