Two-Dimensional Arrays

Arrays that we have consider up to now are one-dimensional arrays, a single line of elements.

Often data come naturally in the form of a table, e.g., spreadsheet, which need a two-dimensional array.

Examples:
- Lab book of multiple readings over several days
- Periodic table
- Movie ratings by multiple reviewers.
  - Each row is a different reviewer
  - Each column is a different movie

Two-Dimensional Arrays

- Two-dimensional (2D) arrays are indexed by two subscripts, one for the row and one for the column.
- Example:

```
rating[0][2] = 2
rating[1][3] = 8
```

Similarity with 1D Arrays

- Each element in the 2D array must by the same type, either a primitive type or object type.
- Subscripted variables can be use just like a variable:

```
rating[0][3] = 10;
```

- Array indices must be of type `int` and can be a literal, variable, or expression.

```
rating[3][j] = j;
```

- If an array element does not exists, the Java runtime system will give you an `ArrayIndexOutOfBoundsException`
Declaring 2D Arrays

• Declare a local variable `rating` that references a 2D array of int:
  ```java
  int[][] rating;
  ```
• Declare a field `family` that references a 2D array of `GiftCard`
  ```java
  private GiftCard[][] family;
  ```
• Create a 2D array with 3 rows and 4 columns and assign the reference to the new array to `rating`:
  ```java
  rating = new int[3][4];
  ```
• Shortcut to declare and create a 2D array:
  ```java
  int[][] rating = new int[3][4];
  ```

Example 1

• Find the average rating by the reviewer in row 2.
  ```java
  int sum = 0;
  for (int col = 0; col <= 3; col++) {
    sum += rating[2][col];
  }
  double average = (double) sum / 4;
  ```

Size of 2D Arrays

• When you write a method that has a 2D array as a parameter, how do you determine the size of the array?

  **Hint:**
  • Consider a variable `words`, a 1D array of `String` references.
  • What is the length of the array?
  • What is the length of the word at index 2?

2D Array Implementation

• A 2D array is a 1D array of (references to) 1D arrays.
  ```java
  int[][] rating = new int[3][4];
  ```
Size of 2D Arrays

- Given
  ```java
  int[][] rating = new int[3][4];
  ```
- What is the value of `rating.length`?
  Answer: 3, the number of rows (first dimension)
- What is the value of `rating[0].length`?
  Answer: 4, the number of columns (second dimension)

Example 2

- Find the number of ratings above the value of the parameter.
  ```java
  public int countAbove(int[][] rating, int num) {
      int count = 0;
      for (int row = 0; row < rating.length; row++) {
          for (int col = 0; col < rating[0].length; col++) {
              if (rating[row][col] > num) {
                  count++;
              }
          }
      }
      return count;
  }
  ```

Example 3

- Print the average rating for the movie in column 3.
  ```java
  int sum = 0;
  for (int row = 0; row < rating.length; row++) {
      sum += rating[row][3];
  }
  System.out.println((double) sum / rating.length);
  ```

Ragged Arrays

- Since a 2D array is a 1D array of references to 1D arrays, each of these latter 1D arrays (rows) can have a different length.
- How? Use an *initializer list.*
  ```java
  int[][] rating = { {3,5,7,9}, {4,2}, {5,7,8,6}, {6} };
  ```
Example 3 Revisited

- Print the average rating for the movie in column 3.

```javaint count = 0;
double sum = 0;
for (int row = 0; row < rating.length; row++) {
    if (rating[row].length > 3) {
        sum += rating[row][3];
        count++;
    }
}
if (count > 0) {
    System.out.println((double) sum / count);
}
```

2D Array of Object References

- Recall that creating an array of object references fills the array with `null` values.

Example:

```java
GiftCard[][] family = new GiftCard[3][4]
```

Example 4

- Print the total value of the gift cards for each family member (rows):

```java
public static void printValueOfRows(GiftCard[][] data) {
    for (int row = 0; row < data.length; row++) {
        double total = 0.0;  // find total for the row
        for (int col = 0; col < data[row].length; col++) {
            if (data[row][col] != null) {
                total += data[row][col].getBalance();
            }
        }
        System.out.println("Row " + row + ": $" + total);
    }
}
```