CSE 12: Programming Assignment 2

4-9-21

Focus: PA2, Generic Array Lists & Linked Lists

PA2 is a **CLOSED** assignment

- Cannot discuss the assignment with anyone other than staff
- Generally, staff will not be able to answer most questions
    - Rephrase your questions! Even though you can’t ask about the PA specifically, you can ask about concepts and material
- If you have a question, please post privately to the instructors on Piazza

However… there are resources!

You are free **(and encouraged!)** to make use of:

- Code from examples, and worksheets from class
- Code posted on Piazza before PA2 release
- Code from discussion (source code will be posted on website)

PA2 Overview

3 parts to this PA

- Part 1: Implementation
  - Implement the given `MyList` interface twice, once for array list and once for linked lists
- Part 2: `MyChooser` and `MyTransformer`
  - Write at least 2 implementations of **each** interface
  - An example of each is given in the respective files.
  - Tip: You will be writing multiple classes in Choosers.java and Transformers.java
- Part 3: Gradescope Questions

PA2 Testing

The thoroughness and correctness of your tests will be graded automatically. To do this, we run your tests on some of our implementations of the PA

- Run your tests against a functional implementation to see if your tests are correct
  - The functional implementation should pass all your tests
- Run your tests against some buggy implementations to see if your tests can catch potential bugs
  - The buggy implementations should fail at least one of your tests

Given Files

- `MyList.java` – you ***cannot*** edit this file
- `MyTransformer.java` – you ***cannot*** edit this file
- `MyChooser.java` – you ***cannot*** edit this file
- `ArrayGL.java` – you will edit this file
- `LinkedGL.java` – you will edit this file
- `TestLists.java` – you will edit this file
- `Choosers.java` – you will edit this file
- `Transformers.java` – you will edit this file

Gradescope Submission

File structure to submit (also in writeup):

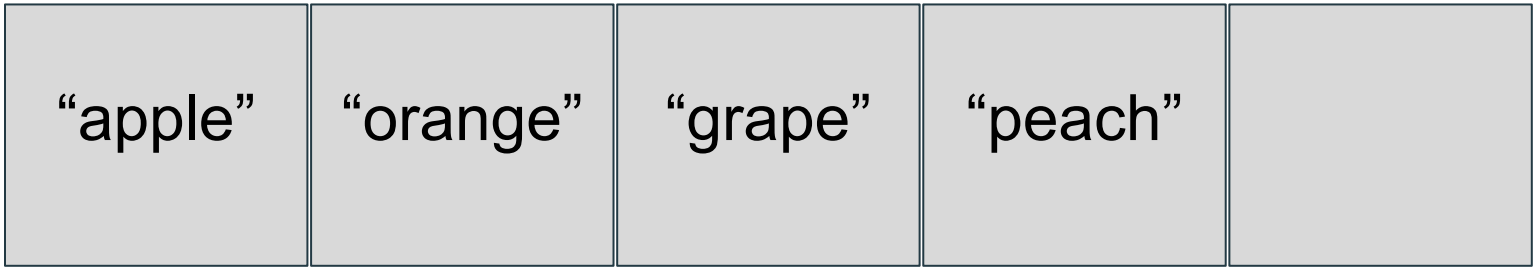
- ArrayGL.java
- Choosers.java
- LinkedGL.java
- MyChooser.java
- MyList.java
- MyTransformer.java
- TestLists.java
- Transformers.java

Note: Some Autograder tests will be hidden this time!!! We will also be testing your tests.

ArrayList

ArrayList overview

- An ArrayList creates a new and larger underlying array once the array is filled up
- Data structure to use when you have unknown number of entries to insert
- Resizes based on a specified size factor
- Can insert, delete, and access elements in ArrayList



Capacity = 5 **Size = 4**

ArrayList implementation - interfaces

```
public interface GList<E>{
  /* Add an element at the end of the list */
  void add(E e);

  /* Get the element at the given index */
  E get(int index);

  /* Get the number of elements in the list
*/
  int size();

  /* Add an element at the specified index */
  void insert(int index, E e);

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

  /* Increase the capacity of underlying
array */
  void expandCapacity();

  /* Get the amount of elements array can
hold */
  int getCapacity();
}
```

```
public class MyArrayList<E>implements
GList<E> {
     E[ ] elements;
     int size;

     void add(E e) { … }
     E get(int index) { … }
     int size() { … }
     void insert(int index, E e) { … }
     void remove(int index) { … }
     void expandCapacity() { … }
     int getCapacity() { … }
}
```

Methods in ArrayList

add()	Adds an element to the end of the list
prepend()	Adds an element to the front of the list
expandCapacity()	Doubles the capacity of the underlying array

What should the initial value of size be?

A) 2

B) 0

C) N/A

```
public class MyArrayList<E> implements GList<E> {
    E[] elements;
    int size;

    public MyArrayList() {
        this.elements = (E[]) new Object[2];
        this.size = /* initial value */;

    }
}
```

What should the initial value of size be?

A) 2

B) 0

C) N/A

```
public class MyArrayList<E> implements GList<E> {
    E[] elements;
    int size;

    public MyArrayList() {
        this.elements = (E[]) new Object[2];
        this.size = /* initial value */;

    }
}
```

What is the initial `capacity` of elements?

A) 2

B) 0

C) N/A

```
public class MyArrayList<E> implements GList<E> {
    E[] elements;
    int size;

    public MyArrayList() {
        this.elements = (E[]) new Object[2];
        this.size = 0;

    }
}
```

What is the initial `capacity` of elements?

A) 2

B) 0

C) N/A

```
public class MyArrayList<E> implements GList<E> {
    E[] elements;
    int size;

    public MyArrayList() {
        this.elements = (E[]) new Object[2];
        this.size = 0;

    }
}
```

ArrayList implementation - expandCapacity()

```
public class MyArrayList<E> implements
GList<E> {
      E[ ] elements;
      int size;

      MyArrayList() { … }
      void add(E e) { … }
      E get(int index) { … }
      int size() { … }
      void insert(int index, E e) { … }
      void remove(int index) { … }
      void expandCapacity() { … }
      Int getCapacity() { … }
}
```

How would we implement expandCapacity()?

A)
```
E[ ] temp = (E []) new Object[this.elements.length + 2];
for (int i = 0; i < this.elements.length; i++) {
    temp[i] = this.elements[i];
}
this.elements = temp;
```

B)
```
this.elements.length = this.elements.length + 1;
```

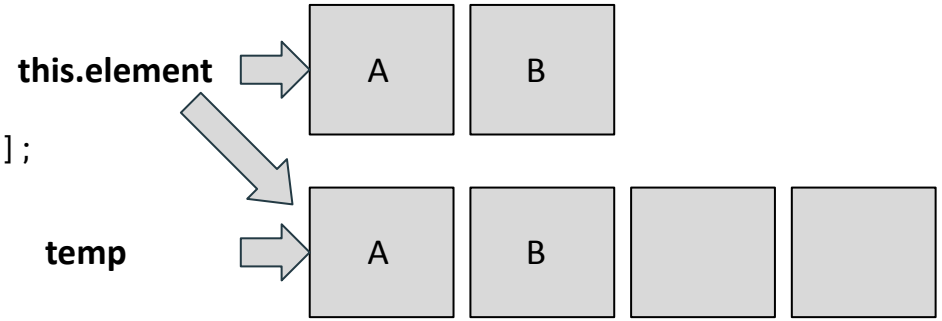
C)
```
E[ ] temp = (E []) new Object[this.elements.length * 2];
for (int i = 0; i < this.elements.length; i++) {
    temp[i] = this.elements[i];
}
this.elements = temp;
```

D) None of the above

ArrayList implementation -expandCapacity()

```
public class MyArrayList<E> implements GList<E> {
      E[ ] elements;
      int size;

      MyArrayList() {
            this.elements = (E[]) new Object[2];
            this.size = 0;
      }
      void add(E e) { … }
      E get(int index) { … }
      int size() { … }
      void insert(int index, E e) { … }
      void remove(int index) { … }
      void expandCapacity() {
            E[ ] temp = (E[])new Object[this.elements.length * 2];
            for (int i = 0; i < this.elements.length; i++) {
                  temp[i] = this.elements[i];
            }
            this.elements = temp;
      }
      int getCapacity() { … }
}
```



Answer - C

DRAW MEMORY DIAGRAMS TO VERIFY SOLUTION

ArrayList implementation - add(E e)

```
public class MyArrayList<E> implements GList<E>
{
      E[ ] elements;
      int size;

      MyArrayList() {
            this.elements =(E[]) new Object[2];
            this.size = 0;
      }
      void add(E e) { … }
      E get(int index) { … }
      int size() { … }
      void insert(int index, E e) { … }
      void remove(int index) { … }
      void expandCapacity() { … }
      int getCapacity() { … }
}
```

How would we implement add(E e)?

A) `this.elements[0] = e;`

B) `this.elements[this.size] = e;`

C) `this.elements[this.elements.length] = e;`

D) None of the above

ArrayList implementation - add(E e)

```
public class MyArrayList<E> implements GList<E> {
      E[ ] elements;
      int size;

      MyArrayList() {
            this.elements = (E[]) new Object[2];
            this.size = 0;
      }
      void add(E e) {
            if (this.size > this.elements.length - 1) {
                  expandCapacity();
            }
            this.elements[this.size] = e;
            size+=1;
      }
      Eget(int index) { … }
      int size() { … }
      void insert(int index, E e) { … }
      void remove(int index) { … }
      void expandCapacity() {...}
      int getCapacity() { … }
}
```

Answer - D

ArrayList implementation - get(int index)

```
public class ArrayStringList implements
StringList {
      String [ ] elements;
      int size;

      ArrayStringList() {
            this.elements = new String[2];
            this.size = 0;
      }
      void add(String s) { … }
      String get(int index) { … }
      int size() { … }
      void insert(int index, String s) { … }
      void remove(int index) { … }
      void expandCapacity() { … }
      int getCapacity() { … }
}
```

How would we implement get(int index)?

A)
```
return this.elements[index];
```

B)
```
if (index < this.elements.length
      && index >= 0) {
      return this.elements[index];
}
return null;
```

C)
```
if (index < this.elements.length - 1)
{
      return this.elements[index];
}
```

D) None of the above

ArrayList implementation - get(int index)

```
public class MyArrayList<E> implements GList<E> {
     E[ ] elements;
     int size;

     MyArrayList() {
          this.elements = (E[]) new Object[2];
          this.size = 0;
     }
     void add(E e) {...}
     E get(int index) {
          if (index < this.elements.length
               && index >= 0) {
               return this.elements[index];
          }
          return null;
     }
     int size() { … }
     void insert(int index, E e) { … }
     void remove(int index) { … }
     void expandCapacity() {...}
     int getCapacity() { … }
}
```

Answer - B

Checking your understanding

You have the following Strings: {“blue”, “red”, “purple”, green”, “yellow”}

You add each of these Strings individually to an ArrayList. If this ArrayList starts at capacity 2 and its capacity increases by a factor of 2 during each resize, what will be the capacity after inserting these 5 Strings?

A) 5
B) 4
C) 2
D) 8
E) 16

Answer - D

You have the following Strings: {“blue”, “red”, “purple”, green”, “yellow”}

ARRAYLIST ELEMENTS	ARRAYLIST CAPACITY
“blue”	2
“blue”, “red”	2
“blue”, “red”, “purple”	4
“blue”, “red”, “purple”, “green”	4
“blue”, “red”, “purple”, “green”, “yellow”	8

ArrayList implementation - insert(int index, E e)

- Add E e at the specified index and shift all subsequent elements to the right by one index
- Practice implementing this method

ArrayList implementation - insert(int index, E e)

```
// assumes index is valid
public void insert(int index, E e){
    if(this.size >= this.elements.length) {
      expandCapacity();
    }
    for(int i = this.size; i > index; i--){
      this.elements[i] = this.elements[i - 1];
    }
    this.elements[index] = e; // why must this come after loop?
    this.size += 1;

    return;
}
```

ArrayList implementation - remove(int index)

- Remove E e at the specified index and shift all subsequent elements to the left by one index
- Practice implementing this method

ArrayList implementation - remove(int index)

```
// assumes index is valid
public void remove(int index){
    for(int i = index; i < this.size - 1; i++){
      this.elements[i] = this.elements[i + 1];
    }

    this.elements[size] = null;
    this.size -= 1;

    return;
  }
```

LinkedList

Linked List implementation

```
public interface GList<E>{
  /* Add an element at the end of the list */
  void add(E e);

  /* Get the element at the given index */
  E get(int index);

  /* Get the number of elements in the list */
  int size();

  /* Add an element at the specified index */
  void insert(int index, E e);

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

  /* Increase the capacity of underlying array */
  void expandCapacity();

  /* Get the amount of elements array can hold */
  int getCapacity();
}
```

```
public class MyLinkedList<E> implements GList<E>{

  class Node {
    E value;
    Node next;
    public Node(E value, Node next) {
      this.value = value;
      this.next = next;
    }
  }

  Node front;
  int size;

  public MyLinkedList() {
    this.front = new Node(null, null);
  }
  public void prepend(E e) {...}
  public E get(int index) {...}
  public void add(E e) {...}
  public int size() {...}
  public void remove(int index){  ...}
  public void insert(int index, E e){...}
```

LinkedList implementation - insert(int index, E e)

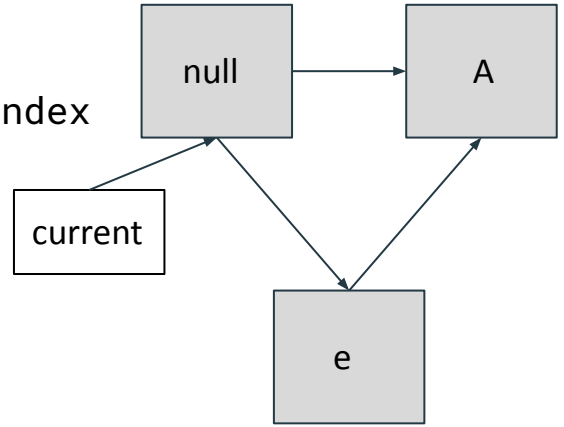
- Add an element at the specified index

```
public void insert(int index, E e){
    Node current = this.front;

    for(int i = 0; i < index; i += 1) {
      current = current.next;
    }

    current.next = new Node(e, current.next);
    this.size += 1;

    return;
  }
```



LinkedList implementation - remove(int index)

- Remove the element at the specified index
- Practice implementing this method

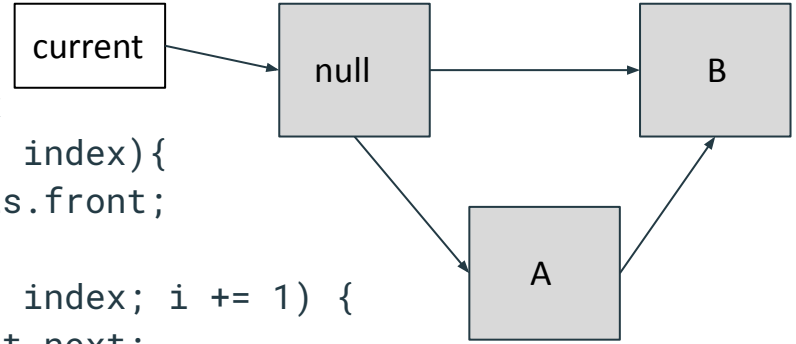
LinkedList implementation - remove(int index)

```
// assumes valid index
public void remove(int index){
    Node current = this.front;

    for(int i = 0; i < index; i += 1) {
      current = current.next;
    }

    current.next = current.next.next;
    this.size -= 1;

    return;
  }
```



Source Code

Posted in the course schedule on the course website is a link to source code with additional implementations for `MyLinkedList` and for `MyArrayList`

Note: These implementations don't necessarily check for out of bounds cases. We leave that to you! Because of this, the current implementations don't pass all of the tests we wrote.

Source code will be posted on the course Github

Link to Google Slides

https://docs.google.com/presentation/d/1GLTzPvfDUpPUgO_lwnuaPDlHLzHrB_0GPwAECFkNSs0/edit?usp=sharing