Practical Exercise 8 – Arrays

An array is an indexed (numbered) list of items that are all of the same data type.

The purpose of this practical exercise is to provide an introduction to arrays, how to create them, and how to use them.

Here is some example code that demonstrates some of these ideas…

**import** java.awt.\*;

**import** javax.swing.\*;

**class** MyCanvas **extends** JPanel

{

String[] shoppingList;

**int**[] sixTimes;

**int** yPos;

**public** **void** init()

{

shoppingList = **new** String[6];

shoppingList[0]="Bread";

shoppingList[1]="Milk";

shoppingList[2]="Butter";

shoppingList[3]="Eggs";

shoppingList[4]="Cheese";

shoppingList[5]="Bandaids";

sixTimes = **new** **int**[10];

**for**(**int** i=0; i<sixTimes.length; i++)

{

sixTimes[i] = 6\*(i+1);

}

}

**public** **void** paint(Graphics g)

{

yPos=20;

**for**(**int** i=0; i<shoppingList.length; i++)

{

g.drawString(i+" "+shoppingList[i], 20, yPos);

yPos=yPos+20;

}

yPos=20;

**for**(**int** i=0; i<sixTimes.length; i++)

{

g.drawString((i+1)+" x 6 = "+sixTimes[i], 100, yPos);

yPos=yPos+20;

}

}

}

**public** **class** ArrayDemo

{

**public** **static** **void** main(String[] a)

{

MyCanvas myCanvas = **new** MyCanvas();

myCanvas.init();

JFrame window = **new** JFrame();

window.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

window.setBounds(30, 30, 400, 300);

window.getContentPane().add(myCanvas);

window.setVisible(**true**);

}

}

# Task 1 – Fruit by the Multiples

Write a program that sets up an array for each of the following lists and displays them as indicated *in the same window*.

1. Create an array that contains seven different types of fruit and display them *across the screen*.
2. Create an array that contains multiples of 12 (in the range 12 to 288) and display them *down the screen*). Use one loop to create the array and a separate one to display its contents.

# Task 2 - Running Total

The following code fragment is intended to add the numbers stored in the array, and display the running total as it goes. Your challenge is to get this working.

int[] a = {6, 7, 9, 12, 4};

int total, ypos;

public void paint(Graphics g)

{

total = 0;

ypos = 20;

for (int i = 0; i < a.length; i++)

total = total + a[i];

g.drawString("" + a[i], 20, ypos);

g.drawString("" + total, 120, ypos);

ypos + ypos + 20;

}

For example, for the list {6, 7, 9, 12, 4}, the output should be…

6 6

7 13

9 22

12 34

4 38

Total: 38

# Task 3 – Search array

SEARCHING AN ARRAY – We often need to search through an array to see whether or not it contains a particular value.

1. Amend the following code to display the contents of the array and output a message indicating if the search item was found.

boolean found;

int searchItem = 9;

int[] a = {6, 7, 9, 12, 4};

public void paint(Graphics g)

{

found = false;

i = 0;

while ((i<a.length) && ! found)

{

if (a[i] == searchItem)

found = true;

else

i++;

}

}

1. If the search item is found, also display the location (index) where it is found.
2. Make searchItem a random number between 1 and 10 and test if this number is in the array.
3. Display the random number as well as the result of the search (including the location if found).

MAX/MIN VALUES - It is often useful to find the largest (or smallest) item in an array. To do this, one approach is to create a variable initially set to the value of the first item in the array, then search through the list to find anything larger. For example…

int max = a[0];

for (int i=0; i<a.length; i++)

{

if (a[i] > max)

max = a[i];

}

1. Amend your code so that the smallest *and* largest numbers in the array are found and displayed.

# Task 4 – Bubble sort integers (stretch goal 1 - for bonus marks)

BUBBLE SORT – There are many different techniques for sorting the contents of an array, including *bubble sort*.

To perform a bubble sort, we progressively compare consecutive elements in the array and, if necessary, swap the values. To swap two values in an array we need another variable to temporarily hold one of the values.

For example, in Java this could look like this…

if (a[i] > a[i+1])

{

hold = a[i];

a[i] = a[i+1];

a[i+1] = hold;

}

1. Use the following global variables to sort an array into ascending order using a bubble sort and display the sorted array. Hint: you will need a *for loop* inside a while loop to do this and some additional research may be required.

public class bubbleSort extends Applet

{

int[] a = {8, 5, 64, 15, 2, 1, -3, 0, 42, 99};

boolean finished;

int i, j, xpos, hold;

// and so on...

1. Modify your code to sort the array in descending order, and display the sorted array.
2. Add two buttons so that the user can choose to sort the array in either ascending or descending order, and output the result.

# Task 5 – Bubble sort strings (stretch goal 2 - for more bonus marks)

Create a program to bubble sort an array of strings rather than integers.  
  
Use this array…

String[] a = {"Greg, "Geoff", "Anthony", "Murray", "Dorothy", "Wags", "Henry"};

In the bubble sort algorithm, when comparing two items that are strings, you cannot use  
 if (a[ i ] > a[i+1]), instead you can use a[i].compareTo(a[i+1])

a[i].compareTo(a[i+1]) will result in a number < 0 if a[i] is < a[i+1]

For example if item a[i] contained "Ant" and item a[i+1] contained "Dog", a[i].compareTo(a[i+1]) will return a negative number.

The reason for this is that the comparison is made using the ASCII numeric code for the characters e.g. 'A' is 65 and 'C' is 67 and on comparison, 65 – 67 gives an answer of –2 which is less than 0.

Display both the original, unsorted array as well as the sorted array.