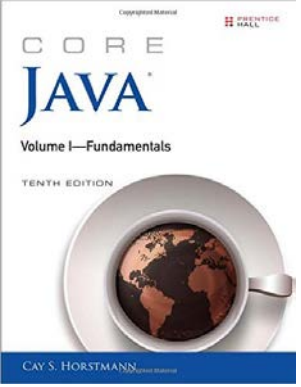


Fundamental Programming Structures in Java – Part3

Chapter 3, Core Java, Volume I

Contents

Arrays

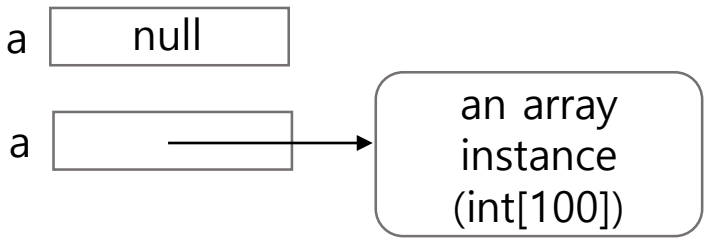
- int[] is an array of integers.
- Array variable declaration:

```
int[] a;
```

- new operator creates array:

```
int[] a = new int[100];
```

Array is a *reference type*!



- Array indexes are from 0 to a.length - 1. (length variable denotes the length of an array)
- Use [ ] to access elements:

```
for (int i = 0; i < a.length; i++)
    System.out.println( a[i] );
```

- Or use the "for-each" loop (*enhanced*-for loop):

```
for (int element : a)
    System.out.println(element);
```

Array Initializer

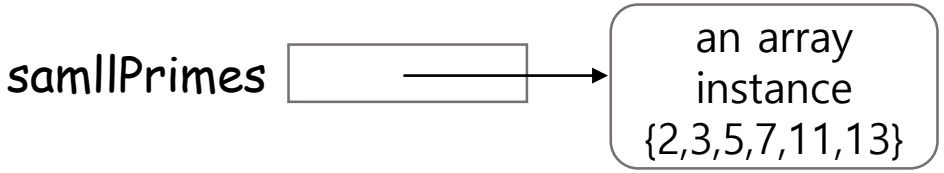
- To create an array object supplying initial values

```
int[] smallPrimes = { 2, 3, 5, 7, 11, 13 };
```

- To use an anonymous array:

```
int[] smallPrimes;
smallPrimes = new int[] { 2, 3, 5, 7, 11, 13 };
```

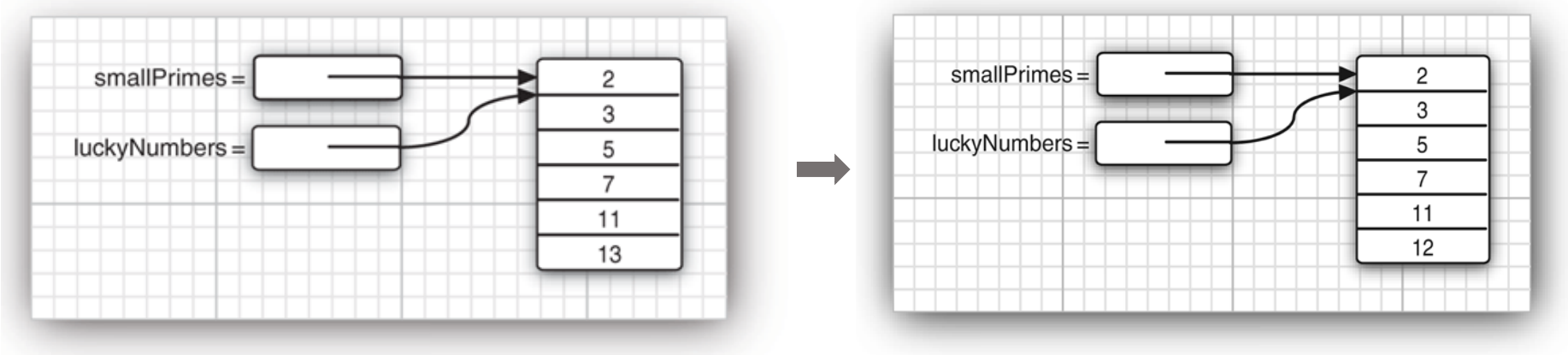
creating and initializing an anonymous array



Copying Arrays

- Copying array variables yields two references to the same array:

```
int[] luckyNumbers = smallPrimes;
luckyNumbers[5] = 12; // now smallPrimes[5] is also 12
```



- Use Arrays.copyOf to make a true copy:

```
int[] copiedLuckyNumbers = Arrays.copyOf(luckyNumbers, luckyNumbers.length);
```

The class Arrays contains a set of useful *static* methods!

Array Sorting

- Use the sort method in the Arrays class

```
int[] a = new int[100];
…
Arrays.sort(a);
```

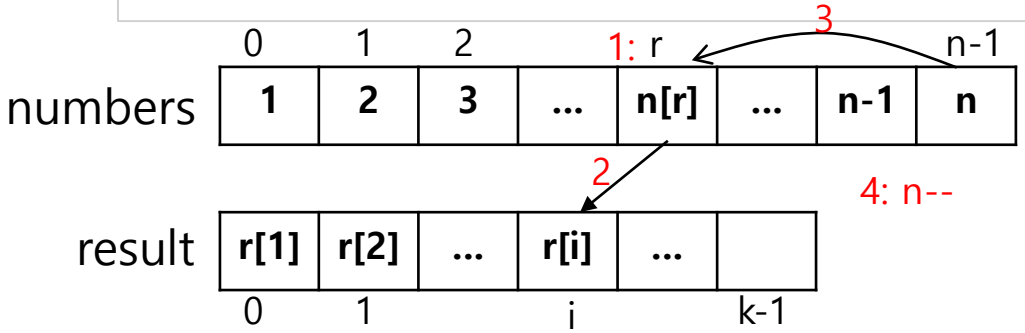
- Arrays class (java.util.Arrays)
  - static void sort(*type*[] a)
  - static int binarySearch(*type*[] a, *type* v)
  - static void fill(*type*[] a, *type* v)
  - static Boolean equals(*type*[] a, *type*[] b)
  - static type[] copyOf(*type*[] a, int length)
  - …

  // *type* : int, long, short, char, byte, float, double

Overloaded methods

Example: LotteryDrawng.java

```java
import java.util.*;

public class LotteryDrawing
{
   public static void main(String[] args)
   {
      Scanner in = new Scanner(System.in);

      System.out.print("How many numbers to draw? ");
      int k = in.nextInt();

      System.out.print("What is the highest number? ");
      int n = in.nextInt();

      // fill an array with numbers 1 2 3 . . . n
      int[] numbers = new int[n];
      for (int i = 0; i < numbers.length; i++)
         numbers[i] = i + 1;
```

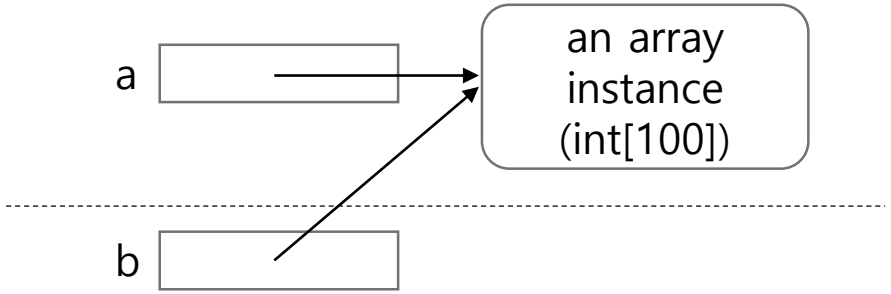



```java
      // draw k numbers and put them into a second array
         int[] result = new int[k];
         for (int i = 0; i < result.length; i++)
         {
            // make a random index between 0 and n - 1
            int r = (int) (Math.random() * n);   // 1

            // pick the element at the random location
            result[i] = numbers[r];  // 2

            // move the last element into the random location
            numbers[r] = numbers[n - 1];  // 3
            n--;  //4
         }

         // print the sorted array
         Arrays.sort(result);
         System.out.println("Bet as follows. It'll make you rich!");
         for (int r : result)
            System.out.println(r);
      }
}
```

Passing Array Parameters

- Reference to an array is passed to the parameter in methods
- Array parameter variables share the variables in the calling methods

```
...
int total = sum( a ); // reference to array a
...
int sum( int[] b )
{
    int total = 0;
    for(int e : a)
        total += e;
    return total;
}
```



- Methods can change the array referred by the variables in the calling methods

```
int[] a = new int[100];
...
Arrays.sort(a);
// passing the reference to array a to the sort method and
// the sort method can change array a in the calling method
```

Example: LotteryDrawng.java (revisiting: array parameters)

```java
import java.util.*;
public class LotteryDrawing
{
  public static void main(String[] args)
  {
    Scanner in = new Scanner(System.in);
    System.out.print("How many numbers to draw? ");
    int k = in.nextInt();

    System.out.print("What is the highest number? ");
    int n = in.nextInt();
    // fill an array with numbers 1 2 3 . . . n
    int[] numbers = new int[n];
    for (int i = 0; i < numbers.length; i++)
      numbers[i] = i + 1;

    int[] result = new int[k];
    draw(numbers, result); // method call using array param

    // print the sorted array
    Arrays.sort(result);
    System.out.println("Bet as follows. It'll make you rich!");
    for (int r : result)
      System.out.println(r);
  }
```

```java
  static void draw(int[] numbers, int[] result)
  {
      // draw k numbers and put them into a second array
      int n = numbers.length;
      for (int i = 0; i < result.length; i++)
      {
        // make a random index between 0 and n - 1
        int r = (int) (Math.random() * n);   // 1

        // pick the element at the random location
        result[i] = numbers[r];  // 2

        // move the last element into the random location
        numbers[r] = numbers[n - 1];  // 3
        n--;  //4
      }
  }
} // end of class
```

Multidimensional Arrays

- int[][] is an array of arrays of int or a two-dimensional array:

```
int[][] magicSquare =
{
  {16, 3, 2, 13},
  {5, 10, 11, 8},
  {9, 6, 7, 12},
  {4, 15, 14, 1}
};
```

- Without initializer:

```
int[][] magicSquare = new int[ROWS][COLUMNS];
```

- Use two indexes to access element: magicSquare[1][2] is 11.
- Use this loop to traverse the elements:

```
for (int i=0; i<ROWS; i++)
  for (int j=0; j<COLUMNS; j++)
    do something with magicSquare[i][j]
```

```
for (int[] row : magicSquare)
  for (int element : row)
    do something with element
```

Example: CompoundInterest.java

```
final double STARTRATE = 10;
final int NRATES = 6;
final int NYEARS = 10;

// set interest rates to 10 . . . 15%
double[] interestRate = new double[NRATES];
for (int j = 0; j < interestRate.length; j++)
   interestRate[j] = (STARTRATE + j) / 100.0;

double[][] balances = new double[NYEARS][NRATES];

// set initial balances to 10000
for (int j = 0; j < balances[0].length; j++)  // 1
   balances[0][j] = 10000;

// compute interest for future years
for (int i = 1; i < balances.length; i++)
{
   for (int j = 0; j < balances[i].length; j++)
   {
      // get last year's balances from previous row
      double oldBalance = balances[i - 1][j]; // 2
      // compute interest
      double interest = oldBalance * interestRate[j];

      // compute this year's balances
      balances[i][j] = oldBalance + interest;  // 3
   }
}
// print one row of interest rates
for (int j = 0; j < interestRate.length; j++)
   System.out.printf("%9.0f%%", 100 * interestRate[j]);

System.out.println();

// print balance table
for (double[] row : balances)
{
   // print table row
   for (double b : row)
      System.out.printf("%10.2f", b);

   System.out.println();
}
```

Example: CompoundInterest.java

interestRate

0	1	2	3	4	5
10	11	12	13	14	15

balances

j = 2

	0	1	2	3	4	5
0	10000	10000	10000	10000	10000	10000
1						
2						
3			b[i-1][j]			
i = 4			b[i][j]			
5						
6						
7						
8						
9						

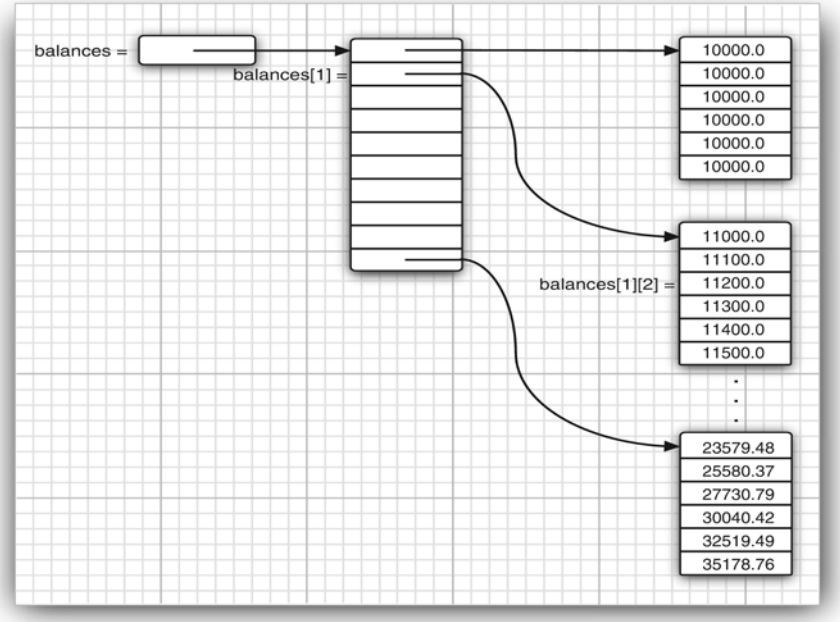
1 : initialize

2: get the old balance

3: calculate the new balance

Ragged Arrays

- Java has no multidimensional arrays at all, only one-dimensional arrays.
- Multidimensional arrays are faked as "arrays of arrays"



- Rows of arrays are individually accessible

```
double[] temp = balances[i];
balance[i] = balances[i+1];
balances[i+1]=temp;
```

- Ragged arrays
  - different rows may have different lengths

```
int[][] odds = new int[7][];
for(int n =0; n<7; n++)
    odds[n] = new int[n+1];
```

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
```

odds[i][j] = C(i, j)

Example: LotteryArray.java

```
public class LotteryArray
{
   public static void main(String[] args)
   {
      final int NMAX = 10;

      // allocate triangular array
      int[][] odds = new int[NMAX + 1][];
      for (int n = 0; n <= NMAX; n++)
         odds[n] = new int[n + 1];

      // print triangular array
      for (int[] row : odds)
      {
         for (int odd : row)
            System.out.printf("%4d", odd);
         System.out.println();
      }
   }
}
```

```
// fill triangular array
for (int n = 0; n < odds.length; n++)
   for (int k = 0; k < odds[n].length; k++)
   {
      /*
       * compute binomial coefficient
            n*(n-1)*(n-2)*...*(n-k+1)/(1*2*3*...*k)
       */
      int lotteryOdds = 1;
      for (int i = 1; i <= k; i++)
         lotteryOdds = lotteryOdds * (n - i + 1) / i;

      odds[n][k] = lotteryOdds;
   }
```

Example: LotteryArray.java (revisiting: static variables)

```
public class LotteryArray
{
   static final int NMAX = 10;
   static int[][] odds;
   public static void main(String[] args)
   {
      initialize();
      makeOdds();
      printOdds();
   }
   static void initialize()
   {
     // allocate triangular array
      odds = new int[NMAX + 1][];
      for (int n = 0; n <= NMAX; n++)
         odds[n] = new int[n + 1];
   }
```

```
   static void makeOdds()
   {
      // fill triangular array
      for (int n = 0; n < odds.length; n++)
         for (int k = 0; k < odds[n].length; k++)
         {
            int lotteryOdds = 1;
            for (int i = 1; i <= k; i++)
               lotteryOdds = lotteryOdds * (n - i + 1) / i;

            odds[n][k] = lotteryOdds;
         }
   }
   static void printOdds()
   {
      // print triangular array
      for (int[] row : odds)
      {
         for (int odd : row)
            System.out.printf("%4d", odd);
         System.out.println();
      }
   }
} // end of class
```

Command-Line Parameters

- Every program has a main method with a String[] args parameter.
- This parameter receives the arguments specified on the command line with an array of strings.

```
public class Message
{
   public static void main( String[] args )
   {
      if (args.length==0 || args[0].equals("-h"))
         System.out.println("Hello");
      else if (args[0].equals("-g"))
         System.out.println("Goodbye");
      for (int i = 1; i< args.length; i++)
         System.out.print(" "+args[i]);
      System.out.println("!");
   }
}
```

```
Java Message -g cruel world

Goodbye, cruel world!
```

```
args[0] : -g
args[1] : cruel
args[2] : world
```