# CS 315 Sample Exam 1 Questions

**Note:** You will NOT have this many questions on the exam. I just want to cover all of the bases here so you don't get surprised.

**Part A**

STUDY the accompanying Account, Savings, and Checking classes and then answer the following questions.

1. Which classes have a default constructor?
2. Suppose that I am create a new subclass of Checking called ExecutiveChecking. Also suppose that this new class has a boolean member variable named vip. What would be wrong with the following constructor?

```
public ExecutiveChecking()
{
    vip = false;
}
```

3. How many member (instance) variables will an object of the Savings class have?
4. How many member (instance) variables will an object of the Checking class have?
5. If the following statements were to be executed in a program, what would be displayed on the screen? Try to figure this out without running code.

```
Account acct = new Checking(1234, 500);
System.out.println(acct);
double amt = acct.withdraw(450);
System.out.println(acct);
```

6. If the following statements were to be executed in a program, what would be displayed on the screen? Try to figure this out without running code.

```
Account acct = new Savings(3456);

acct.deposit(100);

System.out.println(acct);

acct.deposit(200);

System.out.println(acct);
```

7. One of the following statements is syntactically incorrect. State which one it is and explain why it is incorrect. Then give TWO DIFFERENT ways to prevent the error from happening WITHOUT having to fiddle with either of the Account, Checking, or Savings classes.

```
Account acct = new Savings(5432);

acct.setRate(0.2);
```

8. Suppose that you are creating a Bank class that has the following member variables:

```
private ArrayList<Checking> checkingAccounts;

private Savings[] savingsAccounts;

private int numSavings;  //Keeps track of the number
of Savings objects actually stored in the array.
```

   a. Write an addChecking method for the Bank class. Its job is to add a valid Checking object to checkingAccounts. The method either returns what was added or null if nothing could be added. The method header is:

```
public Checking addChecking(Checking acct)
```

b. Write an addSavings method for the Bank class. Its job is to add a valid Savings object to savingsAccounts. The method either returns what was added or null if nothing could be added. The method header is:

    public Savings addSavings(Savings acct)

c. Write a method called calcTotalCheckingBalances that returns the result of adding up the balances of all of the Checking objects stored in checkingAccounts.

d. Write a method called calcTotalSavingsBalances that returns the result of adding up the balances of all of the Savings objects stored in savingsAccounts.

e. Write a method called findSavingsAccount that takes an integer account number as a parameter and will try to find the savings account with that account number. If it finds the account, it returns a reference to the account. Otherwise, it returns null. The method header is: public Savings findSavingsAccount(int number)

**Part B**

The following questions refer to the OurList<T> class, **part** of which is reproduced here.

```
public class OurList<T>
{
    private LinkNode<T> head;  //keeps track of the start of the
list
    private int numNodes;  //keeps track of the number of nodes
on the list.

    public OurList()
    {
        head = null;
        numNodes = 0;
    }
```

```
//Some method prototypes are:
     public int size(); //returns number of nodes on list

     public boolean isEmpty(); //Returns true if list has
     nothing. Otherwise, false.

     public T add(T data); //Attempts to add the data to the
     list. If it can't, it returns null. Otherwise, it returns
     the same data it added.

     public T get(int index); //gets data of node at specified
     index

     public T remove(int index); //removes node at specified
     index and returns data

     public void clear(); //clears the list by resetting all
     member variables.

    /********Inner Class**************/
    private class LinkNode<T> {
        private T data;
        private LinkNode<T> next;

        private LinkNode(T data)
        {
            this.data = data;
            next = null;
        }
    }
}
```

1. Write a statement that will create a variable that refers to an OurList<T> object for storing double types.
2. Write statements that will add any three numbers to the list created in question 1.
3. Pretending that you don't know what is on the list at this time or how many there are, write **general** statements that will **multiply** together all of the numbers on the list.
4. Write **general** statements that will result in **removing** the **last** number on the list and printing it, whatever that number is. You are not to use ANY prior knowledge of the

contents of the list. You can't even assume that there is anything on the list.

5. Write statements that will **clear** the list WITHOUT calling the **clear** method. That is, you are not just allowed to say list.clear(). Be careful here! Every time you remove an element from the list the number of elements goes down by one.
6. Write a method called moveFrontToBack that will be added to the set of OurList<T> methods. Its job is to take the node at the head and move it to the back if there is something on the list. The method header is:

   ```
   public void moveFrontToBack()
   ```

7. Write a method called moveBackToFront that will be added to the set of OurList<T> methods. Its job is to take the node at the back and move it to the front if there is something on the list. The method header is:

   ```
   public void moveBackToFront()
   ```

8. Write a method called swapFrontWithBack that will be added to the set of OurList<T> methods. Its job is to take the node at the front and switch it with the node at the back if there is something on the list. The method header is:

   ```
   public void swapFrontWithBack()
   ```