Announcements

- Program 4 due next Friday
  - Last day of class

- Final exam
  - 6/13, 8–11 AM
  - SN014
Questions?
Today in COMP 110

- Array basics
Think back to Lab 4

- You wrote a program to read in a list of basketball scores from the user and output a bunch of statistics
System.out.println("Enter the list of basketball scores "+
"(enter a negative number to end your list): ");
while ((score = keyboard.nextInt()) >= 0) {
    totalGames++;
scoreSum += score;
    if (score >= 90)
        totalGamesOver90++;

    if (score > highestScore)
        highestScore = score;

    if (score < lowestScore)
        lowestScore = score;
}
if (totalGames > 0) {
    // some stuff
    double average = (double) scoreSum / (double) totalGames;
    // some other stuff
}
System.out.println("Enter the list of basketball scores "+
"(enter a negative number to end your list): ");

while ((score = keyboard.nextInt()) >= 0) {
    totalGames++;
    scoreSum += score;
}

if (totalGames > 0) {
    double average = (double) scoreSum / (double) totalGames;
    System.out.println("Average score: "+ average);
}
What if...

• ...we wanted to know which of the scores entered were
  ◦ above average?
  ◦ below average?

• How would we do it?

• Let’s simplify this a little first
Assume we knew we had exactly 5 scores

```java
System.out.println("Enter 5 basketball scores:");

for (int i = 0; i < 5; i++)
{
    scoreSum += keyboard.nextInt();
}

double average = (double) scoreSum / 5.0;
System.out.println("Average score: " + average);
```
...we wanted to know which of the scores entered were
  ◦ above average?
  ◦ below average?

How would we do it?
System.out.println("Enter 5 basketball scores:");

int score1 = keyboard.nextInt();
int score2 = keyboard.nextInt();
int score3 = keyboard.nextInt();
int score4 = keyboard.nextInt();
int score5 = keyboard.nextInt();

double average =
    (double) (score1 + score2 + score3 + score4 + score5) / 5.0;

System.out.println("Average score: " + average);

// repeat this for each of the 5 scores
if (score1 > average)
    System.out.println(score1 + ": above average");
else if (score1 < average)
    System.out.println(score1 + ": below average");
else
    System.out.println(score1 + ": equal to the average");
What if we had 80 scores?

System.out.println("Enter 80 basketball scores: ");

int score1 = keyboard.nextInt();
int score2 = keyboard.nextInt();
int score3 = keyboard.nextInt();
// ...are we done yet?
int score23 = keyboard.nextInt();
int score24 = keyboard.nextInt();
int score25 = keyboard.nextInt();
// ...how about now?
int score67 = keyboard.nextInt();
int score68 = keyboard.nextInt();
// ...by here we would go crazy...
int score80 = keyboard.nextInt();
// ...whew!

double average = (double) (score1 + score2 + score3 + score4 +
    ... score23 + score24 + score25 + ... ) / 80.0;
System.out.println("Average score: "+average);
// now do below/above average check for all 80 scores
Arrays can solve this problem easily

An array is a collection of items of the same type

Like a list of variables, but with a nice, compact way to name them

A special kind of object in Java
Creating an array

```
int[] scores = new int[5];
```

- This is like declaring 5 strangely named variables of type `int`:
  - `scores[0]`
  - `scores[1]`
  - `scores[2]`
  - `scores[3]`
  - `scores[4]`
Variables such as scores[0] and scores[1] that have an integer expression in square brackets are known as:

- indexed variables, subscripted variables, array elements, or simply elements

An index or subscript is an integer expression inside the square brackets that indicates an array element.
Where have we seen the word index before?
- String’s indexOf method

Index numbers start with 0. They do NOT start with 1 or any other number.
The number inside square brackets can be any integer expression

- An integer: \( \text{scores}[3] \)
- Variable of type int: \( \text{scores}[\text{index}] \)
- Expression that evaluates to int: \( \text{scores}[\text{index} \times 3] \)

We can use these strangely named variables just like any other variables:

- \( \text{scores}[3] = 68; \)
- \( \text{scores}[4] = \text{scores}[4] + 3; \) // just made a 3-pointer!
- System.out.println(scores[1]);
The array itself is referred to by the name scores (in this particular case)

Indices

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>68</td>
<td>73</td>
<td>57</td>
<td>102</td>
<td>94</td>
</tr>
</tbody>
</table>

the array scores

scores[3]
System.out.println("Enter 5 basketball scores:");

int[] scores = new int[5];
int scoreSum = 0;
for (int i = 0; i < 5; i++)
{
    scores[i] = keyboard.nextInt();
    scoreSum += scores[i];
}
double average = (double) scoreSum / 5;
System.out.println("Average score: " + average);

for (int i = 0; i < 5; i++)
{
    if (scores[i] > average)
    System.out.println(scores[i] + ": above average");
    else if (scores[i] < average)
    System.out.println(scores[i] + ": below average");
    else
    System.out.println(scores[i] + ": equal to the average");
}
for–each

- You can also use another form of the for loop with collections (such as arrays)

```java
for (int s : scores)
{
    if (s > average)
        System.out.println(s + " : above average");
    else if (s < average)
        System.out.println(s + " : below average");
    else
        System.out.println(s + " : equal to the average");
}
```

- `s` takes on the value of each element of the array `score`, but you cannot change an element’s value this way
Syntax for creating an array:

```
Base_Type[] Array_Name = new Base_Type[Length]
```

Example:

```
int[] pressure = new int[100];
```

Alternatively:

```
int[] pressure;
pressure = new int[100];
```
Array details

- The base type can be any type
  ```java
double[] temperature = new double[7];
Student[] students = new Student[35];
  ```

- The number of elements in an array is its **length**, **size**, or **capacity**
  - temperature has 7 elements, temperature[0] through temperature[6]
  - students has 35 elements, students[0] through students[34]
Use a named constant

- Usually want to use a named constant when creating an array
  
  ```
  public static final int NUMBER_OF_READINGS = 100;
  int[] pressure = new int[NUMBER_OF_READINGS];
  ```
System.out.println("How many scores?");
int numScores = keyboard.nextInt();
int[] scores = new int[numScores];
Finding the length of an existing array

- An array is a special kind of object
  - It has one public instance variable: `length`
  - `length` is equal to the length of the array
    ```java
    Pet[] pets = new Pet[20];
    pets.length has the value 20
    ```
  - You cannot change the value of `length` because it is `final`
System.out.println("Enter 5 basketball scores:");

int[] scores = new int[5];
int scoreSum = 0;
for (int i = 0; i < scores.length; i++)
{
    scores[i] = keyboard.nextInt();
    scoreSum += scores[i];
}

double average = (double) scoreSum / scores.length;
System.out.println("Average score: " + average);

for (int i = 0; i < scores.length; i++)
{
    if (scores[i] > average)
        System.out.println(scores[i] + ": above average");
    else if (scores[i] < average)
        System.out.println(scores[i] + ": below average");
    else
        System.out.println(scores[i] + ": equal to the average");
}
Be careful with your indices

- Indices MUST be in bounds
  ```java
double[] entries = new double[5];
entries[5] = 3.7;  // ERROR! Index out of bounds
```

- Your code will compile if you are using an index that is out of bounds, but it will give you an error when you run your program
You can initialize arrays when you declare them
```
int[] scores = { 68, 97, 102 };
```

Equivalent to
```
int[] scores = new scores[3];
scores[0] = 68;
scores[1] = 97;
scores[2] = 102;
```