Java Arrays

Algoritmos y estructuras de datos (AyEdD)

Jesús Parrado Alameda

24 de Febrero de 2022

Casos prácticos y ejemplos para la utilización de Arrays en Java.

URLs de Referencia

Algorithms - Sección Arrays

`https://algs4.cs.princeton.edu/11model/`.

Computer Science - Sección Arrays

`https://introcs.cs.princeton.edu/java/14array/`

Repo con ejemplos

`https://github.com/jesusgpa/java-algorithms-and-clients`

Arrays

Un array almacena una secuencia de valores, todos ellos del mismo tipo.

Si tenemos `N` valores se puede usar la notación `a[i]` para referirnos al valor que ocupa la posición `i` para cualquier valor de `i` comprendido entre `0` y `N-1`.

Construcción de arrays. Hay tres pasos en la construcción de un array:

Declaración del identificador y el tipo de los valores que almacenará el array

Creación del espacio en memoria que ocuparán los valores del array

Inicialización de los valores del array. Hay tres formas de hacerlo:

1. En este caso, declaramos la referencia a un array, luego se reserva memoria y con `for` se dan valores:

```
final int N = 10; // N es una constante
double[] a;       // a es una referencia

a = new double[N]; // Reservamos memoria para el array
for (int i = 0; i < N; i++)
    a[i] = 0.0;
```

2. En este caso declaramos y creamos memoria en la misma sentencia:

```
double[] a = new double[N]; // declaramos y creamos
                            // memoria para el array
```

3. Aquí, sin declarar el tamaño pero sí los valores en la inicialización. Se crea memoria implícitamente y el tamaño es el de los valores entre llaves:

```
int[] a = { 1, 1, 2, 3, 5, 8 };
```

Uso de un array. Una vez que se ha creado un array, su tamaño queda fijado.

La longitud de un array `a[]` se obtiene con `a.length`.

Java realiza comprobación automática de los límites: si se intenta acceder a una posición de un array con un índice que no existe se eleva la excepción `ArrayIndexOutOfBoundsException`[1]

[1] Una excepción es un error que ocurre en tiempo de ejecución. Se gestionar o tratar las excepciones para que el programa no termine cuando se produce una.

Aliasing. El identificador de una variable que contiene un array referencia a todo el array: si se asigna una variable que contiene un array a otra, lo que se copia es la referencia, por lo que ambas variables hacen referencia a un único array en memoria.

Veamos un ejemplo:

```
int[] a = new int[N];
...
a[i] = 1234;
...
int[] b = a;
...
b[i] = 5678;
```

La última asignación a `b[i]` hace que también `a[i]` sea 5678 pues `a` y `b` referencian el mismo array.

Esta situación se conoce como aliasing[2] y puede propiciar errores difíciles de detectar.

[2] El array `b` es un alias del array `a`.

Arrays de dos dimensiones. Un array de dos dimensiones en Java es un array de una dimensión cuyos elementos son arrays de una dimensión.

Cada uno de los arrays puede ser de tamaños diferentes si bien es frecuente que sean del mismo tamaño.

Para referirse a un elemento en la fila `i` y la columna `j` de un array de dos dimensiones se utiliza la notación `a[i][j]`.

ArrayExamples.java

En esta clase puedes encontrar algunos ejemplos de operaciones habituales con arrays en Java.

https://introcs.cs.princeton.edu/java/14array/ArrayExamples.java.html

```
/*************************************************
 *  Compilation:  javac ArrayExamples.java
 *  Execution:    java ArrayExamples
 *
 *  Typical array processing code.
 *
 *  % java ArrayExamples 5
 *  a[]
 *  -------------------
 *  0.04851944273872377
 *  0.8311550808965679
 *  0.5288965750549762
 *  0.5053593215147596
 *  0.6162362251917868
 *
 *  a = [D@f62373
 *
 *  max = 0.8311550808965679
 *  average = 0.5060333290793628
 *
 *  b[]
 *  -------------------
 *  0.6162362251917868
 *  0.5053593215147596
 *  0.5288965750549762
 *  0.8311550808965679
 *  0.04851944273872377
 *
 *  dot product of a[] and b[] = 1.1795943990991937
 *
 *************************************************/

public class ArrayExamples {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);

        // initialize to random values between 0 and 1
        double[] a = new double[n];
        for (int i = 0; i < n; i++) {
            a[i] = Math.random();
        }
```

```
        // print array values, one per line
        System.out.println("a[]");
        System.out.println("-------------------");
        for (int i = 0; i < n; i++) {
            System.out.println(a[i]);
        }
        System.out.println();
        System.out.println("a = " + a);
        System.out.println();

        // find the maximum
        double max = Double.NEGATIVE_INFINITY;
        for (int i = 0; i < n; i++) {
            if (a[i] > max) max = a[i];
        }
        System.out.println("max = " + max);


        // average
        double sum = 0.0;
        for (int i = 0; i < n; i++) {
            sum += a[i];
        }
        System.out.println("average = " + sum / n);

        // copy to another array
        double[] b = new double[n];
        for (int i = 0; i < n; i++) {
            b[i] = a[i];
        }

        // reverse the order
        for (int i = 0; i < n/2; i++) {
            double temp = b[i];
            b[i] = b[n-i-1];
            b[n-i-1] = temp;
        }

        // print array values, one per line
        System.out.println();
        System.out.println("b[]");
        System.out.println("-------------------");
        for (int i = 0; i < n; i++) {
            System.out.println(b[i]);
        }
        System.out.println();
```

```
        // dot product of a[] and b[]
        double dotProduct = 0.0;
        for (int i = 0; i < n; i++) {
            dotProduct += a[i] * b[i];
        }
        System.out.println("dot product of a[] and b[] = " + dotProduct);

    }

}
```

Deck.java

Esta clase simula como barajar cartas.

https://introcs.cs.princeton.edu/java/14array/Deck.java.html

```
/*************************************************************************
 *  Compilation:  javac Deck.java
 *  Execution:    java Deck
 *
 *  Deal 52 cards uniformly at random.
 *
 *  % java Deck
 *  Ace of Clubs
 *  8 of Diamonds
 *  5 of Diamonds
 *  ...
 *  8 of Hearts
 *
 *************************************************************************/

public class Deck {
    public static void main(String[] args) {
        String[] SUITS = {
            "Clubs", "Diamonds", "Hearts", "Spades"
        };

        String[] RANKS = {
            "2", "3", "4", "5", "6", "7", "8", "9", "10",
            "Jack", "Queen", "King", "Ace"
        };

        // initialize deck
        int n = SUITS.length * RANKS.length;
        String[] deck = new String[n];
        for (int i = 0; i < RANKS.length; i++) {
            for (int j = 0; j < SUITS.length; j++) {
                deck[SUITS.length*i + j] = RANKS[i] + " of " + SUITS[j];
            }
        }

        // shuffle
        for (int i = 0; i < n; i++) {
            int r = i + (int) (Math.random() * (n-i));
            String temp = deck[r];
            deck[r] = deck[i];
            deck[i] = temp;
        }
```

```
        // print shuffled deck
        for (int i = 0; i < n; i++) {
            System.out.println(deck[i]);
        }
    }

}
```

Sample.java

Esta clase produce una muestra aleatoria entre de m elementos entre o y n.

https://introcs.cs.princeton.edu/java/14array/Sample.java.html

```
/******************************************************************************
 *  Compilation:  javac Sample.java
 *  Execution:    java Sample m n
 *
 *  This program takes two command-line arguments m and n and produces
 *  a random sample of m of the integers from 0 to n-1.
 *
 *  % java Sample 6 49
 *  10 20 0 46 40 6
 *
 *  % java Sample 10 1000
 *  656 488 298 534 811 97 813 156 424 109
 *
 ******************************************************************************/

public class Sample {
    public static void main(String[] args) {
        int m = Integer.parseInt(args[0]);    // choose this many elements
        int n = Integer.parseInt(args[1]);    // from 0, 1, ..., n-1

        // create permutation 0, 1, ..., n-1
        int[] perm = new int[n];
        for (int i = 0; i < n; i++)
            perm[i] = i;

        // create random sample in perm[0], perm[1], ..., perm[m-1]
        for (int i = 0; i < m; i++)  {

            // random integer between i and n-1
            int r = i + (int) (Math.random() * (n-i));

            // swap elements at indices i and r
            int t = perm[r];
            perm[r] = perm[i];
            perm[i] = t;
        }

        // print results
        for (int i = 0; i < m; i++)
            System.out.print(perm[i] + " ");
        System.out.println();
    }
```

```
}
```

CouponCollector.java

Si tenemos n cartas diferentes, cuántas cartas al azar tenemos que sacar hasta tener una de cada tipo.

https://introcs.cs.princeton.edu/java/14array/CouponCollector.java.html

```
/*************************************************************************
 *  Compilation:  javac CouponCollector.java
 *  Execution:    java CouponCollector n
 *
 *  Given n distinct card types, how many random cards do you need
 *  do collect before you have (at least) one of each type?
 *  This program simulates this random process.
 *
 *
 *  % java CouponCollector 1000
 *  6583
 *
 *  % java CouponCollector 1000
 *  6477
 *
 *  % java CouponCollector 1000000
 *  12782673
 *
 *************************************************************************/

public class CouponCollector {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);       // number of card types
        boolean[] isCollected = new boolean[n];  // isCollected[i] = true if card i has been collected
        int count = 0;                           // total number of cards collected
        int distinct = 0;                        // number of distinct cards

        // repeatedly choose a random card and check whether it's a new one
        while (distinct < n) {
            int value = (int) (Math.random() * n);   // random card between 0 and n-1
            count++;                                 // we collected one more card
            if (!isCollected[value]) {
                distinct++;
                isCollected[value] = true;
            }
        }

        // print the total number of cards collected
        System.out.println(count);
    }
}
```

Keep Calm and Learn Java Arrays