Lecture 5: Arrays

A way to organize data

MIT AITI

April 9th, 2005
What are Arrays?

• An array is a series of compartments to store data.

• Essentially a block of variables.

• In Java, arrays can only hold one type.

• For example, int arrays can hold only integers and char arrays can only hold characters.
Array Visualization and Terms

• Arrays have a type, name, and size.
• Array of three integers named \texttt{prices}:
  \begin{itemize}
    \item \texttt{prices}:
      \begin{tabular}{|l|l|l|}
        \hline
        \texttt{int} & \texttt{int} & \texttt{int} \\
        \hline
      \end{tabular}
  \end{itemize}

• Array of four Strings named \texttt{people}:
  \begin{itemize}
    \item \texttt{people}:
      \begin{tabular}{|l|l|l|l|}
        \hline
        \texttt{String} & \texttt{String} & \texttt{String} & \texttt{String} \\
        \hline
      \end{tabular}
  \end{itemize}

• We refer to each item in an array as an \textit{element}.
• The position of each element is known as its \textit{index}.
Declaring an Array

• Array declarations similar to variables, but use square brackets:
  - `datatype[] name;`

• For example:
  - `int[] prices;`
  - `String[] people;`

• Can alternatively use the form:
  - `datatype name[];`
  - `int prices[];`
Allocating an Array

• Unlike variables, we need to allocate memory to store arrays. (malloc() in C.)
• Use the new keyword to allocate memory:
  - name = new type[size];
  - prices = new int[3];
  - people = new String[4];
• This allocates an integer array of size 20 and a String array of size 10.
• Can combine declaration and allocation:
  - int[] prices = new int[3];
Array Indices

• Every element in an array is referenced by its index.

• In Java, the index starts at 0 and ends at $n-1$, where $n$ is the size of the array.

• If the array `prices` has size 3, its valid indices are 0, 1, and 2.

• Beware “Array out of Bounds” errors.
Using an Array

• We access an element of an array using square brackets \([\text{index}]\):
  - name[index]

• Treat array elements just like a variable.

• Example assigning values to each element of prices:
  - prices[0] = 6;
  - prices[1] = 80;
  - prices[2] = 10;
Using an Array

• We assign values to elements of String arrays in a similar fashion:
  
  - `String[] people;`
  - `people = new String[4];`
  - `people[0] = "Alice";`
  - `people[1] = "Bilha";`
  - `people[2] = "Chris";`
  - `people[3] = "David";`
Initializing Arrays

• You can also specify all of the items in an array at its creation.
• Use curly brackets to surround the array’s data and separate the values with commas:
  - String[] people = {“Alice”, “Bilha”, “Chris”, “David”};
  - int[] prices = {6, 80, 10};
• All the items must be of the same type.
• Note: Curly brackets are overloaded because they also designate blocks of code.
Vocabulary Review

• Allocate - Create empty space that will contain the array.
• Initialize - Fill in a newly allocated array with initial values.
• Element - An item in the array.
• Index - Element’s position in the array.
• Size or Length - Number of elements.
Pop Quiz

Which of the following sequences of statements does not create a new array?

a) int[] arr = new int[4];

b) int[] arr;
   arr = new int[4];

c) int[] arr = { 1, 2, 3, 4};

d) int[] arr;
Lengths of Array

• Each array has a default field called `length`.
• Access an array’s `length` using the format: 
  - `arrayName.length`;
• Example:
  - `String[] people = {“Alice”, “Bilha”, “Chris”, “David”};`
  - `int numPeople = people.length;`
• The value of `numPeople` is now 4.
• Arrays are always of the same size. Their lengths cannot be changed once they are created.
Example 1

• Sample Code:

```java
String[] people = {"Alice", "Bilha", "Chris", "David"};
for(int i=0; i<names.length; i++)
    System.out.println(names[i]+"!");
```

• Output:

- Alice!
- Bilha!
- Chris!
- David!
Pop Quiz 2

• Given this code fragment:
  - int[] data = new int[10];
  - System.out.println(data[j]);

• Which are legal values of \( j \)?
  a) -1
  b) 0
  c) 3.5
  d) 10
Pop Quiz 3

• Decide what type and size of array (if any) to store each data set:
  – Score in each quarter of a football game.
    ```java
    int[] quarterScore = new int[4];
    ```
  – Your name, date of birth, and height.
    Not appropriate. Different types.
  – Hourly temperature readings for a week.
    ```java
    double[] tempReadings = new double[168];
    ```
  – Your daily expenses for a year.
    ```java
    float[] dailyExpenses = new float[365];
    ```
Exercise 2

• What are the contents of \( c \) after the following code segment?

```java
int [] a = {1, 2, 3, 4, 5};
int [] b = {11, 12, 13};
int [] c = new int[4];
for (int j = 0; j < 3; j++) {
    c[j] = a[j] + b[j];
}
```
2-Dimensional Arrays

• The arrays we've used so far can be thought of as a single row of values.
• A 2-dimensional array can be thought of as a grid (or matrix) of values.
• Each element of the 2-D array is accessed by providing two indices: a row index and a column index.
• A 2-D array is actually just an array of arrays.

value at row index 2, column index 0 is 3

0 1
0 8 4
1 9 7
2 3 6
2-D Array Example

• Example: A landscape grid of a 20 x 55 acre piece of land. We want to store the height of the land at each row and each column of the grid.

• We declare a 2-D array two sets of square brackets:
  - double[][] heights;
  - heights = new double[20][55];

• This 2-D array has 20 rows and 55 columns

• To access the acre at row index 11 and column index 23: heights[11][23]
More on Dimensionality

• Can have unequal sized sub-arrays:
  
  ```java
  int[][] a = new int[3][];
  int[] b = {1,2,3};
  int[] c = {4,5,6,7};
  int[] d = {8};
  a[0]= b; a[1] = c; a[2] = d;
  ```

• Can have higher dimensions:
  
  ```java
  int[][][][][] a; // 4-D Array
  ```