Introduction to Algorithms and Data Structures

Lecture 4 – Structuring Data:
Multidimensional Arrays, Arrays of Objects, and Objects Containing Arrays

Grouping Data

- We don't buy individual eggs; we buy them in units of 12 ("dozens").
  - We often think in terms of these groups and not the individual members.
  - Examples - classes, baseball teams, encyclopedia sets, etc.
- It is extremely helpful to be able to group data items that are closely related, e.g., class grades on a test, pay rates for a group of employees, etc.

Declaring Arrays

- Instead of writing:

```
int    x;
```

  we can write:

```
int[] x =  new int[10];
```

  the name **x** refers to the collection (or ***array***) of integer values, which can contain up to 10 values.

- The general format is:

  BaseType **[]** *arrayName* **=** **new** *BaseType***[***Length***]***;*

Using An Array

- We can assign a value to any element in the array by specify the array by name and its index:

```
x[0] = 87;
x[1] = 90;
… … …
x[9] = 93;
```

lowest index

highest index

Using An Array (continued)

- An index can be any integer or character literal, constant, variable or expression:

```
x[6] = x[5] + 4;
x[Five] = 34;
x[i+1] = x[i] + 3;
```

- This is really useful, because we do not want to have to write separate statement to assign values to each array element.

Using a Counting Loop To Set An Array

- Counting loops are really useful when manipulating arrays:

```
for (i = 0;  i < 10;  i++)
   x[i] = keyb.nextInt();
```

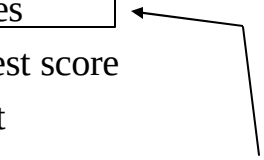
Example – Finding the High Score

We want to find the highest grade among a group of test scores. There will be than 10 grades.

- Available input – the 10 test scores
- Required output – the highest score
- Algorithm:
  1. Read the scores
  2. Find the highest score
  3. Print the result

Refining the Algorithm

1. Read the scores
2. Find the highest score
3. Print the result



1.1 For each score:
1.1.1 Prompt the user
1.1.2 Read the scores

Refining the Algorithm

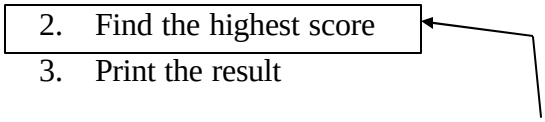
1.1 For each score:

1.1.1 Prompt the user

1.1.2 Read the scores

2. Find the highest score

3. Print the result



2.1 For each score:

2.1.1 If it is higher than the current highest score

2.1.1.1 It is now the new highest score

FirstArray.java

```
import java.util.Scanner;

public class FirstArray {
  // FirstArray() - Find the highest score among a
  //                group of 10 scores
  public static void main(String[] args)  {
    Scanner keyb = new Scanner(System.in);
    int [] scores = new int[10];
    int i, maxScore = 0;

    // Read in the array
    for (i = 0;  i < x.length;  i++)  {
      System.out.println("Enter grade #" + i);
      scores[i] = keyb.nextInt();
    }
```

```
    // Find the high score
    for (i = 0;  i < x.length;  i++)
      if (x[i] > maxScore)
        maxScore = x[i];

    // Print the result
    System.out.println("The highest score is "
                                          + maxScore);
  }
}
```

Initializing Arrays

- You can initialize an entire array by writing:

  `int [] age = { 2, 12, 1};`

  `age` is now an array of length 3 where **`age[0]`** = 2, **`age[1]`** = 12 and **`age[2]`** = 1.

Arrays are Objects

- Arrays are a special type of object. When we write

  ```
  x = new int[10];
  ```

- We are allocating space for the object's data (i.e., properties and methods).
- The name **`x`** by itself refers to the address at which the object **`x`**'s data is stored.

Arrays as Parameters

- Even though Java normally passes parameters by value, it will passes arrays and other types of objects by reference (because the object's name ***is*** a reference).
- Passing an element in an array is different from passing the entire array, i.e.,
- **`f(x)`** and **`f(x[i])`** are very different

SecondArray.java

```
import java.util.Scanner;

public class SecondArray {
  // SecondArray() - Shows how passing array
  //                 parameters works
  public static void main(String[] args)  {
    int [] x = { 1, 2, 3, 4};

    // First use the scalar parameter
    myFunc(x[0]);
    for (int i = 0; i < x.length;  i++)
      System.out.print(" " + x[i]);
    System.out.println();
```

```
    // Second use the array parameter
    myFunc(x);
    for (int i = 0; i < x.length;  i++)
      System.out.print(" " + x[i]);
    System.out.println();

  }
```

```
  // myFunc() - The array method
  public static void myFunc(int [] a)  {
    System.out.println("This is the array method.");
    for (int i = 0; i < a.length; i++)
      a[i] = 5 * (i + 1);
  }

  // myFunc() - The scalar method
  public static void myFunc(int a) {
    System.out.println("This is the scalar
  method.");
    a = 99;
  }
}
```

Comparing Arrays

- Arrays, like any other method, cannot be compared using the == operator. This would only see if they hold the same reference.
- One needs to write an **`equals`** method to compare the arrays.

ArrayEquals.java

```
public class ArrayEquals  {
  public static void main(String [] args) {
    int [] x = new int[10],
           y = new int[10];
    int i;
    for (i = 0;  i < x.length;  i++)
      x[i] = i;
    for (i = 0;  i < y.length; i++)
      y[i] = i;

    if (x == y)
      System.out.println
             ("x and y are equal by \"==\"");
    else
      System.out.println
             ("x and y are not equal by \"==\"");
```

```
    if (arrayEquals(x, y))
      System.out.println
           ("x and y are equal by \"equalsArray\"");
    else
      System.out.println
       ("x and y are not equal by \"equalsArray\"");
  }
```

```
  public static boolean
            arrayEquals(int [] a, int [] b)  {
    if (a.length != b.length)
      return false;

    for (int i = 0;  i < a.length;  i++)
      if (a[i] != b[i])
        return false;

    return true;
  }
}
```

Output

```
x and y are not equal by "=="
x and y are equal by "equalsArray"
```

Parameters for **`main()`**

- Sometimes you may wish for the **`main()`** method to receive parameters. Because this is usually via the command line, they are also called command line parameters.
- These are passed to the **`main()`** method as an array of **`String`**s. Like any other array, it has a length property that we can use.

PrintCommandParameters.java

```
public class PrintCommandParameters  {
  public static void main(String [] args) {
    for (int i = 0;  i < args.length;  i++)
      System.out.println("Parameter #" + i +
                         " is " + args[i]);

  }
}
```

Output

```
>Java PrintCommandParameters this is a test
Parameter #0 is this
Parameter #0 is is
Parameter #0 is a
Parameter #0 is test
```

Grades.java – A Class With an Array

```
import java.util.Scanner;

public class Grades {
  final static int numStudents = 30,
                   numExams = 4;

  static int[][] grades
              = new int [numStudents][numExams];
  static int[] averages = new int[numStudents];
```

```
// CalcAverages() - Calculate the term
//     averages for a class
//     The average is based on
//     four exams
public static void main(String[] args)  {
  //Get the grades, find the averages and
  //print them
  readGrades();
  findAverages();
  writeGrades();
}
```

```
// readGrades() - Read the complete set of
//                grades
public static void readGrades()  {
  Scanner keyb = new Scanner(System.in);
  String  inString = new String();
  int i, j;

  // Get each students grade
  for (i = 0;  i < numStudents;  i++) {
    //Get the next grade for this student
    for (j = 0;  j < numExams;  j++) {
      System.out.println("Grade on test #" + j
              + " for student # " + i + "\t?");
      grades[i][j] = keyb.nextInt();
    }
    //Skip one line for clarity
    System.out.println();
  }
}
```

```
// findAverages() - Find the average for each
//                  student
public static void findAverages()  {
  int i, j, sum;

  for (i = 0;  i < numStudents;  i++) {
    sum = 0;
    for (j = 0;  j < numExams;  j++)
      sum += grades[i][j];
    averages[i] = sum/numExams;
  }
}
```

```
// WriteAverage() - Output the grades and
//                  average for each student
public static void writeGrades()  {
  int i, j;

  // Print a heading
  System.out.println("Student Exam1\tExam2\t"
                   + "Exam3\tExam4\tAverage");

  for (i = 0;  i < numStudents;  i++) {
    // Number each line, then print the grades
    // and the average for the enxt student
    System.out.print(i);
```

```
  for (j = 0;  j < numExams; j++) {
        System.out.print("\t");
        System.out.print(grades[i][j]);
      }
      System.out.println("\t" + averages[i] );
    }
  }
}
```

Revising **`Grades.java`**

```
import java.util.Scanner;

public class Grades  {
  private final int maxStudents = 30,
                    numExams = 4;

  private int numStudents;
  private int[][] grades;
  private int [] averages;

  // Grades() - The default constructor
  public Grades()  {
    // Constructing the array
    grades = new int[maxStudents][numExams];
  }
```

```
// readGrades() - Read the complete set of
//                grades
public int readGrades() {
  int j;
  String inString = new String();
  Scanner keyb = new Scanner(System.in);

  System.out.println("Enter a negative grade "
            + "for exam 1 to finish");
  numStudents = 0; //There are no students yet
                   // Get each students grade
  do {
    //Get the first exam grades.  If it's
    // non-negative, get the rest.  If
    // it IS negative, there are no more
    //  students.
```

```
    System.out.println("Grade on test #1"
              + " for student # "
              + (numStudents+1) + "\t?");
    grades[numStudents][0] = keyb.nextInt();
    if (grades[numStudents][0] < 0)
      break;
    for (j = 1;  j < numExams;  j++) {
      System.out.println("Grade on test #"
              + (j+1)
              + " for student # "
              + (numStudents+1) + "\t?");
      grades[numStudents][j] = keyb.nextInt();
    }
```

```
    //Skip a line for clarity
    System.out.println();
    numStudents++;
  } while (numStudents < maxStudents);

  if (numStudents == maxStudents)
    --numStudents;
  return numStudents;
}
```

```
// FindAverages() - Find the average for each
//                  student
public void findAverages()  {
  int i, j, sum;
  averages = new int[numStudents];
  for (i = 0;  i < numStudents;  i++) {
    sum = 0;
    for (j = 0;  j < numExams;  j++)
      sum += grades[i][j];

    averages[i] = sum/numExams;
  }
}
```

```
  // WriteAverage() - Output the grades and
  //                  average for each student
  public void writeGrades()  {
    int i, j;

    // Print a heading
    System.out.println("Student\tExam1\t"
              + "Exam2\tExam3\tExam4\tAverage");

    for (i = 0;  i < numStudents;  i++) {
      // Number each line, then print the grades
      // and the average for the next student
      System.out.print(i);
      for (j = 0;  j < numExams; j++)
        System.out.print("\t" + grades[i][j]);
      System.out.println("\t" + averages[i]);
    }
  }
}
```

CalcAverages.java

```
// CalcAverages() - Calculate the term
//     averages for a class
//     The average is based on four exams
public class CalcAverages  {
  public static void main(String[] args)  {
    Grades myClass = new Grades();
    int nStudents;
    //Get the grades, find the averages and
    // print them
    nStudents = myClass.readGrades();
    myClass.findAverages();
    myClass.writeGrades();
  }
}
```

Matrices

- A ***matrix*** (the plural is *matrices*) is a two-dimensional array of numbers.
- While it has special uses in mathematics, it is extremely useful to be able to work with 2- and 3-dimensional arrays of data of various types.

Matrix.java

```
import java.util.Scanner;

public class Matrix  {
  public final int numRows = 4,
                   numColumns = 4;

  private int[][] matrix;

  public Matrix()  {
    matrix = new int[numRows][numColumns];
  }
```

```
//read() - Read in a matrix
public void read()  {
  int i, j;
  Scanner keyb = new Scanner(System.in);

  for (i = 0;  i < numRows; i++)  {
    System.out.println("Enter row #" + (i+1)
                       + "\t?");
    for (j = 0;  j < numColumns;  j++) {
      System.out.println("Enter x[" + i
                         + "][" + j + "]");
      matrix[i][j] = keyb.nextInt();
    }
  }
}
```

```
// write() – Write an i x j matrix
void write()  {
  int i, j;

  for (i = 0;  i < numRows;  i++) {
    for (j = 0; j < numColumns;  j++)
      System.out.print("\t" + matrix[i][j]);
    System.out.println();
  }
}
```

```
  // multmatrix() – Multiply this x b to get c
  public Matrix mult(Matrix b)  {
    Matrix c = new Matrix();
    int i, j, k;

    for (i = 0;  i < numRows; i++)
      for (j = 0;  j < numColumns;  j++) {
        c.matrix[i][j] = 0;
        for (k = 0;  k < numRows;  k++)
          c.matrix[i][j]
                  += matrix[i][k]*b.matrix[k][j];
      }
    return c;
  }
}
```

Using An Array of Objects

- Sometimes the array with which we are going to work is an array of objects. This is really not much different from working with an array of integers or doubles or Strings.

StudentRecord.java

```
import java.util.Scanner;

public class StudentRecord {
   private String firstName;
   private String lastName;
   private int year; // Year of graduation
   private double gpa;

   // Accessors
   public String getFirstName()  {
     return firstName;
   }

   public String getLastName()  {
     return lastName;
   }
```

```
   public int getYear()  {
     return year;
   }

   public double getGpa()  {
     return gpa;
   }

   // Mutators
   public void setFirstName(String fn)  {
     firstName = fn;
   }

   public void setLastName(String ln)  {
     lastName = ln;
   }
```

```
   public void setYear(int yr)  {
     year = yr;
   }

   public void setGpa(double gpAv)  {
     gpa = gpAv;
   }
}
```

StudentRecords.java

```
import java.util.Scanner;

public class StudentRecords{
  public final int maxStudents = 15;
  private int numStudents;
  public StudentRecord[] students;

  public StudentRecords()  {
    students = new StudentRecord[maxStudents];
    for  (int i = 0;  i < maxStudents;  i++)
      students[i] = new StudentRecord();
    //Initially, no student records have been
    // entered
    numStudents = 0;
  }
```

```
//ReadRecords() - Read in the input records
public int readRecords()  {
  Scanner keyb = new Scanner(System.in);
  String inString = new String();
  int newYear;
  double newGPA;
  //Tell the user how to end the input
  System.out.println("Enter \"end\" to indicate"
                  + " no more students");

  // For each student, input the first and last
  // name, year of graduation and grade point
  // average
  for  (numStudents = 0;
           numStudents < maxStudents;
                              numStudents++) {
    System.out.println("First name\t?");
    inString = keyb.next();
```

```
    //If they entered "end", there aren't any
    // more  records
    if (inString.equals("end"))
      break;
    students[numStudents].setFirstName(
                               inString.trim());
    System.out.print("Last name\t?");
    inString = keyb.next();
    students[numStudents].setLastName(
                              inString.trim());
    System.out.println("Year\t?");
    newYear = keyb.nextInt();
    students[numStudents].setYear(newYear);

    System.out.println("Grade point average\t?");
    newGPA = keyb.nextDouble();
    students[numStudents].setGpa(newGPA);
  }
  return numStudents;
}
```

```
  //WriteRecord() - Enter the student record
  public void writeRecord(int index)  {
    // Make sure that all significant places print
    //Print the record, using four spaces for year
    // and gpa
    System.out.println(
          students[index].getFirstName() + "\t"
        + students[index].getLastName() + "\t"
        + students[index].getYear() + "\t"
        + students[index].getGpa());
  }
}
```

DeansList.java

```
import java.util.Scanner;

// DeansList() - Read Student records and print
//               the Data on studetns on the
//               Dean's List
public class DeansList  {
  public static void main(String[]args)  {
    StudentRecords stud = new StudentRecords();
    int            i, numStudents = 0;

    //Input the data
    numStudents = stud.readRecords();
```

```
    //Print the data for students with
    // gpa's of 3.5 or higher
    for  (i = 0;  i < numStudents;  i++)
      if (stud.students[i].getGpa() >= 3.5)
        stud.writeRecord(i);
  }
}
```