16 Arrays (3)

Instructor: Ke Wei（柯韋）

A319 ✆ Ext. 6452 ✉ wke@ipm.edu.mo

http://brouwer.ipm.edu.mo/COMP112/18/

Bachelor of Science in Computing, School of Public Administration, Macao Polytechnic Institute

November 9, 2018

Outline

Figuring out the Relation between Adjacent Elements

- A very important technique in programming is to use the values previously stored in variables.
- We can create an array containing

$$1,\ 1+2,\ 1+2+3,\ 1+2+3+4,\ \ldots,\ \sum_{j=1}^{n} j \qquad (n \geq 1)$$

by

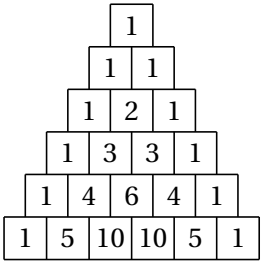
```
int[] a = new int[n];
for ( int i = 0; i < a.length; ++i ) a[i] = (i+1)*(i+2)/2;
```

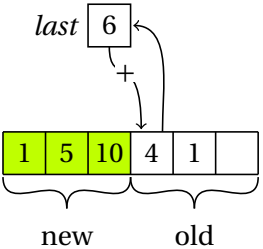
or, by using the value stored in the previous element

```
int[] a = new int[n];
a[0] = 1;
for ( int i = 1; i < a.length; ++i ) a[i] = a[i-1]+(i+1);
```

Binomial Coefficients

- The coefficients of $(x+1)^n$ can be computed by the Pascal's triangle.
- We use a single array, and overwrite the old row with the new row.





Pascal's Triangle — Code

Given $n \geq 0$, we print the Pascal's triangle for $(x+1)^n$ row by row. The format of a full-width item is specified in `FW`, and half-width in `HW`.

```
1  final String FW = "%4d", HW = "%2s";
2  int[] a = new int[n+1];
3
4  for ( int i = 0; i < n+1; ++i ) {
5      for ( int j = n; j > i; --j )
6          System.out.printf(HW, "");
7      int last = 0;
8      for ( int j = 0; j < i; ++j ) {
9          int t = a[j];
10         a[j] += last;
11         last = t;
12         System.out.printf(FW, a[j]);
13     }
14     a[i] = 1;
15     System.out.printf(FW, a[i]);
16     System.out.println();
17 }
```

Initializing Elements by Random Numbers

- We have used *System.currentTimeMillis()* to get random numbers. It is OK with user interactions, since user interactions are slow.
- If we want to initialize 100 array elements, the loop completes within one millisecond, the above method no longer works.
- We need a *random number generator*. Java provides one as the *Random* class.

```
import java.util.Random;
```

- We need to create an object *ran* of *Random* to use the generator, just like with *Scanner*.
- The *nextInt(1000)* and *nextDouble()* methods generate the next random integer between 0 and 1000, and the next random `double` between 0.0 and 1.0.

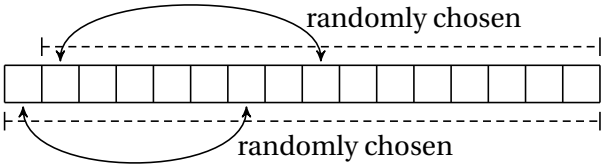
```
Random ran = new Random();
int[] a = new int[50];
for ( int i = 0; i < a.length; ++i )
    a[i] = ran.nextInt(1000);
```

Shuffling Playing Cards

- In games, we often need to shuffle a deck of playing cards.
- To simulate this process, we rearrange the cards in an array randomly.
- The cards are encoded as integers from 0 to 51, with the ranks from 0 to 12, and suits from 0 to 3. A card c has the rank $r = c/4$ and the suit $s = c \% 4$.
- We store the rank names and suit names as arrays of `char`.

```
char[] ranks = {'2','3','4','5','6','7','8','9','T','J','Q','K','A'};
char[] suits = {'C','D','H','S'};
```

- We 1) initialize the deck orderly, 2) randomly choose a card from the deck and place it at index 0, 3) and randomly choose a card from the remaining deck and place it at index 1, and so on.



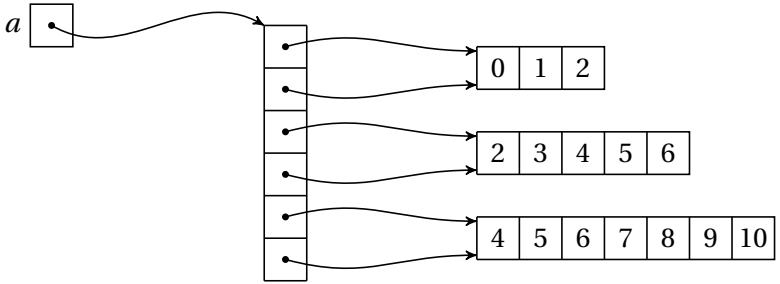
Shuffling Playing Cards — Code

Since the random number generator chooses an integer from 0 to some upper bound, to be convenient for deciding the upper bound of the random positions, we fix the number in the last position first, and go downwards.

```
1  int[] a = new int[52];
2  for ( int i = 0; i < a.length; ++i )
3      a[i] = i;
4  for ( int i = a.length; i > 1; ) {
5      int j = ran.nextInt(i); // a random position between 0 and i−1
6      int t = a[j]; // swap a[i−1] and a[j]
7      a[j] = a[--i];
8      a[i] = t;
9  }
10 for ( int i = 0; i < a.length; ++i )
11     System.out.println(ranks[a[i]/4]+" "+suits[a[i]%4]);
```

Arrays of References

- If the element type is a primitive type, the elements store values.
- If the element type is a class type, the elements store references to objects.
- We need to point the references to some existing objects, or set the references to `null`.
- The array variable, the array of references, and the objects pointed to by the references are separately declared and created.
- Multiple array elements can point to the same object.



Example: an Array of Arrays

- First, we declare the array variable *a*, and create the array of references to `int[]`.

```
int[][] a = new int[6][];
```

- Second, we create some `int[]` (array) objects, and assign them to some outer array elements. We then initialize the inner array elements.

```
for ( int i = 0; i < a.length; i += 2 ) {
    a[i] = new int[3+i];
    for ( int j = 0; j < a[i].length; ++j )
        a[i][j] = i+j;
}
```

- Finally, we share the array objects with the neighboring elements.

```
for ( int i = 1; i < a.length; i += 2 )
    a[i] = a[i-1];
```

Creating a Matrix

- In mathematics, a matrix is a rectangular array of numbers, arranged in rows and columns. An example of a matrix with 3 rows and 5 columns is

$$\begin{pmatrix} 2 & 5 & 11 & 17 \\ 3 & 7 & 19 & 23 \\ 13 & 29 & 31 & 37 \end{pmatrix}$$

- The following method creates the objects for a matrix, to be used as a two-dimensional array, for example `a[1][2] = 19`. The method does not initialize the elements.

```
1 public static double[][] createDoubleMatrix(int rows, int cols) {
2     double[][] a = new double[rows][];
3     for ( int i = 0; i < a.length; ++i ) a[i] = new double[cols];
4     return a;
5 }
```

- Note: `new double[rows][cols]` is equivalent to `createDoubleMatrix(rows, cols)`.

Reading Homework

Textbook

- Section 3.7
- Section 4.2.5
- Section 8.1–8.4, 8.7
- Section 9.6.2

Internet

- Pseudorandom number generator (`https://en.wikipedia.org/wiki/Pseudorandom_number_generator`).
- Two-dimensional Arrays (`https://math.hws.edu/javanotes/c7/s5.html`).