

Drawing before and after of `addLast` to `ArrayList` with wraparound

		"A"	"B"
--	--	-----	-----

(start = 2, end = 3)

`addLast("C"):`

		"A"	"B"
--	--	-----	-----

(start = ____, end = ____)

Test class

```
package ArrayLists;

import org.junit.Assert;
import org.junit.Test;

public class TestArrList {
    ArrList theList;

    @Test
    public void testAddLast() {
        this.theList = new ArrList(4);
        this.theList.addLast("A");
        Assert.assertEquals("A", this.theList.get(0));
    }

    @Test
    public void testAddLastWraparound() {
        this.theList = new ArrList(4);
        this.theList.addLast("A");
        this.theList.addLast("B");
        this.theList.addLast("C");
        Assert.assertEquals("C", this.theList.get(2));
    }
}
```

New/updated methods in ArrList class

```
public class ArrList {
    String[] theArray;
    int eltcount;
    int start;
    int end;

    public ArrList(int initSize) {
        this.theArray = new String[initSize];
        this.start = initSize / 2;
        this.end = this.start;
        this.eltcount = 0;
    }

    public boolean isEmpty() { return this.eltcount == 0;}
    public boolean isFull() { return this.eltcount == this.theArray.length;}

    public String get (int index) {
        if ((index >= 0) && (index < this.eltcount)) {
            return this.theArray[(start + index) % this.theArray.length];
        }
        throw new IllegalArgumentException("array index " + index + " out of
bounds");
    }

    private void resize(int newSize) {
        String[] newArray = new String[newSize];
        for (int index = 0; index < this.theArray.length; index++) {
            newArray[index] = this.theArray[index];
        }
        this.theArray = newArray;
    }

    public void addLast(String value) {
        if (this.isFull()) {
            this.resize(this.theArray.length * 2);
        }
        if (this.eltcount != 0) {
            this.end = this.end + 1;
            if (this.end == this.theArray.length) {
                this.end = (this.end % this.theArray.length) + 1;
            }
        }
        this.theArray[this.end] = value;
        this.eltcount = this.eltcount + 1;
    }
}
```

Banking classes and methods

Account class:

```
Account(int num, Customer c, double bal)
```

Customer class:

```
public Customer(String nm, String pwd)
public void addAccount(Account newAcct)
public void closeAccount(int acctNum)
public boolean hasAccount(int acctNum)
```

BankingService class:

```
public void addAccount(Account newA)
public double withdraw(int forAcctNum, double amt)
public double addInterest(double percentage)
```

Predicted outputs:

```
BankingService B = new BankingService();
Customer kCust = new Customer("kathi", "cs200");
Account kAcct = new Account(100465, kCust, 150);
kCust.addAccount(kAcct);
B.addAccount(kAcct);
System.out.println("Kathi's balance is " + kAcct.balance);
```

What is the output here?

```
B.withdraw(100465, 30);
System.out.println("Kathi's balance is " + kAcct.balance);
```

What is the output here?

```
Customer mCust = new Customer("milda", "csci0200");
Account mAcct = new Account(100023, mCust, 111);
mCust.addAccount(mAcct);
B.addAccount(mAcct);

Account kChecking = new Account(100041, kCust, 330);
kCust.addAccount(kChecking);
B.addAccount(kChecking);

B.withdraw(100023, 11);
kCust.closeAccount(100065);
B.withdraw(100041, 30);

double interestThisMonth = B.addInterest(0.01);

System.out.println("Total interest added this month: " + interestThisMonth);
```

What is the output here?

Class hierarchy/containment

BankingService

```
classDiagram
    class BankingService
    class Customer
    class Account
    BankingService "1" -- "*" Customer
    BankingService "1" -- "*" Account
```

The diagram illustrates a class hierarchy/containment relationship. At the top is the **BankingService** class. Below it are two other classes: **Customer** and **Account**. The **BankingService** class is positioned above the other two, suggesting it is a base or container class. The **Customer** and **Account** classes are positioned below **BankingService**, indicating they are associated with or contained by **BankingService**.

Customer

Account

Before removeAccount

After removeAccount