Unit Seven Lecture Two:

Topic 1: Array Lists

While arrays hold primitive data types like int, double, char, or String, array lists hold a sequence of objects.

To use an array list you must first

```
import java.util.ArrayList;
```

To declare an array list

```
ArrayList<BankAccount> accounts = new ArrayList<BankAccount>();
```

To fill one element of an array list with an object of type BankAccount

```
accounts.add(new BankAccount(acctNum, balance));
```

Once the array list is filled, you can access an individual object

```
BankAccount anAcct = accounts.get(i);
```

i is an index or subscript. The first object added to the array list would have an index of 0, the second object an index of 1, and so forth.

You can determine how many objects are in the array list

```
int x = accounts.size();
```

If x has a value of 20, this means there are 20 objects in the array list with indices ranging from 0 to 19.

To overwrite an object that is already in the array list

```
BankAccount myAcct = new BankAccount("132s", 500);
accounts.set(12, myAcct);
```

The object that had an index of 12 is now gone and *myAcct* takes its place.

To insert a new object into the array list

```
BankAccount myAcct = new BankAccount("132s", 500);
accounts.add(12, myAcct);
```

The size of the array list is increased by one and all objects from index 12 to the end are moved forward one position. *myAcct* is inserted at index 12.

To delete an object from the array list

```
accounts.remove(15);
```

The object at index 15 is removed, all objects from index 16 to the end are moved back one position, and the size of the array list is decreased by one.

Topic 2: Reading multiple lines from a text file

To use this code you must first

```
import java.util.Scanner;
```

To read multiple lines from a text file

```
try
{
      Scanner reader = new Scanner(new File("h:\\myFile.txt"));

      while (reader.hasNext())
      {
            String inputLine = reader.nextLine();
            .
            .
            .
      }
      reader.close();
}
catch (IOException e)
{
      throw new RuntimeException(e.toString());
}
```

Topic 3: Using the java.io package to print data to a text file

To use this package you must first

```
import java.io.*;
```

To print data to a text file

```
try
{
      FileWriter writer = new FileWriter("f:\\javaData\\example.txt");
      PrintWriter out = new PrintWriter(writer);

      String myString = "Today is Monday";

      out.println(myString);

      out.close();
}
catch (IOException e)
{
      throw new RuntimeException(e.toString());
}
```

There are two good reasons for closing a file when you are finished with it.

1 - It is always a good idea to release all resources back to the system as soon as we no longer need them.

2 - If you fail to close a file opened for write only, some of the text you think you have written to the file may not wind up in the file. This is because a PrintWriter object collects many characters and actually transfers a whole bunch at a time.

Topic 4: Switching from a JApplet to a JFrame

As you will soon find out, u8a2 requires you to read data from a text file, manipulate it, and send the new data to a new text file. If you attempt to complete this project with an applet, you will find out that Java will not allow you to send data from an applet to a text file.

This is for security reasons because an applet can come to your computer from an unknown source on the World Wide Web, and you do not want to give it access to your files.

What if you didn't discover this until your applet project was finished? How can you change it to an application that extends JFrame?

1 - instance variables will probably stay the same

2 - public void init() will become the default constructor of the class i.e. public u8a2()
3 - you will have to create public static void main(String args[]) that will call the default constructor i.e. u8a2 x = new u8a2();
4 - In either the default constructor or main you will have to set the size of the JFrame, make it visible, and close the application when the JFrame is closed.

Topic 5: Using ArrayLists for primitive data types!

Because primitive data types are not objects in Java, you cannot directly insert them into an arrayList. You must first turn them into objects by using wrapper classes (i.e. Integer).

Starting with Java version 1.5, conversion between primitive types and the corresponding wrapper classes is automatic. This process is called auto-boxing (**Integer x = 5;**) or auto-unboxing (**int y = x;**).

Example:

```
import java.util.ArrayList;

public class Wrapper
{
      public static void main(String args[])
      {
            ArrayList<Integer> list1 = new ArrayList<Integer>();
            ArrayList<Integer> list2 = new ArrayList<Integer>();
            ArrayList<Integer> list3 = new ArrayList<Integer>();

            for (int j=0; j<5; j++)
            {
                  list1.add(j*j);         // auto-boxing
                  list2.add(j);                 // auto-boxing
            }
            for (int k=0; k<list1.size(); k++)
            {
                  int x = list1.get(k);   // auto-unboxing
                  int y = list2.get(k);   // auto-unboxing

                  list3.add(x - y);       // auto-boxing
            }
            for (int m=0; m<list3.size(); m++)
            {
                  System.out.print(list1.get(m) + "\t");
                  System.out.print(list2.get(m) + "\t");
                  System.out.println(list3.get(m));
            }
      }
}
```

Unit 7 Assignment 2: Manipulating an Array List of BankAccounts