**import** java.util.Scanner; // import this package to use it for input from user in console

**public** **class** rainfallTester {

/\*\* Patricia Organ - 01110489 - Assignment 5 Q1

\*Write an application that uses an array data structure to store 12 numerical values

\*entered by the uses. Each of these values represents the amount of rainfall for a

\*particular month of the year. The application should declare and create the array

\*and then pass it to a method called populateArray, which takes input from the user

\*and inserts it into the array. The application should then pass the array to a method

\*called calculateAverage, which will calculate and return the average rainfall for the

\*12 months entered.

\*/

**public** **static** **void** main(String[] args) {

// Declare array for inputed values of double type

//Initializing array as the question indicated it would be only for 12 months

//but the one place of setting size can easily be changed in future requirements

//if required and methods will still work

**double**[] myArray = **new** **double**[12];

//declare a double variable to receive the result from the CalculateAverage

//return value

**double** average;

//call the populate method to add the values to the array

*PopulateArray*(myArray);

// now the array is full we need to call the calculate average method to output

//the average

average = *CalculateAverage*(myArray);

//output to console the average, using printf to display result in 2 decimals

System.*out*.printf("Annual Average Rainfall: %.2f", average);

} //end main

**public** **static** **void** PopulateArray(**double**[] array){

//Scanner object declared to receive the values coming in from user

Scanner input = **new** Scanner(System.*in*);

//loop from 0 to 11 if array length is 12

//I used array.length to keep the method flexible to any number of months input

**for**(**int** index=0; index < array.length; index++){

//asking user for the rainfall and using the index plus 1 as it started at 0

System.*out*.print("Enter rainfall in cm for month "+ (index+1) + " ");

//assign the input value to the index position of array, allowing a

//decimal value

array[index] = input.nextDouble();

}

//for clean code can close the input object once method completed

input.close();

}//end method PopulateArray

**public** **static** **double** CalculateAverage(**double**[] array){

//declare and initialize local variable to keep running total required to

//calculate average

**double** total = 0;

// declare variable in loop, and loop from 0 to 11 if array length is 12

**for** (**int** index=0; index <array.length; index++){

//use the shorthand operand to add value to the total variable

total += array[index];

}//end for loop

//I used array.length to keep the method flexible to any number of months inputed

**return** (total/array.length); //send back the double average result

}//end method CalculateAverage

}//end class rainfallTester

**OUTPUT**

Enter rainfall in cm for month 1 116.7

Enter rainfall in cm for month 2 87.8

Enter rainfall in cm for month 3 94.7

Enter rainfall in cm for month 4 72.0

Enter rainfall in cm for month 5 75.3

Enter rainfall in cm for month 6 79.6

Enter rainfall in cm for month 7 86.5

Enter rainfall in cm for month 8 107.8

Enter rainfall in cm for month 9 100.3

Enter rainfall in cm for month 10 128.9

Enter rainfall in cm for month 11 120.3

Enter rainfall in cm for month 12 123.2

Annual Average Rainfall: 99.43

**public** **class** SelectionSort {

/\*\* Patricia Organ - 01110489 - Assignment 5 Q2

\* You are required to create an application, which creates an unsorted array

\* and passes it to a selection sort method for sorting. The application should

\* also contain a method to display the array before and after sorting.

\*/

**public** **static** **void** main(String[] args) {

// declare and initialize the Array with unsorted values

//making the assumption it did not matter how the array was received so hard

//coding it other alternatives could have been to generate a random number array,

//or to ask the user via jOptionPane for values like the size of array and then

//individual values to populate the array

**int**[] myArray = {1,4,53,3,7,123,543,653,987,10};

//write to console before print method with the heading

System.*out*.println("Array before sorting:");

//call the PrintArray method and pass by reference myArray

*PrintArray*(myArray);

//call the sorting method passing by reference myArray and the starting position

//as int

*SortingArray*(myArray,0);

//write to console before print method with the heading

System.*out*.println("\nArray after sorting:");

//call the PrintArray method and pass by reference myArray

*PrintArray*(myArray);

}//end main method

**public** **static** **void** SortingArray(**int**[] array, **int** start){

// I have chosen recursion as my method type, alternative solutions would have been

// to do 2 nested for loops reasons for choosing one over the other might depend

// on the size of the array in this case the size is small so the overhead on

// the stack is not going to be high

//declare variable to hold the minimum value

**int** min;

//start by assigning the value of the first position to start comparing

//in the array, starting point decided by the passed value

min = array[start]; //array passed by reference

//loop through the array starting from the next position from start

**for** (**int** i = start+1;i < array.length; i++ ){

//compare min value to the value of position i in array

**if** (array[i] < min){

//assign this new value to the start position

array[start] = array[i];

//swap the old min into that current position i in array

array[i] = min;

//now update the min value so it has the lowest value so far

min = array[start];

}//end if

}//end for

//need a base case for recursion so don't need to call again when argument

//for start +1 passed is equal to last position in array,

//when there is only one cell left to sort it is already sorted

**if** ((start + 1) != (array.length - 1)){

//recursively call the sorting method again with a new start position over one

*SortingArray*(array, start+1);

}

}//end method SortingArray

**public** **static** **void** PrintArray(**int**[] array){

//using an enhanced for statement

//display the array values in console with a loop through each element

**for** (**int** value: array){

System.*out*.print(value + " ");

}

}//end method PrintArray

}//end Class SelctionSort

**OUTPUT**

Array before sorting:

1 4 53 3 7 123 543 653 987 10

Array after sorting:

1 3 4 7 10 53 123 543 653 987