MAX, MIN, AVERAGE

The use of an accumulator can be extended beyond sums and products. Three very common programming idioms involve loops that find the maximum, minimum and average of a series of values.

Finding the maximum of a series of values is rather like kids playing “King of the Mountain” on the playground – a kid gets to be king until a larger kid comes along and throws him off the mountain. We can adapt the accumulator idiom to this, using the accumulator to hold the current “king.”

To fashion a loop that finds a maximum (minimum):

1. Before the loop begins, declare the king.
2. If possible, initialize the king to be absolutely smaller (larger) than any of the values in the series. This insures that the first value will become king on the first loop cycle.
3. If step 2 is not possible, initialize the king to the first value in the series.
4. During each loop cycle, if the next value is greater than (less than) the king then replace the king’s value with it.
### Example
This code fragment illustrates a loop that finds the maximum and minimum of a series of exam scores that you know range from 0 to 100.

```java
int max = -1;  // initialize the largest
int min = 101;  // initialize the smallest
while ( . . . )
{
    // obtain the next value
    . . .
    if ( value > max ) max = value;  // update max
    if ( value < min ) min = value;  // update min
}
```

### Example
This code fragment illustrates a loop that finds the maximum and minimum of a series values whose range is unknown.

```java
double max = 0;  // these initializations serve to keep
double min = 0;  // the Java compiler happy
int count = 0;   // count the number of loop cycles
while ( . . . )
{
    // obtain the next value
    . . .
    if ( count == 0 )     // initialize max and min to
        max = min = value; // the first value of the series
    else
    {
        if ( value > max ) max = value;  // update max
        if ( value < min ) min = value;  // update min
    }
    count++;
}
```
To fashion a loop that averages a series of values:

1. Construct a loop that counts the number of values and accumulates their sum.
2. After the loop quits divide the sum by the count, using a test to make sure the count is not zero (otherwise, you’ll do the nonsensical division by 0).

**Example**
This code fragment illustrates the programming idiom for averaging a series of values.

```c
int count = 0;    // initialize the count
double sum = 0.0; // initialize the accumulator
while ( . . . )
{
    // obtain the next value
    . . .
    sum += value; // add the value to the accumulator
    count++;      // count it
}
double avg = (count > 0) ? sum / count : 0.0;
```
Max, Min, Average Example – Peoples’ Heights

Pretend we have data for an indefinite number of people consisting of each person’s first name and height in inches. The Java application must read each person’s name and height in inches, break the height into feet and inches and print it along with the name.

When finished processing the data for all the people, the application must print the number of persons, the average height of a person, the largest height and the smallest. All heights must print out in feet and inches.

The design of this program is a combination of the input loop idiom (covered in the topic Input Loops) and the idioms covered in this topic.

```
initialize sum and counter to zero
initialize maxHeight to zero and minHeight to a value bigger than all heights
while there's another person do
  read his or her name
  read his or her height in inches
  add height in inches onto the sum
  increment the counter
  if height in inches > maxHeight then
    maxHeight = height in inches
  end if
  if height in inches < minHeight then
    minHeight = height in inches
  end if
  calculate the height in feet and inches
  print the person’s name and height in feet and inches
end while
if count > 0 then average = sum / counter end if
calculate the maxHeight and minHeight in feet and inches
output counter, average, maxHeight and minHeight
```
The complete Java application is given below.

```java
import java.util.Scanner;

public class PeoplesAverageHeight {
    public static void main( String [] args ) {
        // declare data
        String name;       // person's name
        int hgtFt, hgtIn;  // person's height (feet, inches)
        // initialize counters and accumulators
        int sum, count, minHgt, maxHgt;
        sum = count = maxHgt = 0;
        minHgt = 120; // assume nobody's 10' tall
        // create Scanner
        Scanner in = new Scanner( System.in );
        // prompt for and input person's name
        System.out.print( "Name? " );
        name = in.next( );
        // quit if sentinel is read
        while ( ! name.equalsIgnoreCase( "quit" ) )
        {
            // input remaining data
            System.out.print( "Height (inches)? " );
            hgtIn = in.nextInt( );
            // update counters and accumulators
            sum += hgtIn;
            count++;
            if ( hgtIn > maxHgt ) maxHgt = hgtIn;
            if ( hgtIn < minHgt ) minHgt = hgtIn;
            // process data and print
            hgtFt = hgtIn / 12;
            hgtIn %= 12;
            System.out.println( name + " is " + hgtFt + "' " + hgtIn + "" tall" );
            // prompt for and input person's name
            System.out.print( "Name? " );
            name = in.next( );
        }
    }
}
```
38         }
39         double avg = count > 0 ? (double)sum/count : 0.0;
40         System.out.println( "\n" + count
41             + " people processed\n"
42             + "Average height: " + (int)avg/12 + "' "
43             + avg%12 + "\"\n"
44             + "Largest height: " + maxHgt/12 + "' "
45             + maxHgt%12 + "\"\n"
46             + "Smallest height: " + minHgt/12 + "' "
47             + minHgt%12 + "\"\n" );
48     }
49 }
### Programming Exercises

1. Extend one or more of your solutions to Exercise #1 of the topic *Input Loops* to include these **additional** requirements:

   After data for all workers has been processed, print:
   - The number of workers
   - The total pay of all workers
   - The average pay of a worker
   - The largest pay given to a worker
   - The smallest pay given to a worker

2. Extend one or more of your solutions to Exercise #2 of the topic *Input Loops* to include these **additional** requirements:

   After data for all students has been processed, print:
   - The number of students
   - The total number of credits all students have enrolled in
   - The total tuition of all students
   - The average number of credits per student

3. Extend one or more of your solutions to Exercise #3 of the topic *Input Loops* to include these **additional** requirements:

   After data for all patients has been processed, print:
   - The number of patients
   - The average height per patient (printed as feet and inches)
   - The average weight per patient
   - The largest BMI value and the name of the patient who has it
   - The smallest BMI value and the name of the patient who has it

4. Extend one or more of your solutions to Exercise #4 of the topic *Input Loops* to include these **additional** requirements:

   After data for all CDs has been processed, print:
   - The number of CDs
   - The total extended price of all CDs (i.e. the total of all the quantity times price)
   - The largest number of individual CDs sold along with its description
   - The smallest number of individual CDs sold along with its description
5. Modify the application **PeoplesAverageHeight** (shown on page 5) so that, in addition to the maximum and minimum heights, it prints the names of the people who have the maximum and minimum heights.

6. You have a data file containing a list of boys’ names and heights followed by a list girls’ names and heights. Each list is terminated by the sentinel value “stop.”

   Write a Java application to calculate and print the number of boys and their average height; the number of girls and their average height; and the total number of children and the average height over all children.

   A sample data file is shown at right.

```
data.txt
Tom 66
Tyler 73
Kevin 72
stop
Mary 67
Pam 59
stop
```

7. You have a data file containing a list of male patients followed by a list of female patients. Data for each patient consists of his or her name (a string that may include embedded blanks), weight in pounds (a floating-point value) and height in feet and inches (two integers). Each list is terminated by the sentinel value “stop.”

   Write a Java application to calculate and print the number of men and their average height, weight and BMI; the number of women and their average height, weight and BMI; and the total number of patients and their average height, weight and BMI.