CSE 12 – Basic Data Structures and Object-Oriented Design Lecture 4

Greg Miranda & Paul Cao, Winter 2021

This lecture is being recorded

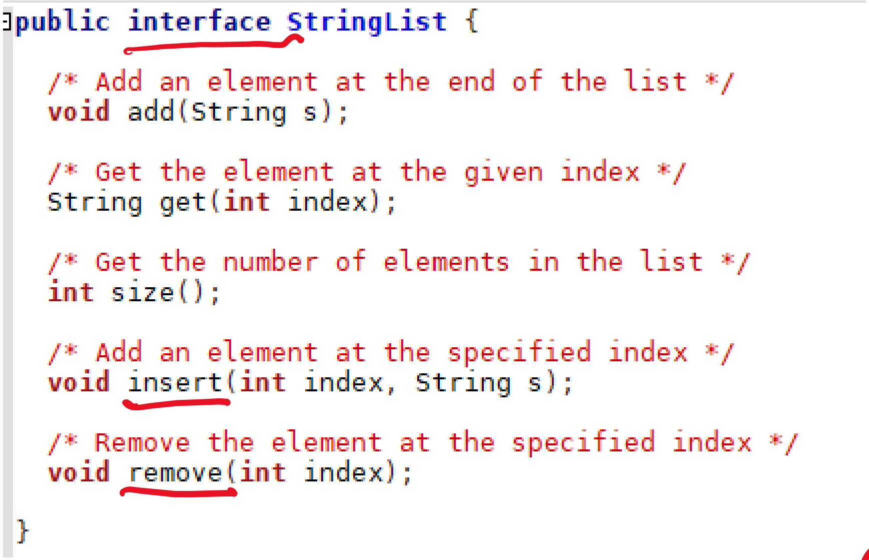
Announcements

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

Survey 2 due Friday

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

17

[✓ B.] We didn't plan to implement them and commenting them out will avoid a compiler error

64

~~C.~~ We were overloading those two methods

0

~~D.~~ None of the above

3

In the ArrayStringList class, we have the following fields

```
String[] elements;
int size;
```

→ # of elements inside the ArrayStringList

↑

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

length

A. It is duplicate information for ease of use 3
B. It avoid calling element.length to save time 1
[✓ C.] size indicates how full the array is 64
D. More than one of the above is correct 25
~~e~~ 1

length = 2

null	null

size = ~~0~~ 1

"apple"	null

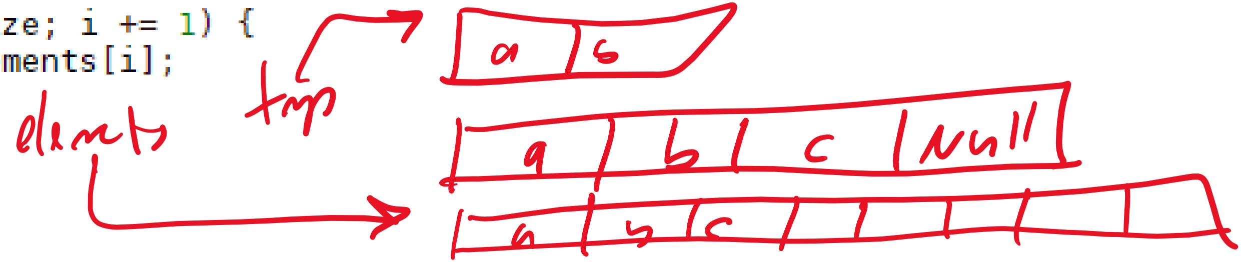
In the ArrayStringList class, we have a private helper method expandCapcity

```java
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 ArrayStringList class and have the following code what will be printed out? Assume that the array starts empty and has a capacity of 2.

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

?

8 A. true

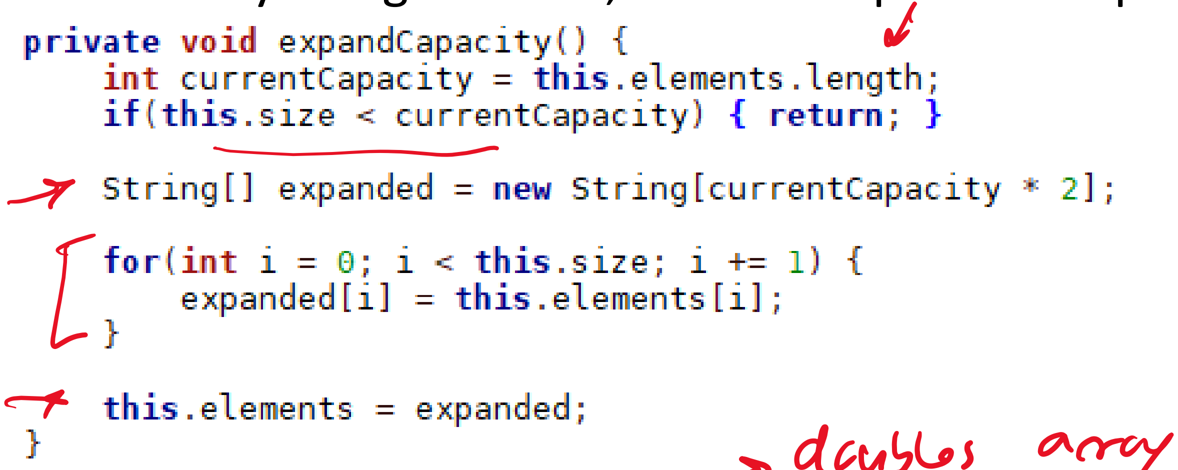
24 [✓ B. false]

21 ~~C.~~ there will be a compiler error

33 ~~D.~~ there will be a runtime error

2

In the ArrayStringList class, we have a private helper method expandCapcity



```
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;
}
```

doubles array size

When do I need to call this expandCapacity function?

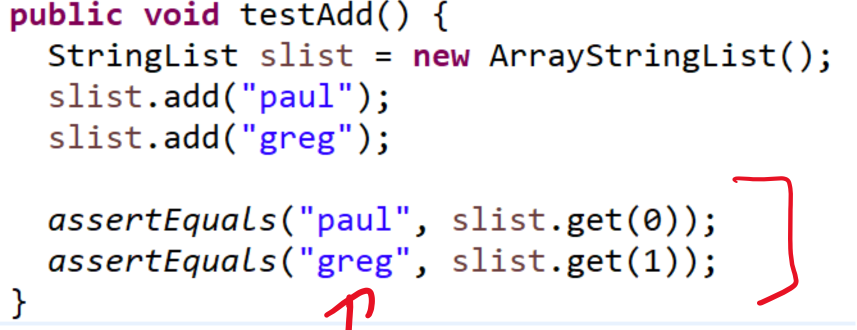
~~A.~~ Inside the constructors 6

[✓ B.] Inside the insert method 90

~~C.~~ Inside the remove method 1

~~D.~~ Inside the get method 0

~~E. Inside the add method~~



```
public void testAdd() {
  StringList slist = new ArrayStringList();
  slist.add("paul");
  slist.add("greg");

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

assertEquals(exp, actual)

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”);

Exp <?> —— Actual <? >

58 A. Yes they are basically the same as what we wrote in pre-lecture video

9 ~~B.~~ No you can’t switch the order as it will generate the wrong test result

29 [✓ C.] No you can’t switch the order as it makes the interpretation of the test result inaccurate

StringList Interface

```java
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