# Array Basics

An array is used to store a collection of data, but is more easily remembered as a collection of variables of the same type.

You will want to import java.util.\*; so you have access to array list characteristics.

# Declaring Array Variables

datatype[ ] arrayRefVar;

Example; double[ ] myList;

# Creating Arrays

Unlike primitive data type variables, the declaration of an array variable does not allocate any space in memory for the array. It creates only a storage location for the reference to the array. If a variable does not contain a reference, the value of the variable is **null**.

arrayRefVar = new dataType [arraySize];

Example: **double**[ ] myList = **new double**[10]

# Array Size and Default Values

Array size when it is allocated is given by the number of elements and cannot be changed after created.

arrayRefVar[length]

Example: myList[10]

Common usage: **for** (**in**t I = 0; I < myList.length; i++) {

myList[i] = I;

# Array Initializers

Java shorthand – known as the array initializer, combines in one statement declaring the array while creating an array and initializing

datatype[ ] arrayRefVar = {value0, value1, …, value k}

For Exmaple**; double**[ ] myList = {1.9, 2.9, 3.4, 3.4}

The statement both declares, creates and initializes the array myList with 4 elements:

**double [ ] myList =** new **double[4];**

myList[0] = 1.5;

myList[1] = 2.5;

myList[2} = 3.5;

myList[3] = 4.5;

# For-Each Loops

Note: JDK 1.5 introduced a new **for** loop, known as a *for-each* or *enhanced for loop*, which enables you to transverse the complete array sequentially. Look at the examples…

Example TestArray.java

Other example AssignGrade.java

# Adding items to a list

myList.add(value99);

you could also add to an array list as the result of a button-click or other event, calling the **add** to place an item at the end of the array list.

**private void** addAnItem(myList list) {

list.add(textField.getText());

}

# Length of a list

You can find out how long an array list is by using the library method **size**.

**int** numberOfItems = myList.size();

This can also be used within a method using an option pane

**private void** displayLength(myList list) {

JOptionPane.showMessageDialog(null, Integer.toString(myList.size( ));

}

# Using index values

You can display not just the values, but the values alongside their index values:

**private void** displayWithIndices(myList list) {

String newLine = “\n”;

String tab = “\t”;

textArea.setTabSize(3);

textArea.setText(“”);

for (int index = 0; index < myList.size(); index ++) {

textArea.append(Integer.toString(index)

+ tab

+ list.get(index) + newline);

}

}

# Removing from an Array List

You may also remove items, just like you can add. The **remove** method of the **ArrayList** removes the item at a particular index value.

myList.remove(3);

We`ve removed whatever was at position 3 and now position 4 takes it`s spot. The gap is closed. The array shrinks. You can remove everything by using the **clear** method.

myList.clear();

# Time to practice:

1. Write a program in which an item in an array list is deleted.

1. Write a program that allows items to be inserted into or removed from any position within an array list.
2. Do some research, find out how to search an array.