Structured Data in Java
Array Examples

Sample problem

- Given a set of test scores for a class of students, compute the average ***and report which scores were below the average***

- Solution:
  - We now need each score twice: once to compute the average and once more to compare against the average
  - We don't know how many students until the program is running
  - We cannot do this with our current knowledge of Java

can ^ now

Problem solution

- We need to:

Problem solution

- We need to:
  1. Access the data from the user

Problem solution

- We need to:
  1. Access the data from the user
  2. Create an array of the appropriate size

Problem solution

- We need to:
  1. Access the data from the user
  2. Create an array of the appropriate size
  3. Get a score and store in the array
     - Total all the scores as well

Problem solution

- We need to:
  1. Access the data from the user
  2. Create an array of the appropriate size
  3. Get a score and store in the array
     - Total all the scores as well
  4. Compute & report the average

Problem solution

- We need to:
  1. Access the data from the user
  2. Create an array of the appropriate size
  3. Get a score and store in the array
     - Total all the scores as well
  4. Compute & report the average
  5. Compare each score to the average

- Let's address these one at a time…

Problem solution

1. Access the data from the user
   - For this problem, we will assume that the user enters all scores into a textbox, separating each with a single blank character
   - This data will be given to our app as a one large String value
   - We can use methods of the String class to separate this one long string into an array of Strings
   - We can then convert each String into its corresponding integer value, storing them in a separate array

Problem solution

1. Access the data from the user

```
public void process(String input) {

   // All scores are text data in the string "input".
   // Use methods of the String class to create
   // an array of separate scores by splitting on the
   // blanks.
   String[] vals = input.split(" ");
   // Each element of vals is a String representation
   // of a score.
```

Problem solution

2. Create an array of the appropriate size

```
// Create an integer array that is the same size
// as the String array
int[] score = new int[vals.length];
```

Problem solution

3. Get a score and store in the array
   - Total all the scores as well

```
// Convert each score from String into an integer
// and store into array and also total them
int total = 0;
for (int i=0; i<score.length; i++) {
  score[i] = Integer.parseInt(vals[i]);
  total += score[i];
}
```

Problem solution

4. Compute & report the average
5. Compare each score to the average

```
// compute average
double average = (double)total/score.length;
out.println("The average was: " + average);
// compare each score against the average
for (int i=0; i<score.length; i++) {
  if (score[i] < average) {
    out.println("Score " + score[i] +
                " was less than the average");
  }
}
```

Sample execution

- Given that the input was:

```
"98 87 63 92 81 89 94 88 79 83"
```

- The program would produce this output:

```
The average was: 85.4
Score 63 was less than the average
Score 81 was less than the average
Score 79 was less than the average
Score 83 was less than the average
```

Sample problem #2

- Simulate the rolling of two dice many times, and report the percentage of times each value was rolled

- Solution:
  - We will use an array to keep track of multiple counters
    - One counter for each possible value rolled
  - For each roll, we increment the corresponding counter
  - The final percentages are produced by dividing the counter values by the total number of rolls

Problem #2 solution

```
public void process() {

   // Declare a constant for how many rolls we want.
   final int NUMROLLS = 1000000;  // Let's try a million

   // Declare an array of counters. All initialized to zero.
   // Note: we will not use the first two elements.
   int[] count = new int[13];

   // Create a Random object for simulating a die.
   Random rand = new Random();
```

Problem #2 solution

```
// Perform the desired number of rolls:
for (int i=0; i<NUMROLLS; i++) {

  // Roll the two dice.
  int die1 = rand.nextInt(6)+1;
  int die2 = rand.nextInt(6)+1;

  // Increment the corresponding counter.
  count[die1+die2]++;
}
```

Problem #2 solution

```
   // After all rolls are done, report percentages:
   for (int i=2; i<count.length; i++) {

      out.println("The value " + i + " was rolled " +
                  100.0*count[i]/NUMROLLS +
                  "% of the time.");
   }
}
```

Sample execution

```
The value 2 was rolled 2.7929% of the time.
The value 3 was rolled 5.5685% of the time.
The value 4 was rolled 8.3406% of the time.
The value 5 was rolled 11.1355% of the time.
The value 6 was rolled 13.8801% of the time.
The value 7 was rolled 16.6425% of the time.
The value 8 was rolled 13.8778% of the time.
The value 9 was rolled 11.1282% of the time.
The value 10 was rolled 8.3227% of the time.
The value 11 was rolled 5.5452% of the time.
The value 12 was rolled 2.766% of the time.
```