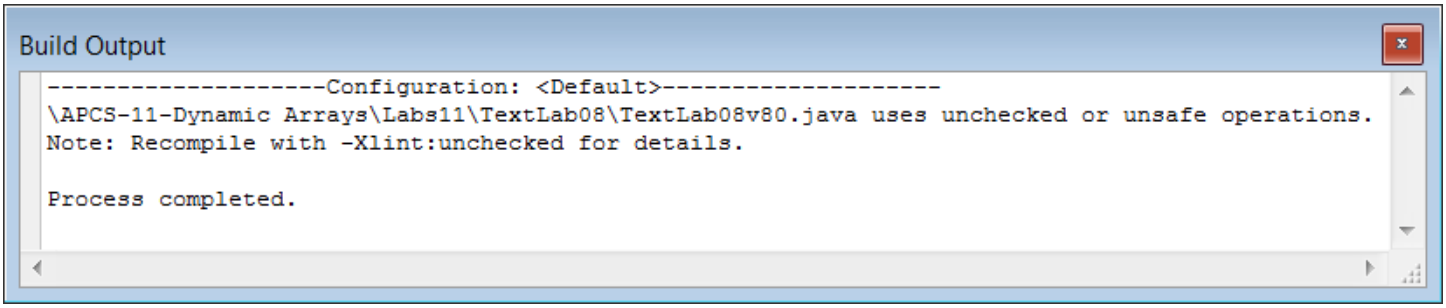
AP Computer Science A	TextLab08 Java Assignment
The "Dynamic Matrix" Program	**80 & 100 Point Versions**

Assignment Purpose:

The purpose of this assignment is to demonstrate knowledge of the **ArrayList** class by creating a dynamic two-dimensional **Matrix** class implemented with the **ArrayList** class.

Write a **Matrix** class that handles two-dimensional array-needs "dynamically". Dynamically means that the size of the array can be altered during program execution. You are required to use the Java one-dimensional **ArrayList** class to implement the **Matrix** class. You will be provided with a student file, which includes the **main** methods for the 80-point and 100-point version.

Chapter 11 introduces the "generics" concept with the **ArrayList** class. You have not yet learned how to create your own generic class, which is shown in a later chapter. For that reason this assignment and the **Matrix** class will be implemented without the use of generics. Now Java is very touchy when *new and improved* features are not used. You will get the warning message shown below. Ignore the message. Later in the course the warning will disappear when you use generics.



TextLab08 80 and 100 Point Versions	Student Provided File

```
// TextLab08st.java
// This is the student starting version of the TextLab08 assignment.
// Testing <main> methods are provided for the 80-point and 100-point versions.
// This means that this version will not compile as provided.

import java.util.ArrayList;

public class TextLab08st
{
    public static void main(String args[])
    {
        System.out.println("\nTextLab08 STUDENT VERSION\n");

        Matrix m1 = new Matrix();
        m1.displayMatrix("Matrix m1 Default Display");
        System.out.println();

        Matrix m2 = new Matrix(3,5);
        m2.displayMatrix("Matrix m2 3 X 5 Display");
        System.out.println();
        int count = 100;
        for (int r = 0; r < m2.getRows(); r++)
        {
            for (int c = 0; c < m2.getCols(); c++)
            {
                m2.setValue(r,c,count);
                count++;
            }
        }
        m2.displayMatrix("Matrix m2 3 X 5 Consecutive Integers Display");
        System.out.println();

        Matrix m3 = new Matrix(3,3,100);
        m3.displayMatrix("Matrix m3 3 X 3 Initialized to 100 Display");
        System.out.println();
    }

}



class Matrix
{

    private ArrayList list;     // one-dimensional array stores matrix values
    private int listSize;       // total number of elements in the matrix
    private int numRows;        // number of rows in the matrix
    private int numCols;        // number of cols in the matrix


}
```

80 & 100 Point Versions

The 80-point version of the **Matrix** class requires implementing three constructors plus methods **getRows**, **getCols**, **getSize**, **getValue**, **setValue** and **displayMatrix**. There are four private date

fields, which are **list**, **listSize**, **numRows** and **numCols**. Dynamic resizing is not required for the 80-point version, but method **resize** is required and this is the only difference with the 100-point version.

TextLab08 80 & 100 Point Version	**Matrix Class Interface**

```
private ArrayList list;     // one-dimensional array stores matrix values
private int listSize;       // total number of elements in the matrix
private int numRows;        // number of rows in the matrix
private int numCols;        // number of cols in the matrix


/**  Constructs empty ArrayList object and sets all values to 0  **/
public Matrix()

/**  Constructs r X c matrix with all elements initialized to 0  **/
public Matrix(int r, int c)

/** Constructs r X c matrix will all elements initialized to value  **/
public Matrix(int r, int c, int value)

/**  Returns numRows value  **/
public int getRows()

/**  Returns numCols value  **/
public int getCols()

/**  Returns listSize value  **/
public int getSize()

/**  Returns Matrix object value at (r,c) location  **/
public int getValue(int r, int c)

/**  Alters Matrix object value at (r,c) to value  **/
public void setValue(int r, int c, int value)

/**  Displays Matrix object in two-dimensional matrix format  **/
public void displayMatrix(String str)

/**  Resizes Matrix object to new rows X cols size, copies all possible
     previous values and initializes new elements to zero  **/
public void resize(int rows, int cols)
```

TextLab08 80 Point Version | One Required Output

```
TextLab08 80-POINT VERSION

Matrix m1 Default Display
Matrix has no elements

Matrix m2 3 X 5 Display
0  0  0  0  0
0  0  0  0  0
0  0  0  0  0

Matrix m2 3 X 5 Consecutive Integers Display
100  101  102  103  104
105  106  107  108  109
110  111  112  113  114

Matrix m3 3 X 3 Initialized to 100 Display
100  100  100
100  100  100
100  100  100
```

TextLab08 100 Point Version | One Required Output

```
TextLab08 100-POINT VERSION

Matrix m1 Default Display
Matrix has no elements

Matrix m2 3 X 5 Consecutive Integers Display
100  101  102  103  104
105  106  107  108  109
110  111  112  113  114

Matrix m2 After 4 X 4 Resizing Display
100  101  102  103
105  106  107  108
110  111  112  113
0  0  0  0

Matrix m3 3 X 3 Initialized to 100 Display
100  100  100
100  100  100
100  100  100
```