Lecture 9

Passing An Array as a Parameter to a Method

When a variable is passed as a parameter to a method in Java the net result depends on whether the variable is of primitive type, such as a double or int, or is a reference variable, such as an array. The following application illustrates this point.

The class PassArray is, for the most part, printlines. It does use the class Utility, and the file Utility.class must be in the same directory as PassArray.java.

```java
public class PassArray {
    public static double increment(double x) {
        x = x+2;
        return x;
    }

    public static void main(String[] args) {
        double x = 7.23;
        System.out.println(" The original value of x = " + x);
        double y = increment(x);
        System.out.println(" After incrementing x ");
        System.out.println(" the incremented value is y = " + y );
        System.out.println(" we still have x = " + x);
        System.out.println();
        double num[] = {23,18, 42, 31,12, 29};
        String out1 = " The original values of the array num[] are:";
        System.out.println(out1);
        int n = num.length;
        for (int i = 0; i < n; i++)
            System.out.print(" "+num[i]+ " ");
        Utility.bubblesort(num);
        String out2 = "\n After running the array num[] through bubble sort";
        System.out.println(out2);
        System.out.println(" the values of the num[i] are now: ");
    }
}
```
for (int i = 0; i < n; i++)
    System.out.print(" "+num[i]+ " ");      //1

System.out.println();

} // end main(String args[])

} // end class PassArray

The output follows:

The original value of x = 7.23
After incrementing x
the incremented value is y = 9.23
we still have x = 7.23

The original values of the array num[] are:
23.0 18.0 42.0 31.0 12.0 29.0
After running the array num[] through bubble sort
the values of the num[i] are now:
12.0 18.0 23.0 29.0 31.0 42.0

The important thing to note is that the primitive type was
incremented by 2, but the variable x retained its value 7.23. This is
because java passes a copy of the value of x to the method call
increment(x); the value of the variable x is kept at 7.23.

In contrast, when the array num is passed to the method
bubblesort()

double num[] = {23,18, 42, 31,12, 29};
int n = num.length;

Utility.bubblesort(num);

each of the elements in the array in the original program is changed.
The array values num[] are now defined to be (in //1) in increasing
order:
In sum, variables of primitive type, such as double or int, are passed "call-by-value". Non primitive variables are passed "call-by-reference".

Multiple Arrays

A multiple array is one that is indexed by two (or more) variables. For example

```java
int student = new[8][3];
```

defines a 8 x 3 array of 24 elements whose individual elements are of type int. The elements, if listed would be

```java
student[0][0], student[0][1], student[0][2],
student[1][0], student[1][1], student[1][2],
student[2][0], student[2][1], student[2][2],
student[3][0], student[3][1], student[3][2],
student[4][0], student[4][1], student[4][2],
student[5][0], student[5][1], student[5][2],
student[6][0], student[6][1], student[6][2],
student[7][0], student[7][1], student[7][2]
```

Note that the particular element student[i][j] is an element of the (i+1)st row and (j+1)st column in the rectangular displayed above.

In the example that follows student[i][j] will be the score the ith student received on the jth exam, for i = 0,..7 and i = 0,1,2. The 0th exam will correspond to the first hour exam, the 1st to the second, and the 2nd to the final.

Our initial code shows one way of defining a double array:

```java
import javax.swing.*;

public class StudentScores {
    public static void main(String args[])
    {
        int[][] student = {{70,80,90},{91,92,41},{73,66,70},
                           {90,85,99},{94,72,81},{93,85,99},
                           {76,89,76},{98,88,90}};

        System.out.println("number of rows = " + student.length);
    }
}
```
System.out.print("number of columns = ");
System.out.println(student[0].length+"\n");

for (int i = 0; i < 8; i++)
{
    for(int j = 0; j < 3; j++)
    { System.out.print("student["+i+"]["+j+"] = ");
        System.out.print(student[i][j] + " ");
    }
    System.out.println(" ");
} // end for loop

} // end main
} // end StudentScores

The line

    int[][] student = {{70,80, 90},{91,92, 41},{73,66, 70},
    {90,85, 99},{94,72, 81},{93,85, 99},
    {76,89, 76},{98,88, 90}};

...can be considered as one that first defines a single array

    student[0] = {{70,80, 90},.., student[7] = {98,88, 90}

which is an array of 8 elements. The double array is then defined by

    student[0][0] = 70, student[0][1] = 80, student[0][2] = 90

The next few lines

    System.out.println("number of rows = " + student.length);
    System.out.print("number of columns = ");
    System.out.println(student[0].length+"\n");

reflect this structure. The length of the array student is given by student.length, and is equal to 8. The length of the array student[0] is given by student[0].length, and is equal to 3.

The output of the program is:
Information about the rows and columns is gained by adding the lines printed in blue below to the main() method:

```java
public class StudentScores {
    public static void main(String args[]) {
        int[][] student = {{70,80,90},{91,92,41},{73,66,70},
                          {90,85,99},{94,72,81},{93,85,99},
                          {76,89,76},{98,88,90}};

        System.out.println("number of rows = " + student.length);
        System.out.println("number of columns = ");
        System.out.println(student[0].length + ",");

        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print("student[" + i + "][" + j + "] = ");
                System.out.print(student[i][j] + " ");
            }
            System.out.println();
        } // end for loop

        System.out.println("Grades on third exam (j = 2): ");
        for (int i = 0; i < 8; i++)
            System.out.print(" "+ student[i][2]);
        System.out.println();

        System.out.println("Grades for for fifth student (i = 4) ");
        for (int j = 0; j < 3; j++)
            System.out.print(" "+student[4][j]);
        System.out.println();
    } // end main
} // end StudentScores
```
The additional output of these new lines is:

Grades on third exam (j = 2):
90 41 70 99 81 99 76 90
Grades for fifth student (i = 4)
94 72 81

Suppose next you wished to add two methods to the class
StudentScores that would determine the maximum of each row or column
element. The methods for doing this, columnmax(int a[][], int c) and
rowmax(int a[][], int r), are defined by standard techniques. The
complete program, with the new methods and the corresponding addition
to main() printed in blue follows:

```java
public class StudentScores {
    public static int columnmax(int a[][], int c) {
        int n = a.length;
        int M = a[0][c];
        for (int i = 1; i < n; i++)
            if (a[i][c] > M) M = a[i][c];
        return M;
    }

    public static int rowmax(int a[][], int r) {
        int n = a[0].length;
        int M = a[r][0];
        for (int j = 1; j < n; j++)
            if (a[r][j] > M) M = a[r][j];
        return M;
    }

    public static void main(String args[]) {
        int[][] student = {
            {70, 80, 90}, {91, 92, 41}, {73, 66, 70},
            {90, 85, 99}, {94, 72, 81}, {93, 85, 99},
            {76, 89, 76}, {98, 88, 90}};

        System.out.println("number of rows = " + student.length);
        System.out.print("number of columns = ");
        System.out.println(student[0].length + "\n");

        for (int i = 0; i < 8; i++)
```

{ 
    for (int j = 0; j < 3; j++)
    {
        System.out.print("student["+i+"]["+j+"] = " );
        System.out.print(student[i][j] + " ");
    }
    System.out.println(" ");
}

System.out.println("Grades on third exam (j = 2): ");
for (int i = 0; i < 8; i++)
    System.out.print(" + student[i][2]);
System.out.println();

System.out.println("Grades for fifth student (i = 4)");
for (int j = 0; j < 3; j++)
    System.out.print(" +student[4][j]);
System.out.println();

StudentScores s = new StudentScores();
System.out.println("Max in second row = ");
System.out.println(s.rowmax(student, 1));
System.out.println("Max in third column = ");
System.out.println(s.columnmax(student, 2));

System.out.println("\n\n");
}  // end main()
}// end StudentScores

The complete output of the program is:
number of rows = 8
number of columns = 3

student[0][0] = 70  student[0][1] = 80  student[0][2] = 90
student[1][0] = 91  student[1][1] = 92  student[1][2] = 41
student[4][0] = 94  student[4][1] = 72  student[4][2] = 81
student[5][0] = 93  student[5][1] = 85  student[5][2] = 99
student[6][0] = 76  student[6][1] = 89  student[6][2] = 76
student[7][0] = 98  student[7][1] = 88  student[7][2] = 90

Grades on third exam (j = 2):
   90  41  70  99  81  99  76  90

Grades for fifth student (i = 4)
   94  72  81

Max in second row = 92
Max in third column = 99
Homework Problem: Suppose you are given a function $f(x,y)$ defined by

$$f(x,y) = x \cos(x+y) - y \sin(x-y)$$

where $x$ and $y$ are of type double. You are to write a java application
that will compute the values of 1600 equally spaced values of $x$ and $y$
that fall in the rectangle $0 \leq x \leq 6.28$, $0 \leq y \leq 6.28$, and then compute
the minimum and maximum of these 1600 values. Discuss how you might do
this.

In addition, discuss how you would print all of the values of $(x, y)$
such that the $|f(x,y)| < 1/10$. A typical statement to be printed out
would be

for $x = --$ and $y = --$, $f(x,y) = --$

where the double blanks -- are to be actual numbers, and the one for
$f(x,y)$ is to be -1/10 and 1/10

Outline:

(1) The points at which the function is to be evaluated are

$$x[i] = i \times 6.28/39, \ i = 0, 1, \ldots, 39$$
$$y[j] = j \times 6.28/39, \ j = 0, 1, \ldots, 39$$

(Why do you use 39 instead of 40? What are the corresponding array
structures?)

(2) The function can be written as a java method:

public static double f(double x, double y)
{
    double u = Math.cos(x+y);
    double v = Math.sin(x-y)
    // Then what? What is to be returned?
}

(3) What is the array structure for holding the 1600 calculated values?

(4) How do you calculate the minimum? The maximum?

(5) How do you print out those values whose absolute value is $< 1/10$?