Array basics
Sometimes, we have data that have some natural structure to them

A few examples:
- Texts are sequences of characters
- Images are matrices of pixels
- Classes contain sets of students

Java provides a variety of classes and tools called data structures
- They help organize your data
- They make it convenient to access and update your data
Some common data structure classes in Java

- **Array/Arrays** (the data structure we will cover)
- ArrayList
- HashSet
- LinkedHashSet
- LinkedList
- TreeSet
- Vector
- HashMap
How would you solve this?

Consider the following program:

How many days' temperatures? 7
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Average temp = 44.57142857142857
4 days were above average.
What makes the problem hard?

- We need each input value twice
  - ... to compute the average via a cumulative sum
  - ... to count how many were above the average

- What about putting the values into variables?
  - How many variables would we declare?

- Need a way to declare many variables at once.
Arrays

- **array**: An object that stores many values of the same type.
  - **element**: a value in an array
  - **index**: an integer indicating the position of a value in an array
Declaring/initializing an array:
<type>[] <name> = new <type>[<length>];

Example:
int[] numbers = new int[10];

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The length can be any integer expression:
int x = 2 * 3 + 1;
int[] data = new int[x % 5 + 2];
Array auto-initialization

- When arrays are initially constructed, every element is automatically initialized to a "zero-equivalent" value.
  - int: 0
  - double: 0.0
  - boolean: false
  - object type: null (null means "no object")
Array auto-initialization: Example

- An array of doubles

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

- An array of booleans

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
Assigning array elements

- Assigning a value to an array element:
  
  `<array name>[<index>] = <value>;`

- Example:
  
  ```
  numbers[0] = 27;
  numbers[3] = -6;
  ```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>-6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Accessing array elements

- Using an array element's value in an expression:
  `<array name>[<index>]`

- Example:

  ```java
  System.out.println(numbers[0]);
  if (numbers[3] < 0) {
      System.out.println("Element 3 is negative.");
  }
  ```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>-6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Don't go out of bounds!

- Reading or writing any index outside the valid range will throw an `ArrayIndexOutOfBoundsException`.

Example:

```java
int[] data = new int[10];
System.out.println(data[0]);       // okay
System.out.println(data[-1]);      // exception!
System.out.println(data[9]);       // okay
System.out.println(data[10]);      // exception!
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example

```java
int[] numbers = new int[8];
numbers[1] = 4;
numbers[4] = 99;
numbers[7] = 2;

int x = numbers[1];  x: [4]
numbers[x] = 44;
numbers[numbers[7]] = 11;  // use numbers[7] as index!
```

```
numbers:  0  4  11  0  44  0  0  2
```
Arrays and for loops

- Arrays are very commonly used with for loops to access each element

- Example:
  ```java
  for (int i = 0; i < 8; i++) {
      System.out.print(numbers[i] + " ");
  }
  System.out.println();  // end the line of output
  ```

Output:

```
0 4 11 0 44 0 0 2
```
Arrays and for loops

```java
for (int i = 0; i < 8; i++) {
    numbers[i] = 2 * i;
}
```

- What’s in the array?

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>
Arrays and for loops

for (int i = 0; i < 8; i++) {
    numbers[i] = i * i;
}

- What’s in the array?

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>25</td>
<td>36</td>
<td>49</td>
</tr>
</tbody>
</table>
The length field

- An array's length field stores its number of elements.

- General syntax:
  
  `<array name>.length`

- NB: Because it's a field (i.e. not a method), it does not use parentheses like a String's .length()!
Example

```java
for (int i = 0; i < numbers.length; i++) {
    System.out.print(numbers[i] + " ");
}
```

**Output:**

```
0 1 4 9 16 25 36 49
```

- What expression refers to the last element of an array? The middle element?
How it all started...

- Solve the following problem:

How many days' temperatures? 7

Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53

Average temp = 44.57142857142857

4 days were above average.
// This program reads several days' temperatures from the user
// and computes the average and how many days were above average.
import java.util.*;

class Weather {  
    public static void main(String[] args) {  
        Scanner console = new Scanner(System.in);  
        System.out.print("How many days' temperatures? ");  
        int days = console.nextInt();  

        int[] temperatures = new int[days];   // array to store days' temperatures
        int sum = 0;

        for (int i = 0; i < days; i++) {  // read/store each day's temperature
            System.out.print("Day "+(i+1)+"'s high temp: ");
            temperatures[i] = console.nextInt();
            sum += temperatures[i];
        }

        double average = (double) sum / days;

        int count = 0;                  // see if each day is above average
        for (int i = 0; i < days; i++) { 
            if (temperatures[i] > average) { 
                count++;
            }
        }

        // report results
        System.out.println("Average temp = "+average);
        System.out.println(count+" days above average");
    }
}
Arrays for counting / tallying
A multi-counter problem

Problem: Examine a number and count the number of occurrences of every digit.

Example: The number 229231007 contains: two 0s, one 1, three 2s, one 7, and one 9

Solution?

Declare 10 counter variables—one per digit. Eewwww!!!!

```c
int counter0, counter1, counter2, counter3;
int counter4, counter5, counter6, counter7;
int counter8, counter9;
```
A multi-counter problem

Problem: Examine a number and count the number of occurrences of every digit.
- Example: The number 229231007 contains: two 0s, one 1, three 2s, one 7, and one 9

Solution:
- Declare an array of 10 elements—the element at index \(i\) will store the counter for digit value \(i\).

```java
int[] counts = new int[10];
```
An array of counters

```java
int num = 229231007;
int[] counts = new int[10];
while (num > 0) {
    int digit = num % 10;
    counts[digit]++;
    num = num / 10;
}
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Histogram: Exercise

- If a user enters a set of integer exam scores, such as:

  Enter an exam score (-1 to exit): 82
  Enter an exam score (-1 to exit): 66
  Enter an exam score (-1 to exit): 79
  Enter an exam score (-1 to exit): 63
  Enter an exam score (-1 to exit): -1

  Write a program that will print a histogram of stars indicating the number of students who earned each unique exam score.

  85: *****
  86: ************
  87: ***
  88: *
  91: ****


// Reads a set of test scores (integers) and displays a
// histogram of the score distribution.
import java.util.*;

public class Histogram {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int[] counts = new int[101]; // counters of test scores 0 - 100

        do {
            System.out.print("Enter an exam score (-1 to exit): ");
            int score = input.nextInt();
            if(score >= 0) {
                counts[score]++;
            }
        } while(score >= 0);

        for (int i = 0; i < counts.length; i++) {  // print star histogram
            if (counts[i] > 0) {
                System.out.print(i + ": ");
                for (int j = 0; j < counts[i]; j++) {
                    System.out.print("*");
                }
            }
        }
    }
}
Why are arrays useful

- Arrays store a large amount of data accessible from one variable.

- Arrays let us access data in random order.
  - Cassette tape vs. DVD

- Arrays can represent sequential data.
  - An array of quiz scores can store not just the scores, but also the order in which the quizzes were taken.
Array initialization statement

- Quick array initialization, general syntax:
  
  `<type>[] <name> = { <value>, <value>, ..., <value> } ;`

- Example:
  
  ```
  int[] numbers = { 12, 49, -2, 26, 5, 17, -6 } ;
  ```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>12</td>
<td>49</td>
<td>-2</td>
<td>26</td>
<td>5</td>
<td>17</td>
<td>-6</td>
</tr>
</tbody>
</table>

- Useful when you know in advance what the array's element values will be.
Example

```java
int[] a = { 2, 5, 1, 6, 14, 7, 9 };
for (int i = 1; i < a.length; i++) {
    a[i] += a[i - 1];
}
```

- What’s in the array?

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>16</td>
<td>28</td>
<td>35</td>
<td>44</td>
</tr>
</tbody>
</table>
Printing arrays: Arrays.toString

- **Arrays.toString** accepts an array as a parameter and returns the **String** representation, which you can then print.

**Example:**

```java
int[] a = { 2, 5, 1, 6, 14, 7, 9 };
for (int i = 1; i < a.length; i++) {
    a[i] += a[i - 1];
}
System.out.println("a is " + Arrays.toString(a));
```

**Output:**

```
a is [2, 7, 8, 14, 28, 35, 44]
```