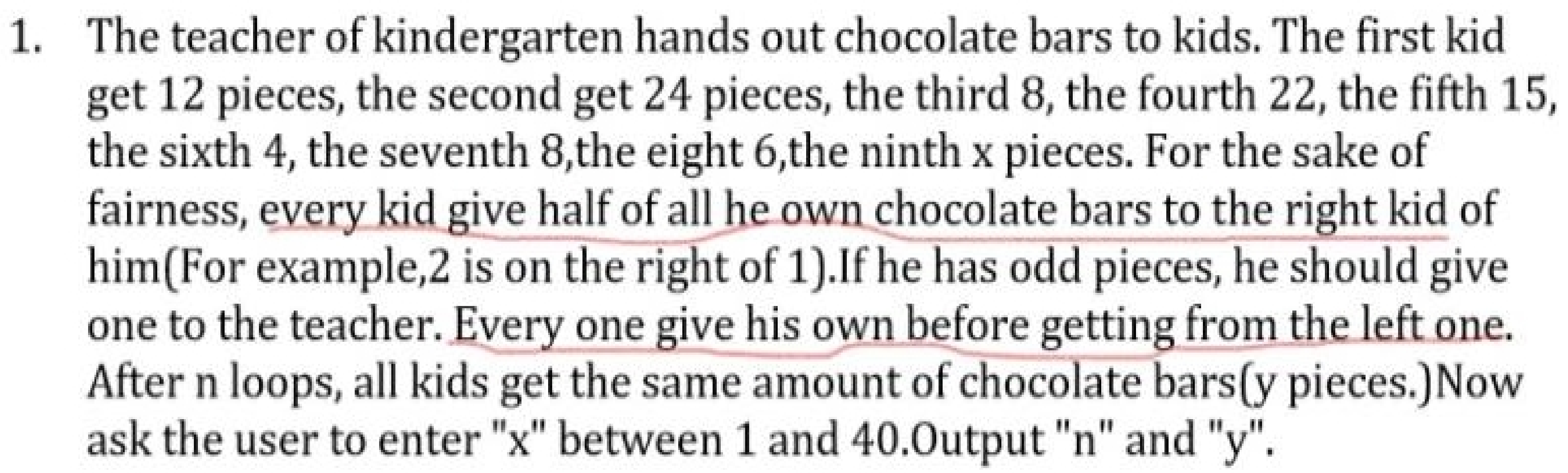
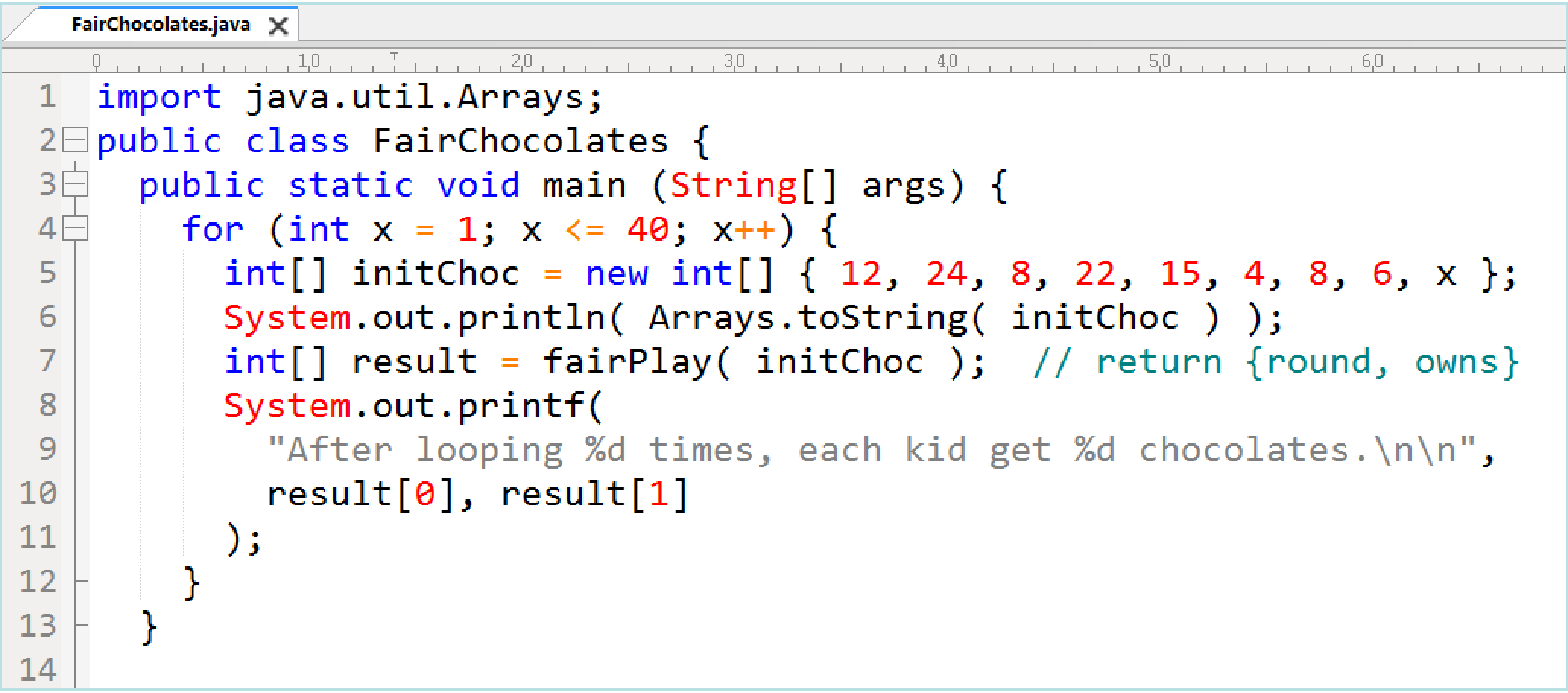
Solutions for Exercises 05

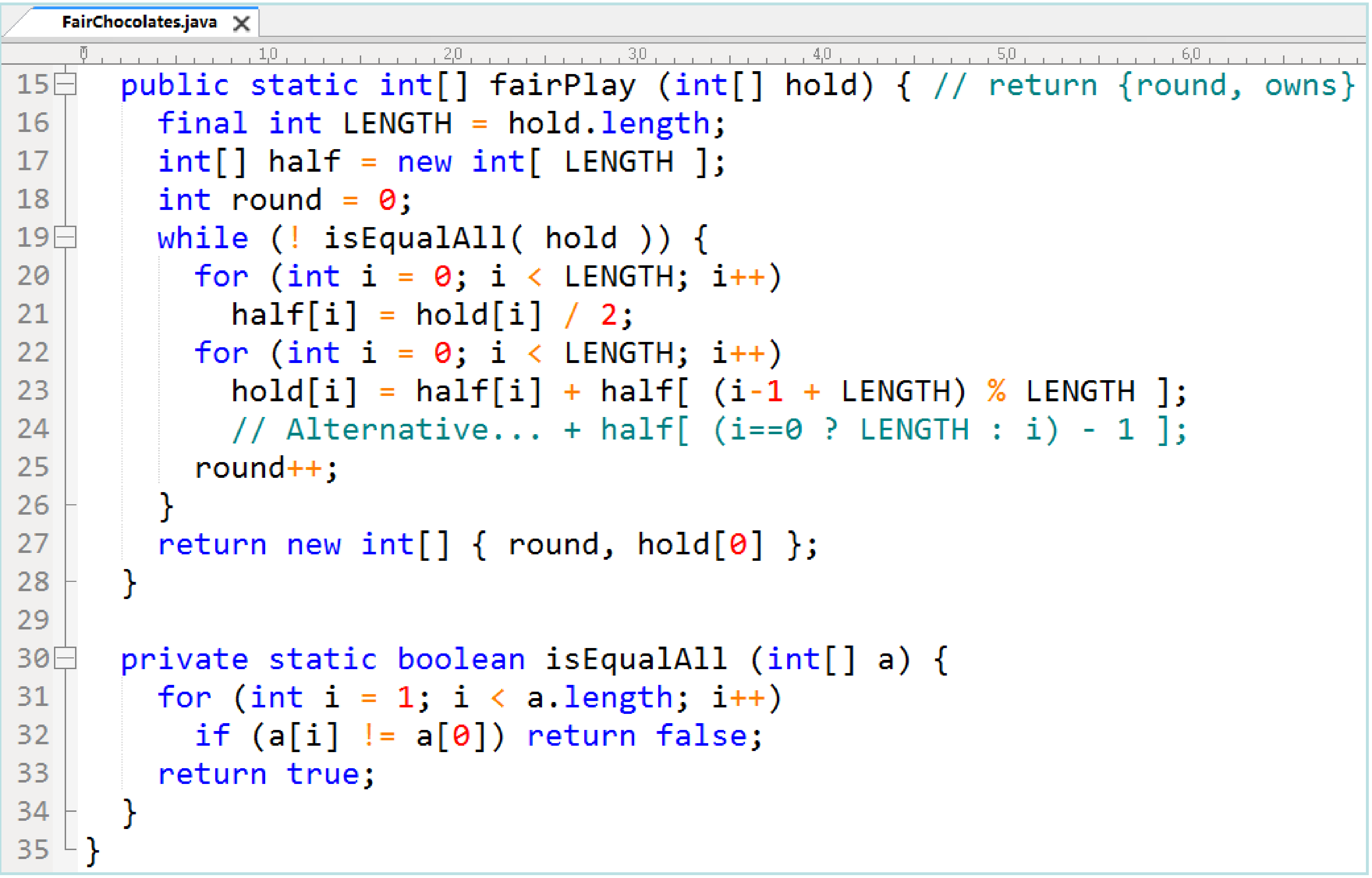


1. The teacher of kindergarten hands out chocolate bars to kids. The first kid get 12 pieces, the second get 24 pieces, the third 8, the fourth 22, the fifth 15, the sixth 4, the seventh 8,the eight 6,the ninth x pieces. For the sake of fairness, every kid give half of all he own chocolate bars to the right kid of him(For example,2 is on the right of 1).If he has odd pieces, he should give one to the teacher. Every one give his own before getting from the left one. After n loops, all kids get the same amount of chocolate bars(y pieces.)Now ask the user to enter "x" between 1 and 40.Output "n" and "y".

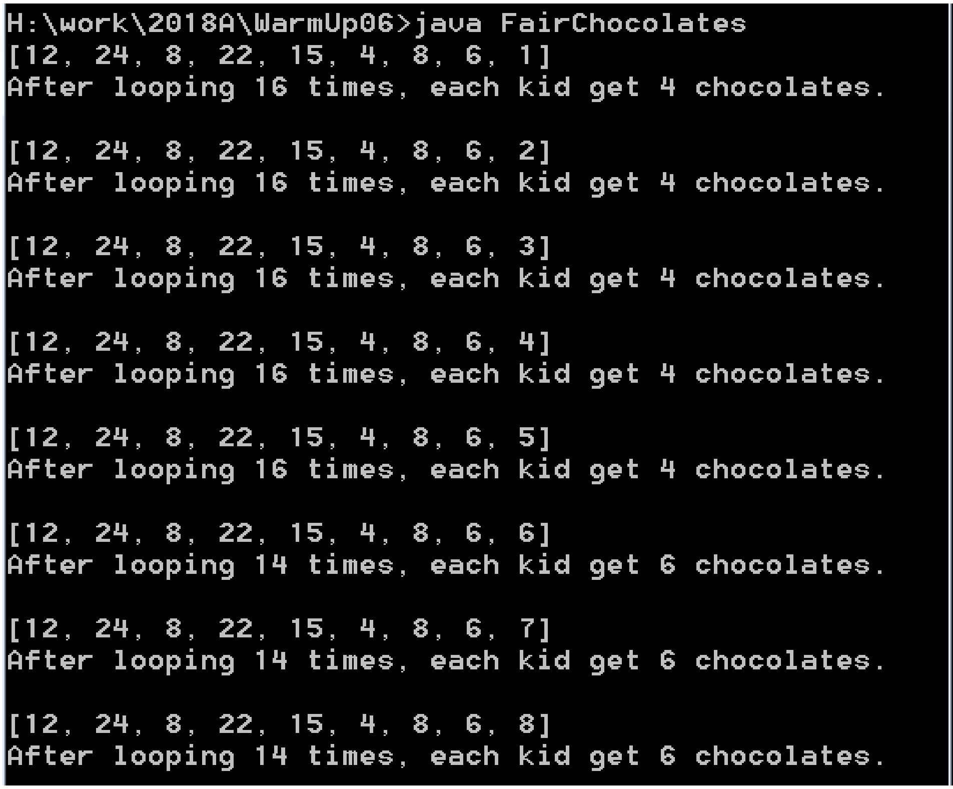
```
Enter x: 13
After looping 14 times, each kid get 6 chocolates.
```


FairChocolates.java

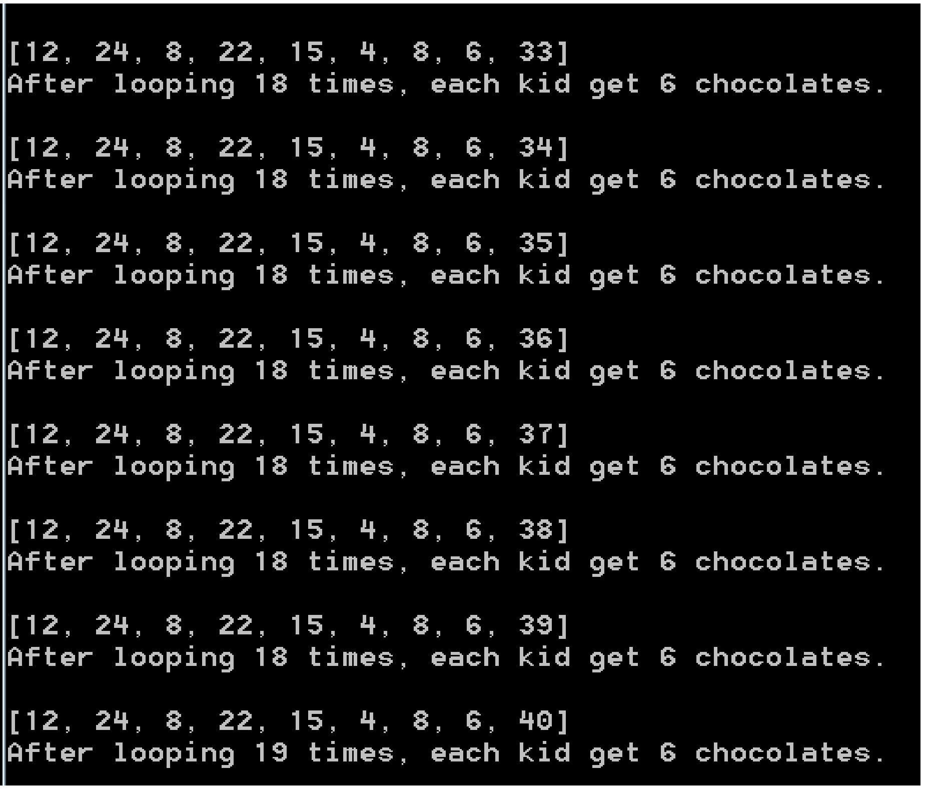
```java
import java.util.Arrays;
public class FairChocolates {
  public static void main (String[] args) {
    for (int x = 1; x <= 40; x++) {
      int[] initChoc = new int[] { 12, 24, 8, 22, 15, 4, 8, 6, x };
      System.out.println( Arrays.toString( initChoc ) );
      int[] result = fairPlay( initChoc );  // return {round, owns}
      System.out.printf(
        "After looping %d times, each kid get %d chocolates.\n\n",
        result[0], result[1]
      );
    }
  }

```


FairChocolates.java

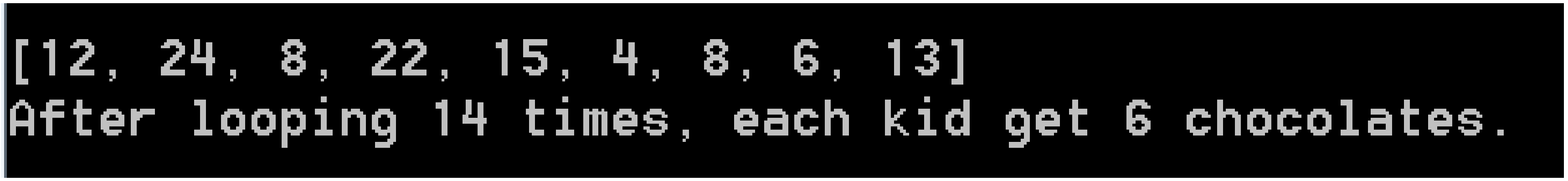
```java
  public static int[] fairPlay (int[] hold) { // return {round, owns}
    final int LENGTH = hold.length;
    int[] half = new int[ LENGTH ];
    int round = 0;
    while (! isEqualAll( hold )) {
      for (int i = 0; i < LENGTH; i++)
        half[i] = hold[i] / 2;
      for (int i = 0; i < LENGTH; i++)
        hold[i] = half[i] + half[ (i-1 + LENGTH) % LENGTH ];
        // Alternative... + half[ (i==0 ? LENGTH : i) - 1 ];
      round++;
    }
    return new int[] { round, hold[0] };
  }

  private static boolean isEqualAll (int[] a) {
    for (int i = 1; i < a.length; i++)
      if (a[i] != a[0]) return false;
    return true;
  }
}
```



```
H:\work\2018A\WarmUp06>java FairChocolates
[12, 24, 8, 22, 15, 4, 8, 6, 1]
After looping 16 times, each kid get 4 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 2]
After looping 16 times, each kid get 4 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 3]
After looping 16 times, each kid get 4 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 4]
After looping 16 times, each kid get 4 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 5]
After looping 16 times, each kid get 4 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 6]
After looping 14 times, each kid get 6 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 7]
After looping 14 times, each kid get 6 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 8]
After looping 14 times, each kid get 6 chocolates.
```



```
[12, 24, 8, 22, 15, 4, 8, 6, 33]
After looping 18 times, each kid get 6 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 34]
After looping 18 times, each kid get 6 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 35]
After looping 18 times, each kid get 6 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 36]
After looping 18 times, each kid get 6 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 37]
After looping 18 times, each kid get 6 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 38]
After looping 18 times, each kid get 6 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 39]
After looping 18 times, each kid get 6 chocolates.

[12, 24, 8, 22, 15, 4, 8, 6, 40]
After looping 19 times, each kid get 6 chocolates.
```



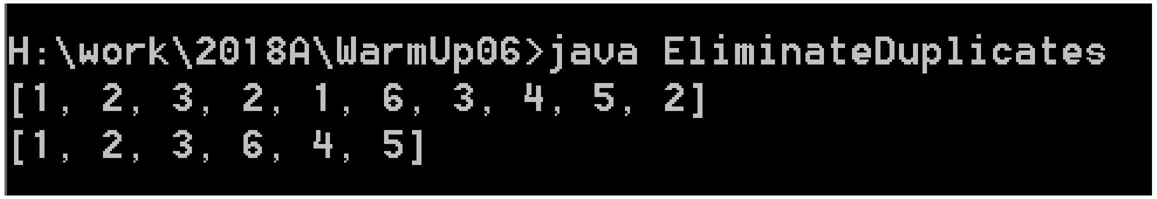
```
[12, 24, 8, 22, 15, 4, 8, 6, 13]
After looping 14 times, each kid get 6 chocolates.
```

2 (*Eliminate duplicates*) Write a method that returns a new array by eliminating the duplicate values in the array using the following method header:

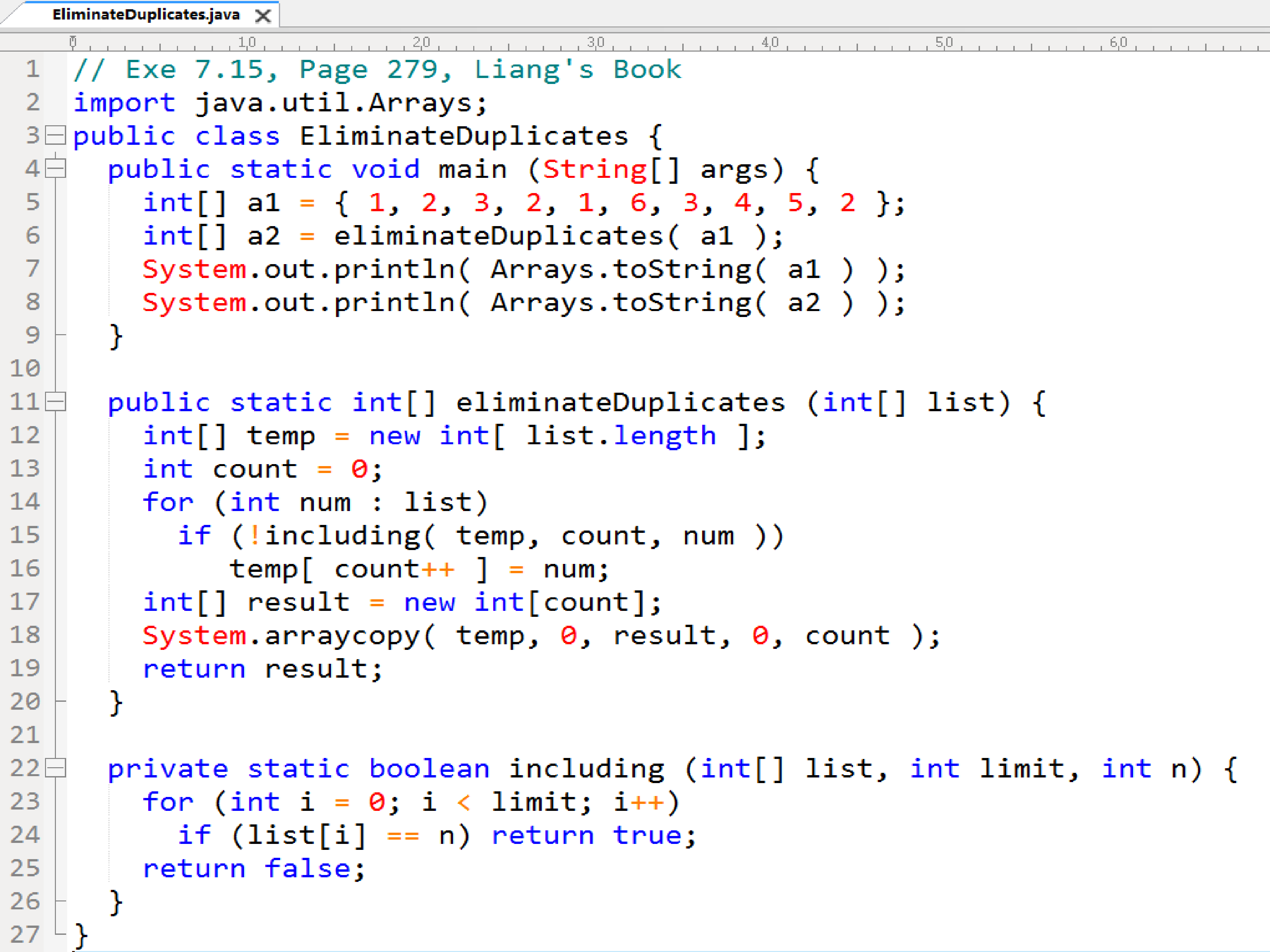
```
public static int[] eliminateDuplicates(int[] list)
```

Write a test program that reads in ten integers, invokes the method, and displays the result. Here is the sample run of the program:

```
Enter ten numbers: 1 2 3 2 1 6 3 4 5 2
The distinct numbers are: 1 2 3 6 4 5
```



```
H:\work\2018A\WarmUp06>java EliminateDuplicates
[1, 2, 3, 2, 1, 6, 3, 4, 5, 2]
[1, 2, 3, 6, 4, 5]
```



```java
// Exe 7.15, Page 279, Liang's Book
import java.util.Arrays;
public class EliminateDuplicates {
  public static void main (String[] args) {
    int[] a1 = { 1, 2, 3, 2, 1, 6, 3, 4, 5, 2 };
    int[] a2 = eliminateDuplicates( a1 );
    System.out.println( Arrays.toString( a1 ) );
    System.out.println( Arrays.toString( a2 ) );
  }

  public static int[] eliminateDuplicates (int[] list) {
    int[] temp = new int[ list.length ];
    int count = 0;
    for (int num : list)
      if (!including( temp, count, num ))
         temp[ count++ ] = num;
    int[] result = new int[count];
    System.arraycopy( temp, 0, result, 0, count );
    return result;
  }

  private static boolean including (int[] list, int limit, int n) {
    for (int i = 0; i < limit; i++)
      if (list[i] == n) return true;
    return false;
  }
}
```

3 (*Algebra: solve linear equations*) Write a method that solves the following 2×2 system of linear equations:

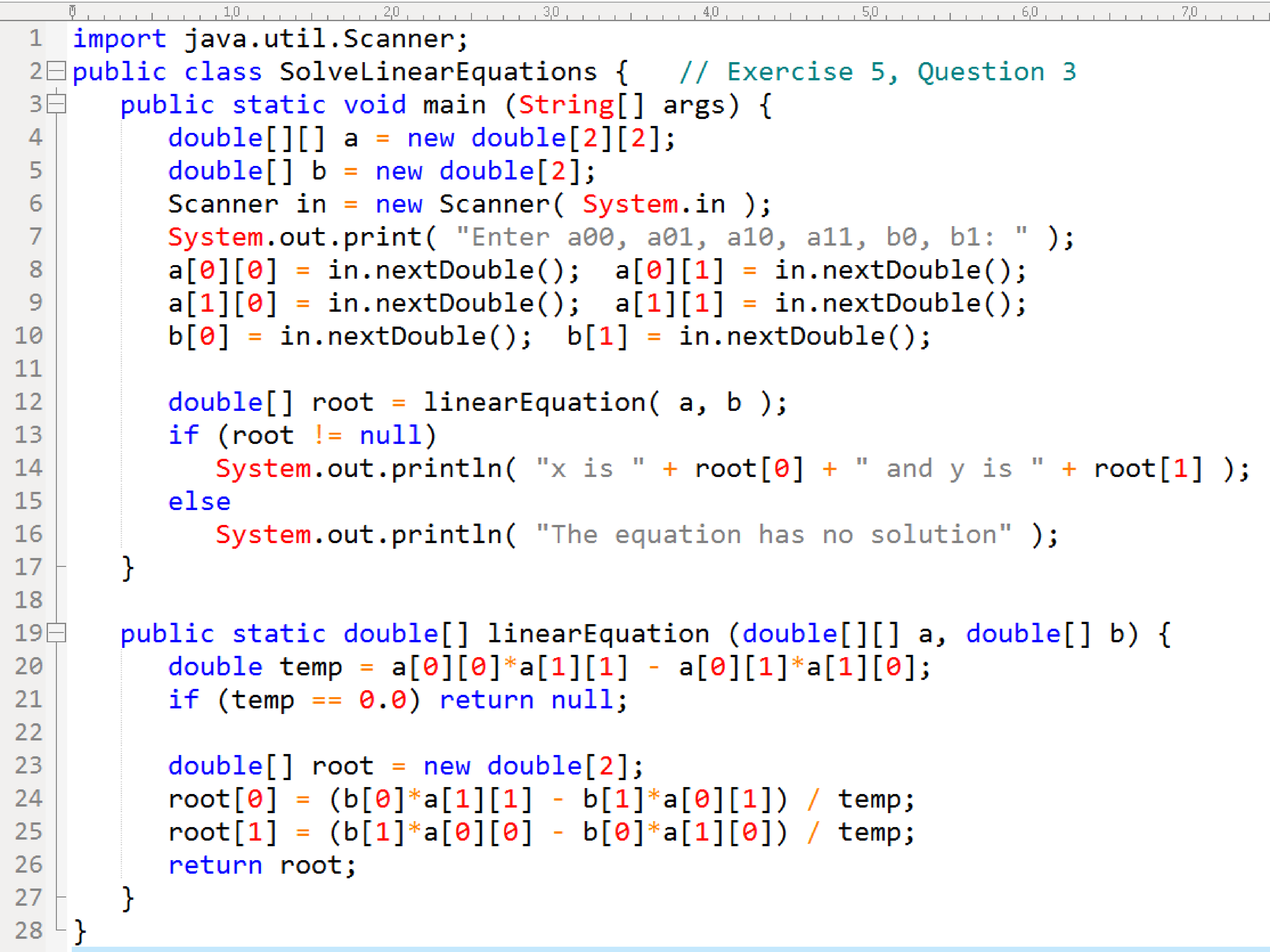
$$a_{00}x + a_{01}y = b_0 \qquad x = \frac{b_0 a_{11} - b_1 a_{01}}{a_{00}a_{11} - a_{01}a_{10}} \qquad y = \frac{b_1 a_{00} - b_0 a_{10}}{a_{00}a_{11} - a_{01}a_{10}}$$
$$a_{10}x + a_{11}y = b_1$$

The method header is

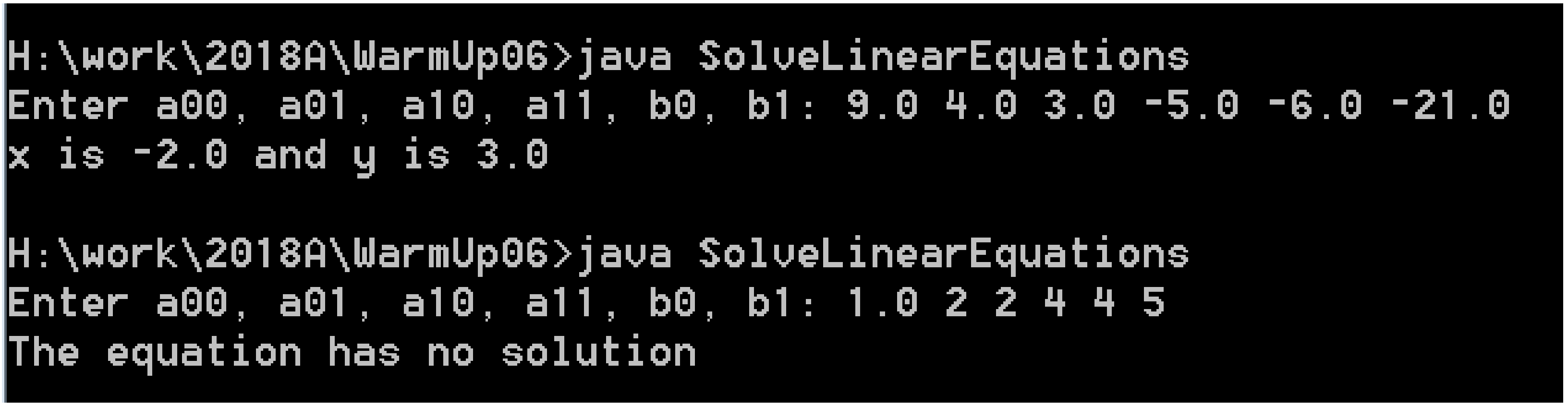
```
public static double[] linearEquation(double[][] a, double[] b)
```

The method returns `null` if $a_{00}a_{11} - a_{01}a_{10}$ is `0`. Write a test program that prompts the user to enter a_{00}, a_{01}, a_{10}, a_{11}, b_0, and b_1, and displays the result. If $a_{00}a_{11} - a_{01}a_{10}$ is `0`, report that "The equation has no solution."

```
Enter a00, a01, a10, a11, b0, b1: 9.0 4.0 3.0 -5.0 -6.0 -21.0
x is -2.0 and y is 3.0

Enter a00, a01, a10, a11, b0, b1: 1.0 2.0 2.0 4.0 4.0 5.0
The equation has no solution
```



```java
import java.util.Scanner;
public class SolveLinearEquations {    // Exercise 5, Question 3
   public static void main (String[] args) {
      double[][] a = new double[2][2];
      double[] b = new double[2];
      Scanner in = new Scanner( System.in );
      System.out.print( "Enter a00, a01, a10, a11, b0, b1: " );
      a[0][0] = in.nextDouble();  a[0][1] = in.nextDouble();
      a[1][0] = in.nextDouble();  a[1][1] = in.nextDouble();
      b[0] = in.nextDouble();  b[1] = in.nextDouble();

      double[] root = linearEquation( a, b );
      if (root != null)
         System.out.println( "x is " + root[0] + " and y is " + root[1] );
      else
         System.out.println( "The equation has no solution" );
   }

   public static double[] linearEquation (double[][] a, double[] b) {
      double temp = a[0][0]*a[1][1] - a[0][1]*a[1][0];
      if (temp == 0.0) return null;

      double[] root = new double[2];
      root[0] = (b[0]*a[1][1] - b[1]*a[0][1]) / temp;
      root[1] = (b[1]*a[0][0] - b[0]*a[1][0]) / temp;
      return root;
   }
}
```



```
H:\work\2018A\WarmUp06>java SolveLinearEquations
Enter a00, a01, a10, a11, b0, b1: 9.0 4.0 3.0 -5.0 -6.0 -21.0
x is -2.0 and y is 3.0

H:\work\2018A\WarmUp06>java SolveLinearEquations
Enter a00, a01, a10, a11, b0, b1: 1.0 2 2 4 4 5
The equation has no solution
```

***4** (*Largest block*) Given a square matrix with the elements 0 or 1, write a program to find a maximum square submatrix whose elements are all 1s. Your program should prompt the user to enter the number of rows in the matrix. The program then displays the location of the first element in the maximum square submatrix and the number of the rows in the submatrix. Here is a sample run:

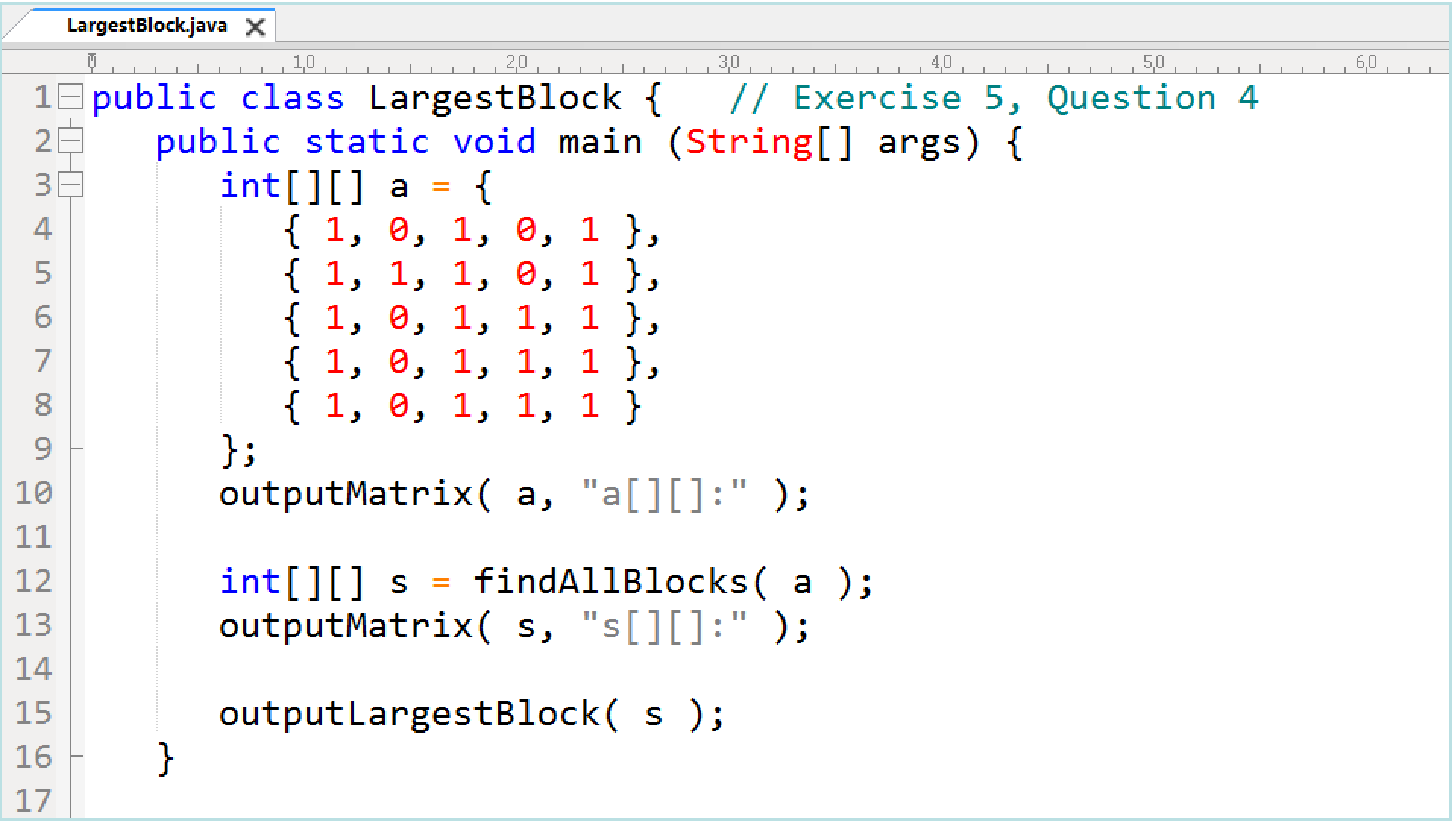
```
Enter the number of rows in the matrix: 5 ↵Enter
Enter the matrix row by row:
1 0 1 0 1 ↵Enter
1 1 1 0 1 ↵Enter
1 0 1 1 1 ↵Enter
1 0 1 1 1 ↵Enter
1 0 1 1 1 ↵Enter

The maximum square submatrix is at (2, 2) with size 3
```

If `int[][] a = int[N][N]` with elements 0 or 1,

let `int[][] s = int[N][N]`, `s[r][c]` denotes the size of the maximum sub-matrix start from `a[r][c]` towards right-down direction.

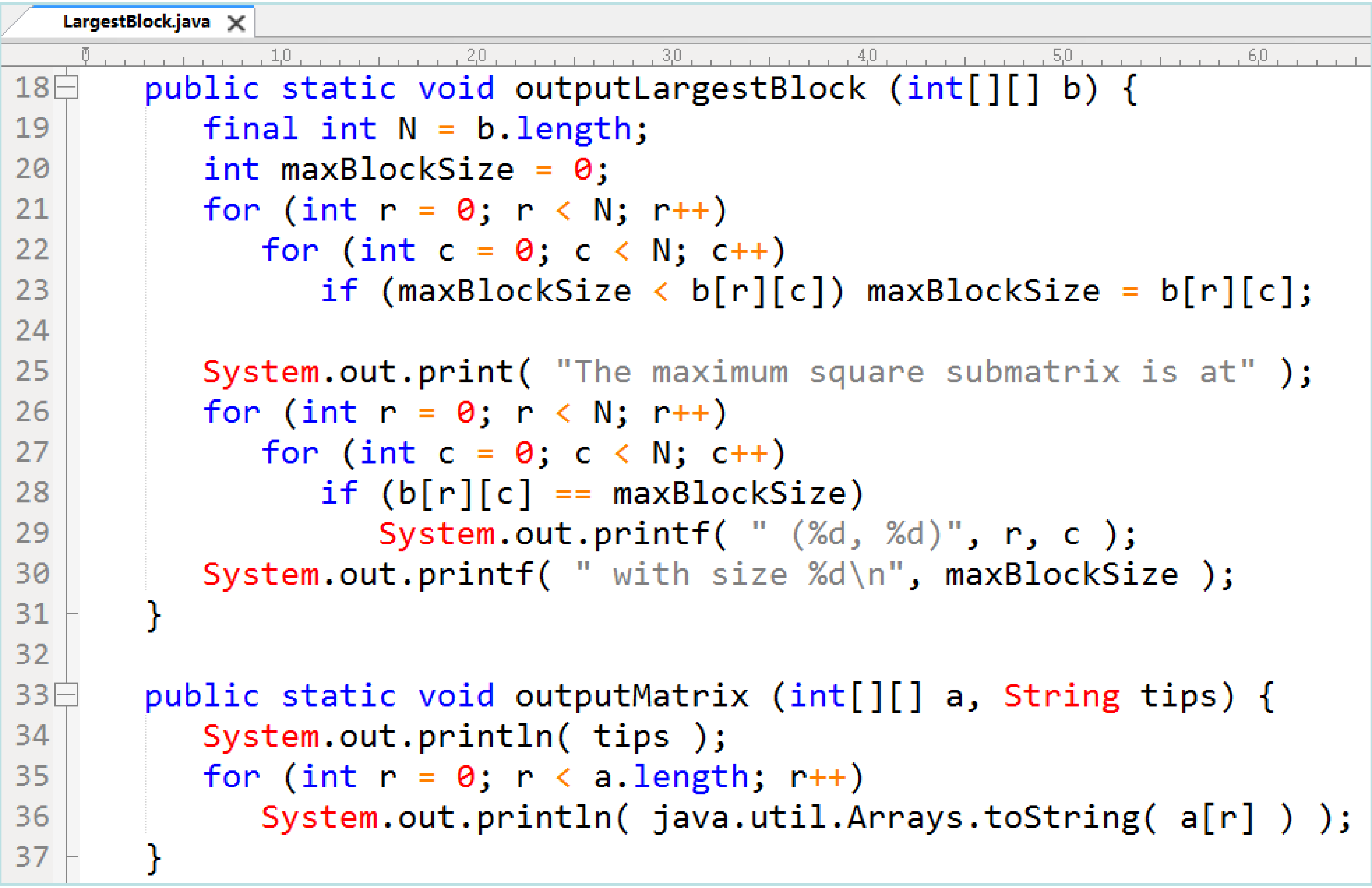
Then we could compute `s[r][c]` in the following way (using pseudo-code):

```
s[N-1][N-1 .. 0] = a[N-1][N-1 .. 0]     // last row
s[N-2 .. 0][N-1] = a[N-2 .. 0][N-1]     // last column

for (r from N-2 down to 0)
  for (c from N-2 down to 0)
     s[r][c] = (a[r][c] == 0) ? 0 :
               1 + min(s[r][c+1], s[r+1][c], s[r+1][c+1])
```

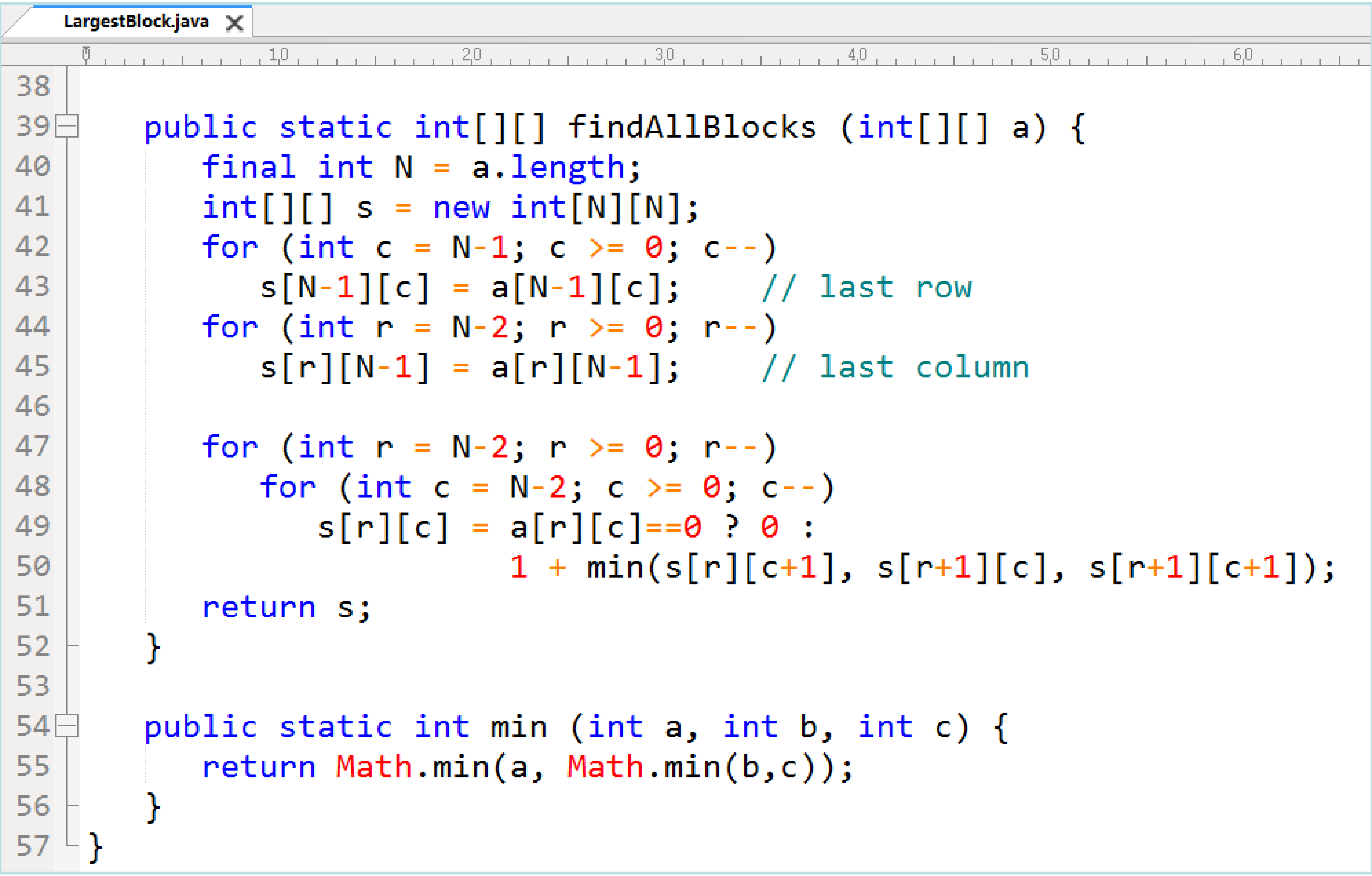


LargestBlock.java

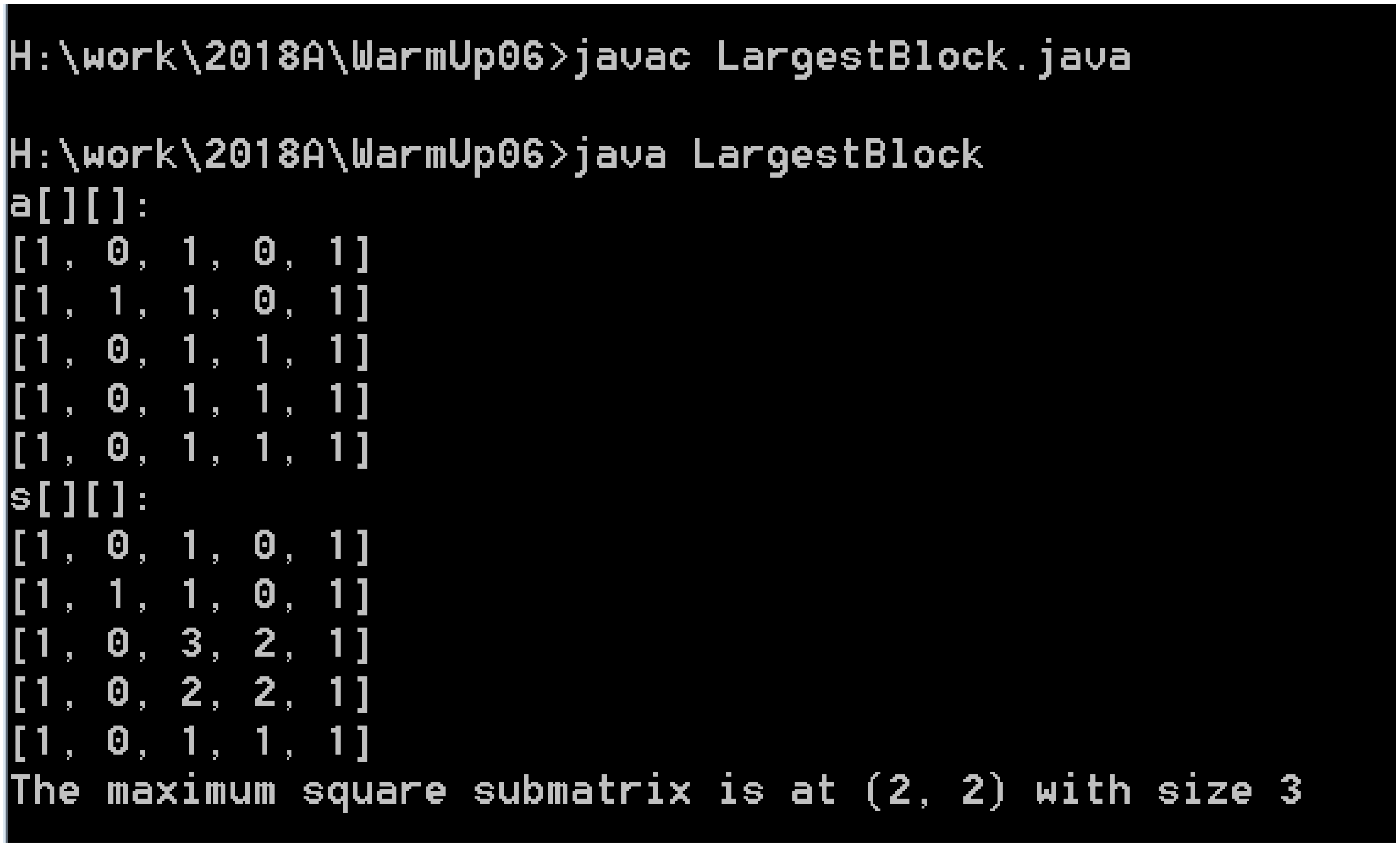
```java
public class LargestBlock {   // Exercise 5, Question 4
    public static void main (String[] args) {
        int[][] a = {
            { 1, 0, 1, 0, 1 },
            { 1, 1, 1, 0, 1 },
            { 1, 0, 1, 1, 1 },
            { 1, 0, 1, 1, 1 },
            { 1, 0, 1, 1, 1 }
        };
        outputMatrix( a, "a[][]:" );

        int[][] s = findAllBlocks( a );
        outputMatrix( s, "s[][]:" );

        outputLargestBlock( s );
    }

```


LargestBlock.java

```
   public static void outputLargestBlock (int[][] b) {
      final int N = b.length;
      int maxBlockSize = 0;
      for (int r = 0; r < N; r++)
         for (int c = 0; c < N; c++)
            if (maxBlockSize < b[r][c]) maxBlockSize = b[r][c];

      System.out.print( "The maximum square submatrix is at" );
      for (int r = 0; r < N; r++)
         for (int c = 0; c < N; c++)
            if (b[r][c] == maxBlockSize)
               System.out.printf( " (%d, %d)", r, c );
      System.out.printf( " with size %d\n", maxBlockSize );
   }

   public static void outputMatrix (int[][] a, String tips) {
      System.out.println( tips );
      for (int r = 0; r < a.length; r++)
         System.out.println( java.util.Arrays.toString( a[r] ) );
   }
```



```java

    public static int[][] findAllBlocks (int[][] a) {
        final int N = a.length;
        int[][] s = new int[N][N];
        for (int c = N-1; c >= 0; c--)
            s[N-1][c] = a[N-1][c];     // last row
        for (int r = N-2; r >= 0; r--)
            s[r][N-1] = a[r][N-1];     // last column

        for (int r = N-2; r >= 0; r--)
            for (int c = N-2; c >= 0; c--)
                s[r][c] = a[r][c]==0 ? 0 :
                          1 + min(s[r][c+1], s[r+1][c], s[r+1][c+1]);
        return s;
    }

    public static int min (int a, int b, int c) {
        return Math.min(a, Math.min(b,c));
    }
}
```



```
H:\work\2018A\WarmUp06>javac LargestBlock.java

H:\work\2018A\WarmUp06>java LargestBlock
a[][]:
[1, 0, 1, 0, 1]
[1, 1, 1, 0, 1]
[1, 0, 1, 1, 1]
[1, 0, 1, 1, 1]
[1, 0, 1, 1, 1]
s[][]:
[1, 0, 1, 0, 1]
[1, 1, 1, 0, 1]
[1, 0, 3, 2, 1]
[1, 0, 2, 2, 1]
[1, 0, 1, 1, 1]
The maximum square submatrix is at (2, 2) with size 3
```