CS 24 AP Computer Science A Review

Week 9: Algorithm I

DR. ERIC CHOU
IEEE SENIOR MEMBER

SECTION 1

Algorithms

Algorithm

- The word algorithm can be simply defined as "a process to be followed to solve a problem."
- The number of algorithms to learn is infinite, so rather than attempting to learn all possible algorithms, our goal will be to learn how to develop an algorithm.
- In fact, most FRQ problems in AP Computer Science is targeted at demonstration of problem solving capability which is actually writing algorithms.

Study of Algorithm and Coding

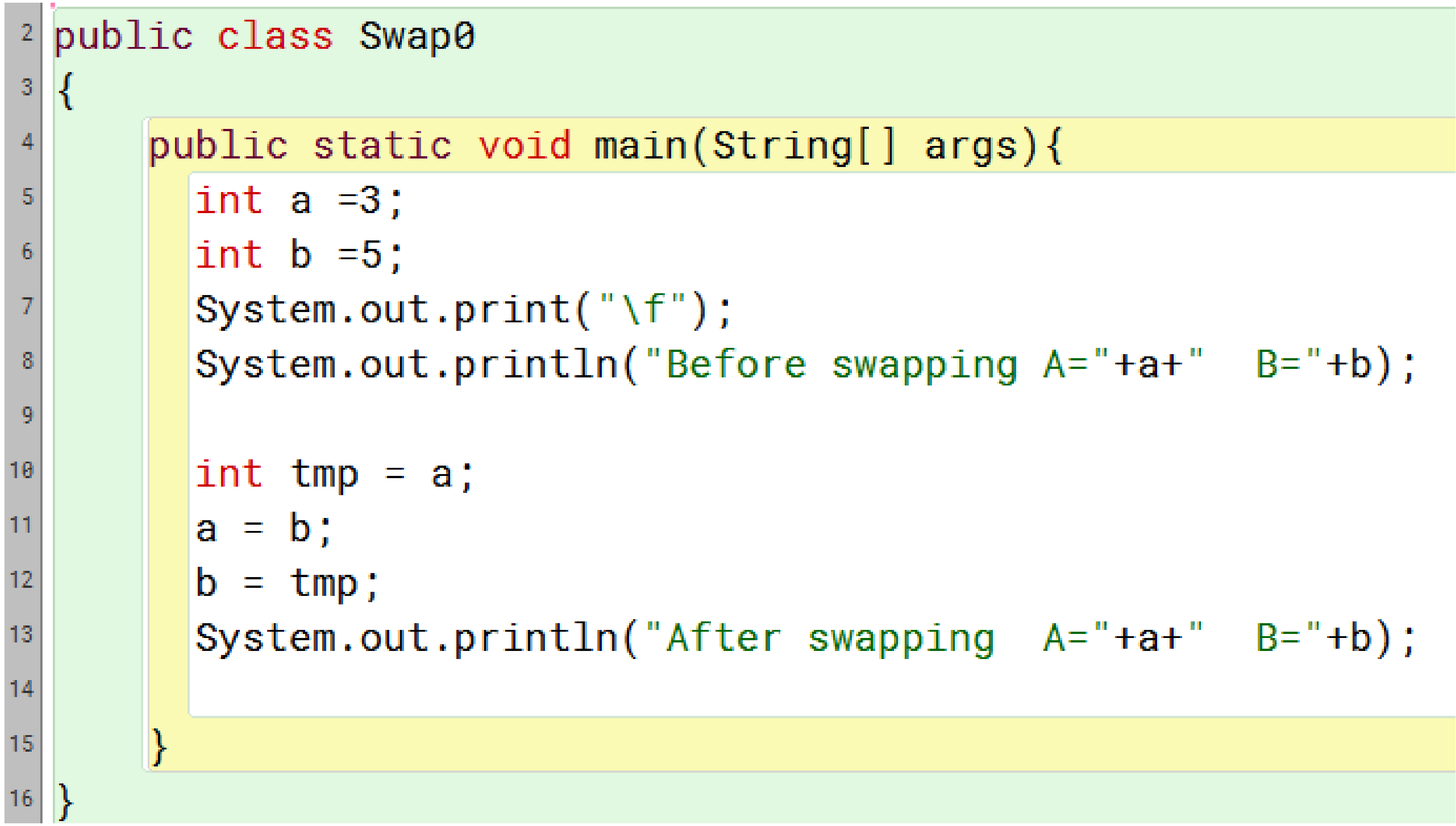
1. Basic design patterns.
2. Program control structure.
3. Development of algorithms from scratch.

SECTION 2

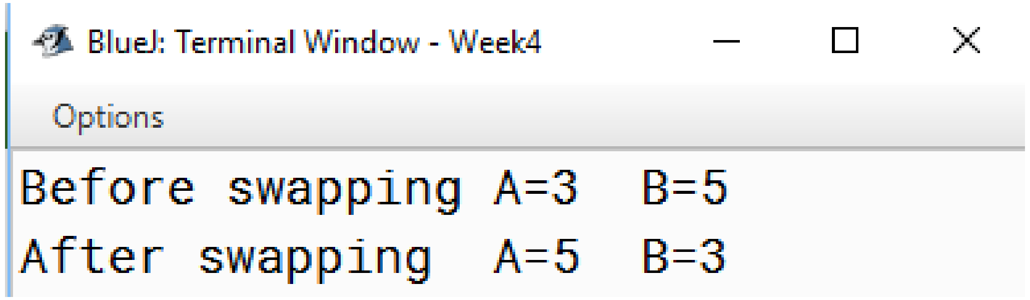
Basic Design Patterns

Swap Algorithm

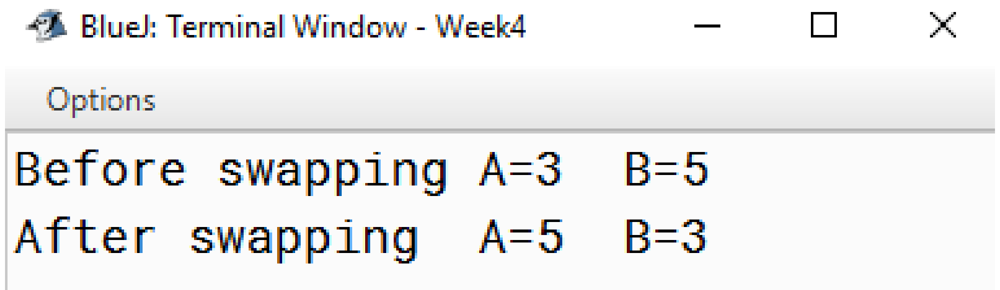
```
int a = 3;
 int b = 5;


 /* swap pattern */
  int tmp = a;
  a = b;
  b = tmp;
```



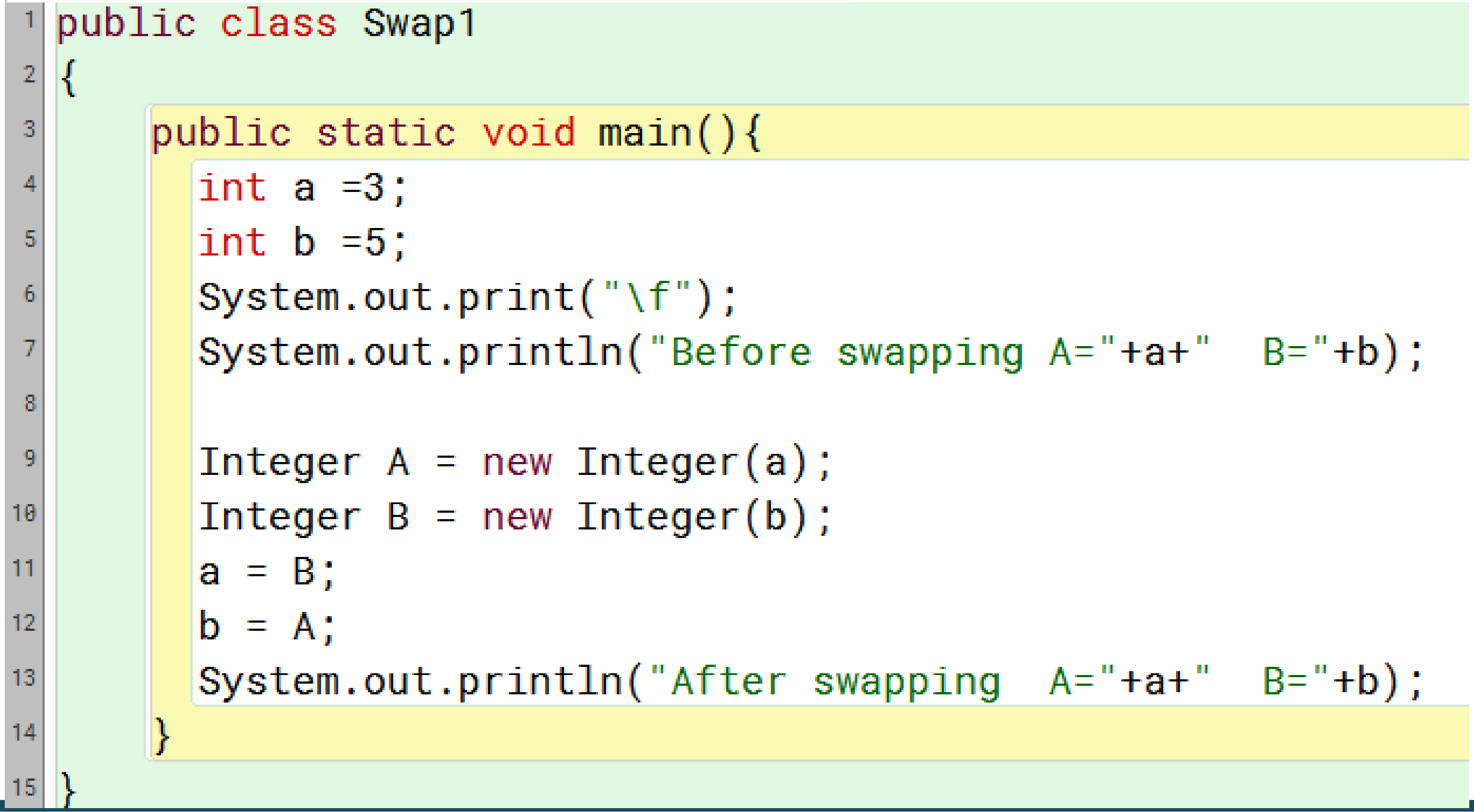
```java
public class Swap0
{
    public static void main(String[] args){
      int a =3;
      int b =5;
      System.out.print("\f");
      System.out.println("Before swapping A="+a+"  B="+b);

      int tmp = a;
      a = b;
      b = tmp;
      System.out.println("After swapping  A="+a+"  B="+b);

    }
}
```



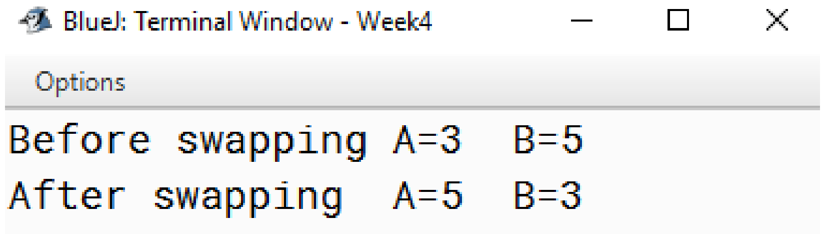
```
Before swapping A=3  B=5
After swapping  A=5  B=3
```



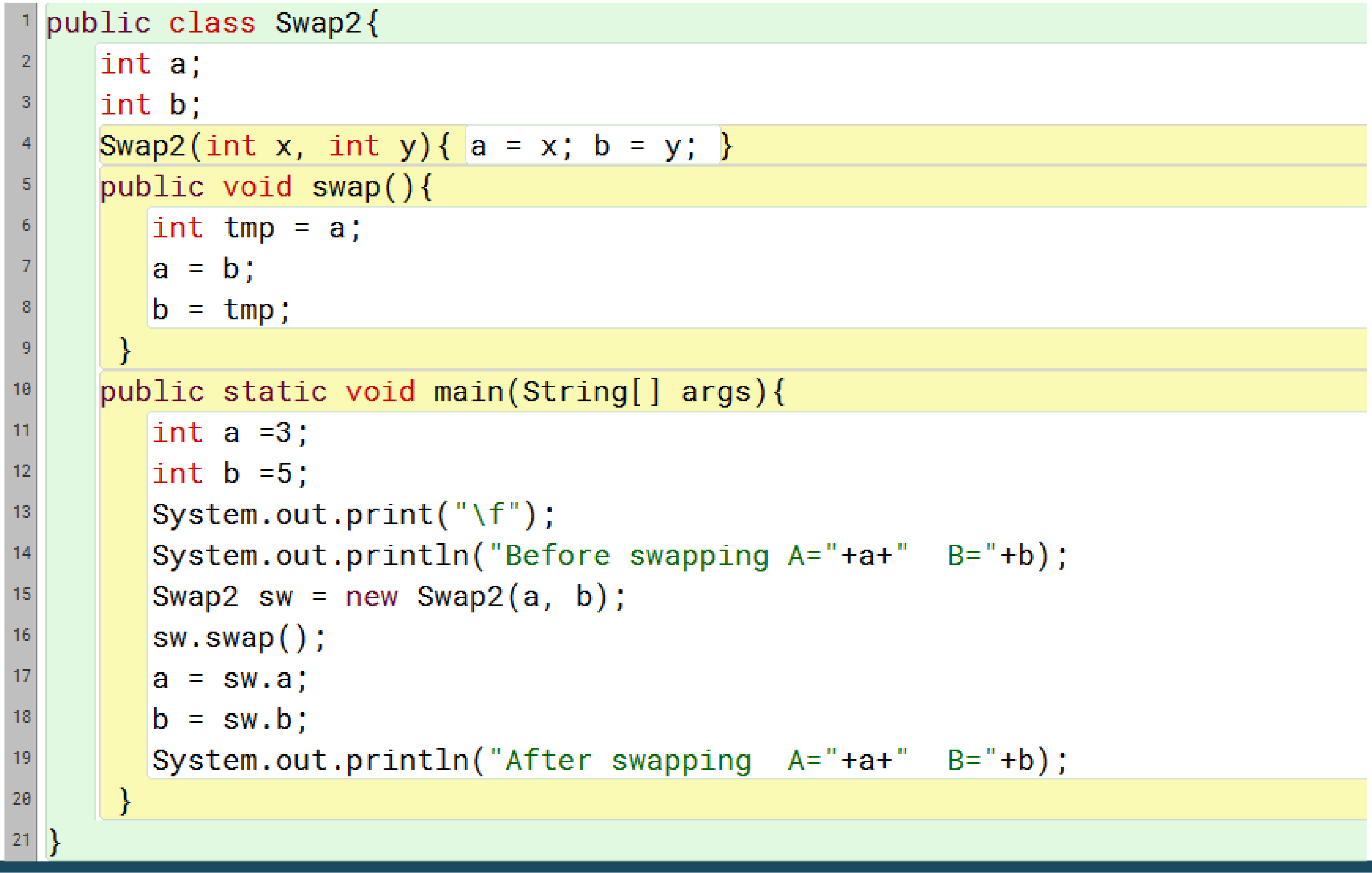
```
Before swapping A=3  B=5
After swapping  A=5  B=3
```



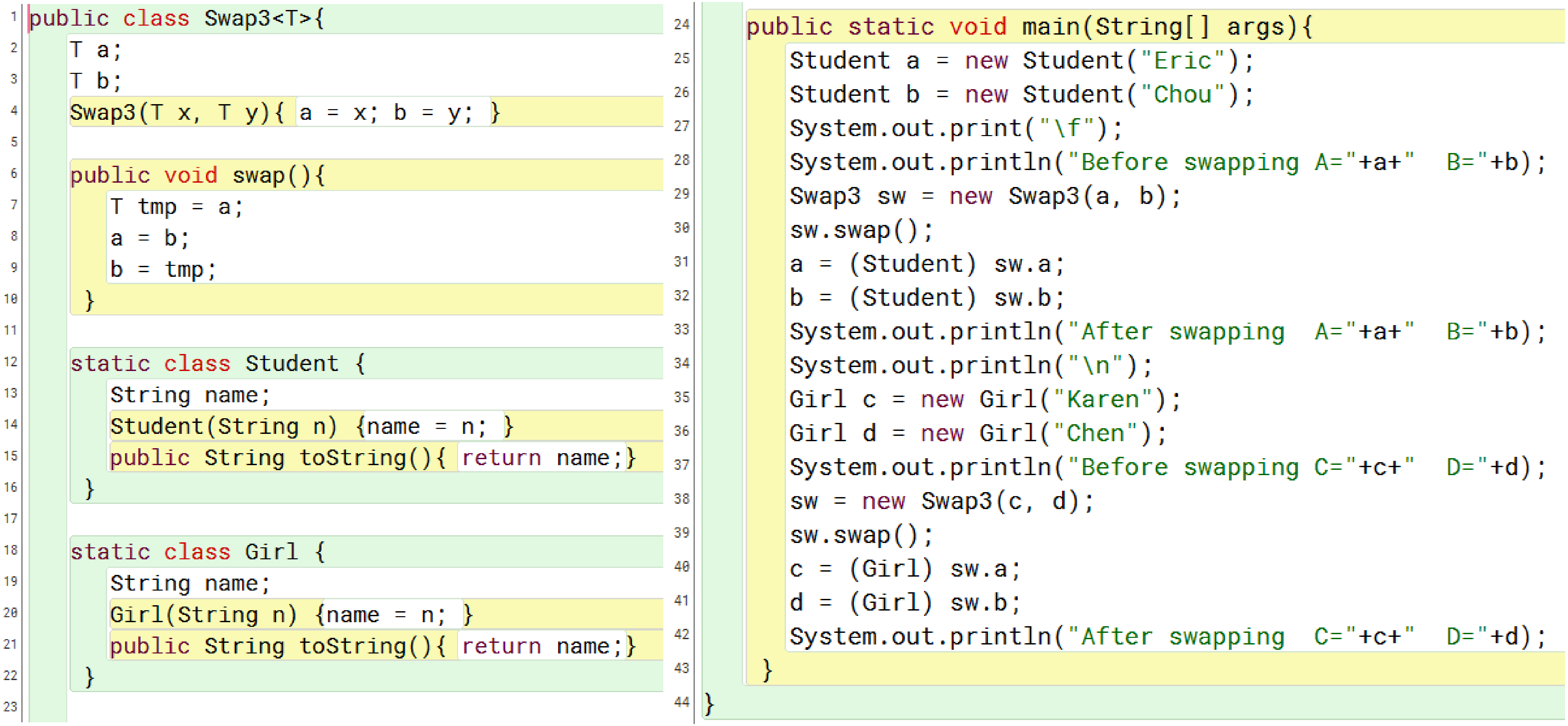
```java
public class Swap1
{
    public static void main(){
      int a =3;
      int b =5;
      System.out.print("\f");
      System.out.println("Before swapping A="+a+"  B="+b);

      Integer A = new Integer(a);
      Integer B = new Integer(b);
      a = B;
      b = A;
      System.out.println("After swapping  A="+a+"  B="+b);
    }
}
```



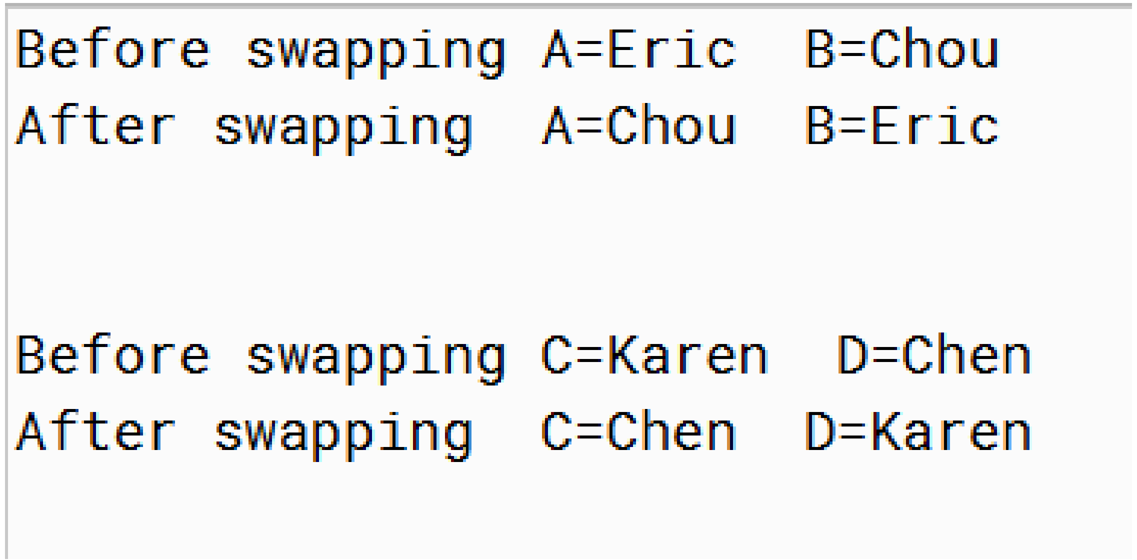
```
Before swapping A=3   B=5
After swapping  A=5   B=3
```



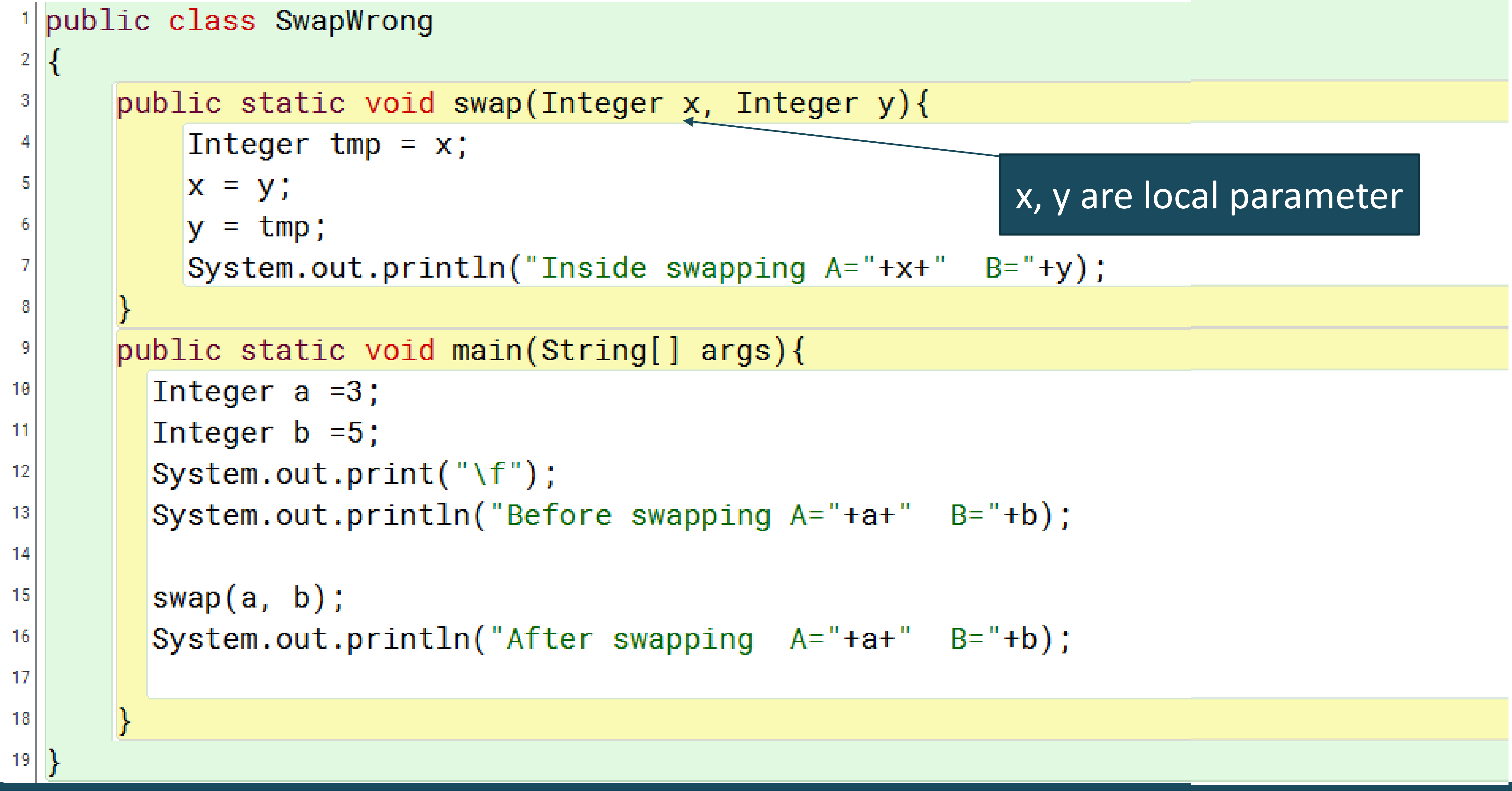
```
public class Swap2{
    int a;
    int b;
    Swap2(int x, int y){ a = x; b = y; }
    public void swap(){
        int tmp = a;
        a = b;
        b = tmp;
     }
    public static void main(String[] args){
        int a =3;
        int b =5;
        System.out.print("\f");
        System.out.println("Before swapping A="+a+"   B="+b);
        Swap2 sw = new Swap2(a, b);
        sw.swap();
        a = sw.a;
        b = sw.b;
        System.out.println("After swapping  A="+a+"   B="+b);
     }
}
```



```java
public class Swap3<T>{
    T a;
    T b;
    Swap3(T x, T y){ a = x; b = y; }

    public void swap(){
        T tmp = a;
        a = b;
        b = tmp;
    }

    static class Student {
        String name;
        Student(String n) {name = n; }
        public String toString(){ return name;}
    }

    static class Girl {
        String name;
        Girl(String n) {name = n; }
        public String toString(){ return name;}
    }

    public static void main(String[] args){
        Student a = new Student("Eric");
        Student b = new Student("Chou");
        System.out.print("\f");
        System.out.println("Before swapping A="+a+"  B="+b);
        Swap3 sw = new Swap3(a, b);
        sw.swap();
        a = (Student) sw.a;
        b = (Student) sw.b;
        System.out.println("After swapping  A="+a+"  B="+b);
        System.out.println("\n");
        Girl c = new Girl("Karen");
        Girl d = new Girl("Chen");
        System.out.println("Before swapping C="+c+"  D="+d);
        sw = new Swap3(c, d);
        sw.swap();
        c = (Girl) sw.a;
        d = (Girl) sw.b;
        System.out.println("After swapping  C="+c+"  D="+d);
    }
}
```



```
Before swapping A=Eric  B=Chou
After swapping  A=Chou  B=Eric



Before swapping C=Karen  D=Chen
After swapping  C=Chen  D=Karen
```



```java
public class SwapWrong
{
    public static void swap(Integer x, Integer y){
        Integer tmp = x;
        x = y;
        y = tmp;
        System.out.println("Inside swapping A="+x+"  B="+y);
    }
    public static void main(String[] args){
      Integer a =3;
      Integer b =5;
      System.out.print("\f");
      System.out.println("Before swapping A="+a+"  B="+b);

      swap(a, b);
      System.out.println("After swapping  A="+a+"  B="+b);

    }
}
```

x, y are local parameter

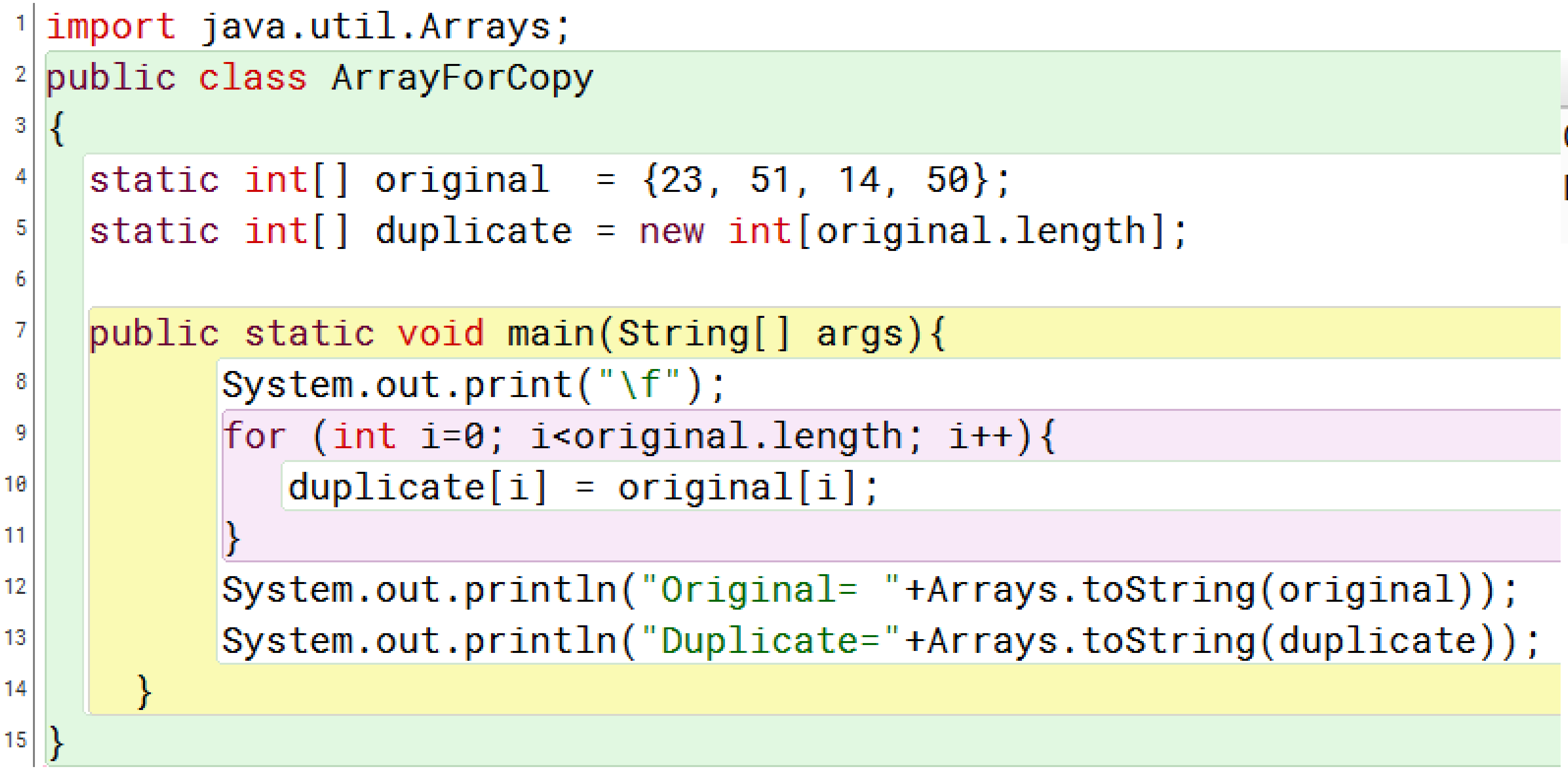
SECTION 3

Copy Algorithms

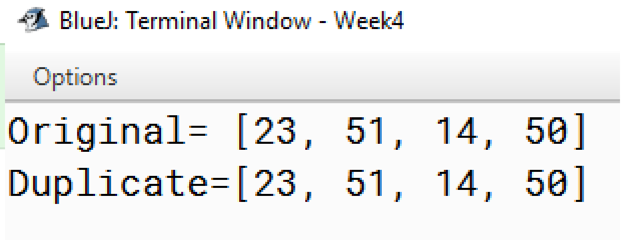
Copy algorithm for the Array and ArrayList

- Array copy (for-loop)
- Array copy (for-each loop)
- ArrayList copy (for-loop)
- ArrayList copy (for-each loop)
- 1 D Array copied to 2D Array
- 2 D Array copied to 1D Array

Array copy (for-loop)

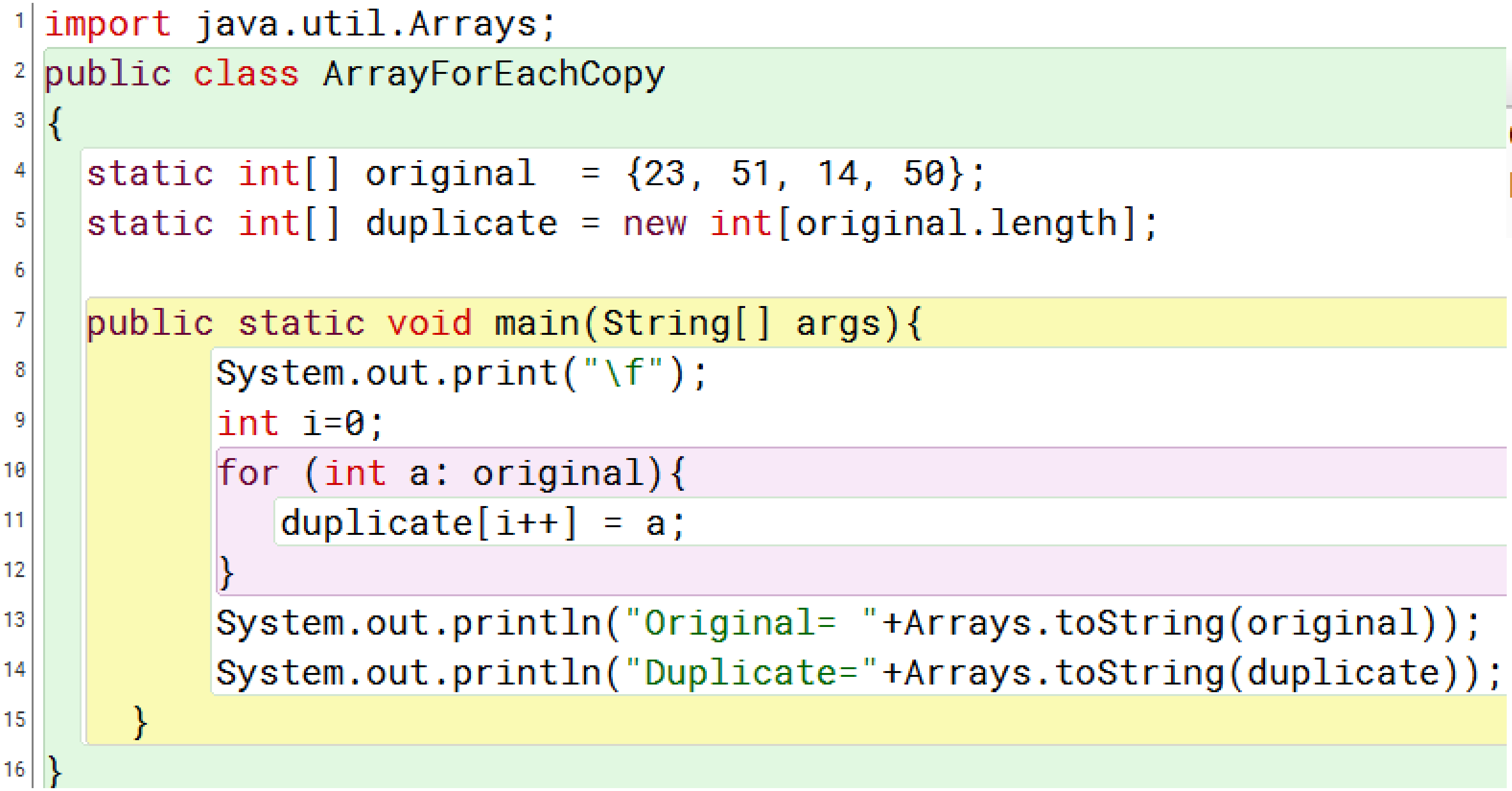


```java
import java.util.Arrays;
public class ArrayForCopy
{
    static int[] original  = {23, 51, 14, 50};
    static int[] duplicate = new int[original.length];

    public static void main(String[] args){
            System.out.print("\f");
            for (int i=0; i<original.length; i++){
                duplicate[i] = original[i];
            }
            System.out.println("Original= "+Arrays.toString(original));
            System.out.println("Duplicate="+Arrays.toString(duplicate));
      }
}
```

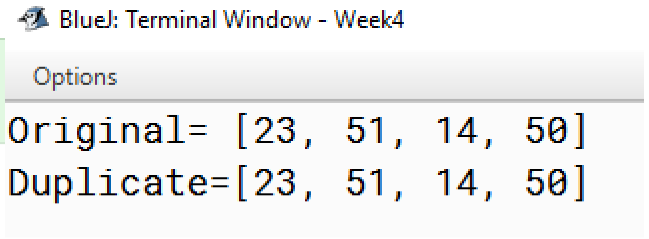


```
Original= [23, 51, 14, 50]
Duplicate=[23, 51, 14, 50]
```

Array copy (for-each loop)

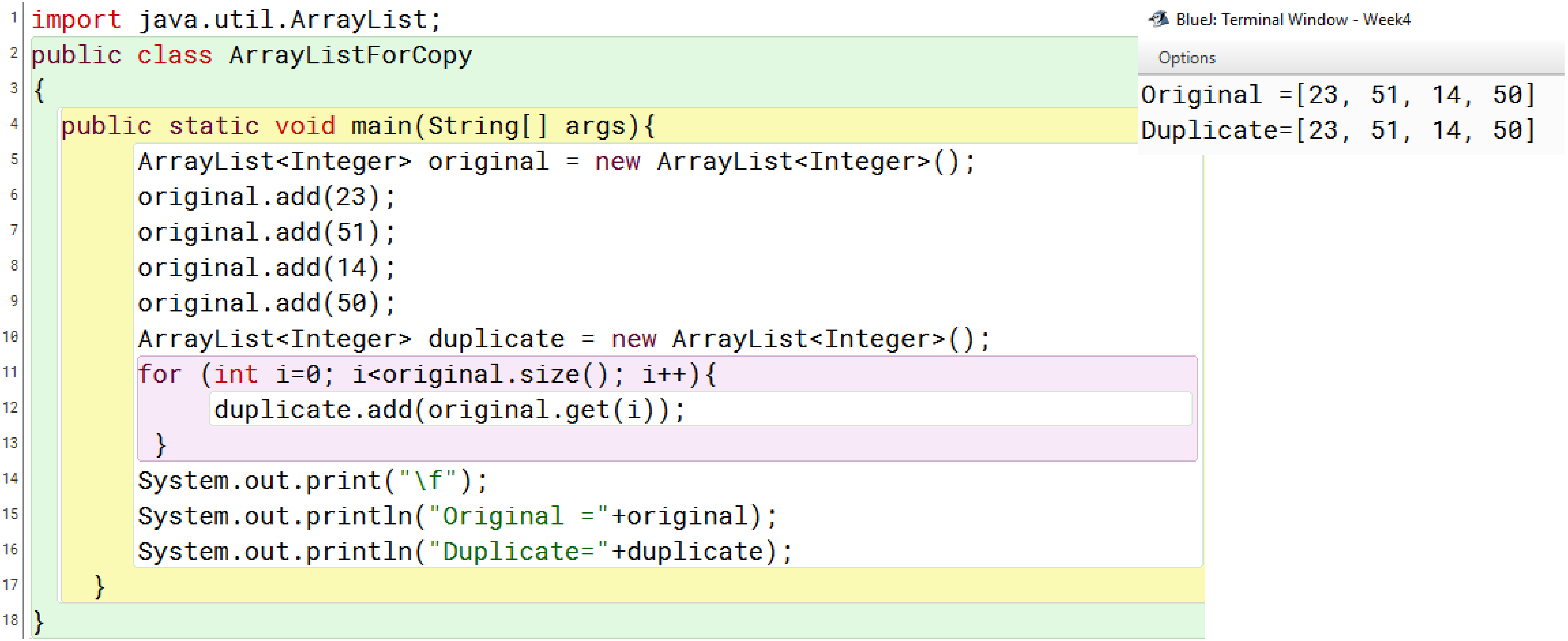


```java
import java.util.Arrays;
public class ArrayForEachCopy
{
  static int[] original  = {23, 51, 14, 50};
  static int[] duplicate = new int[original.length];

  public static void main(String[] args){
        System.out.print("\f");
        int i=0;
        for (int a: original){
           duplicate[i++] = a;
        }
        System.out.println("Original= "+Arrays.toString(original));
        System.out.println("Duplicate="+Arrays.toString(duplicate));
    }
}
```



```
Original= [23, 51, 14, 50]
Duplicate=[23, 51, 14, 50]
```

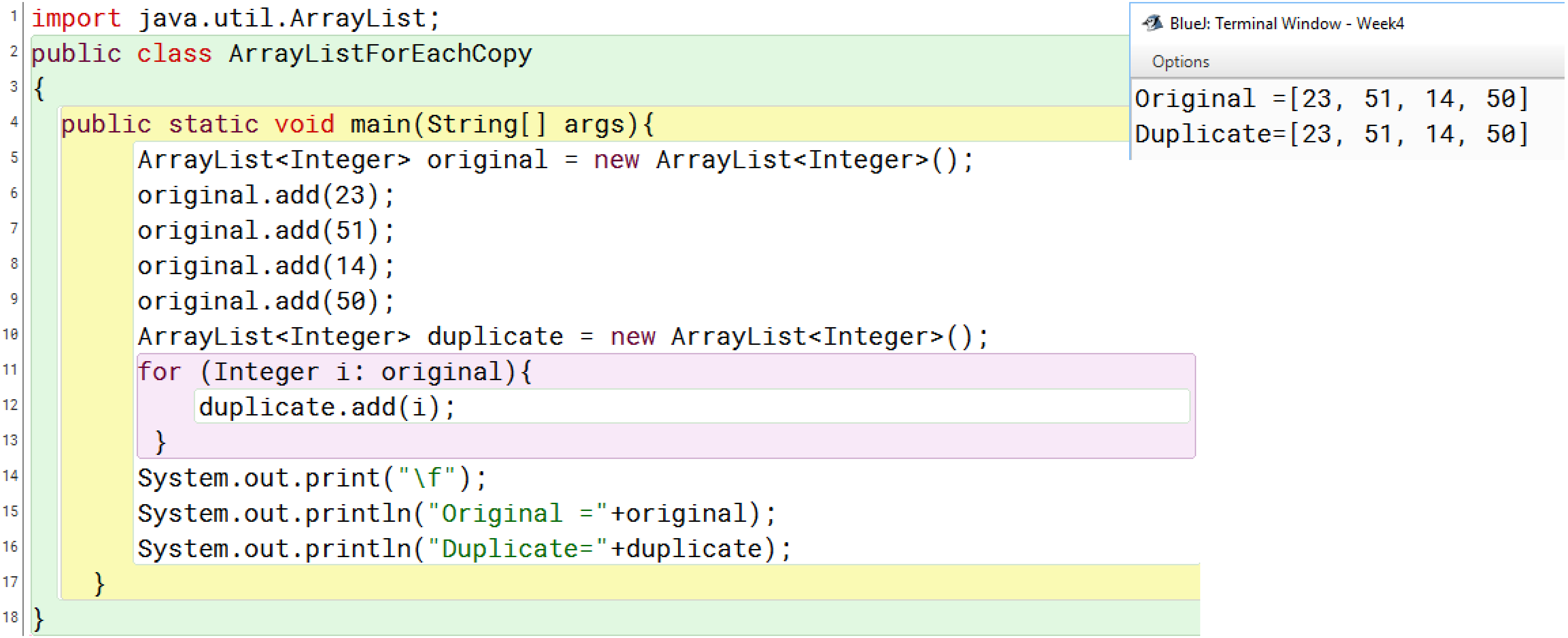
ArrayList copy (for-loop)



```java
import java.util.ArrayList;
public class ArrayListForCopy
{
  public static void main(String[] args){
       ArrayList<Integer> original = new ArrayList<Integer>();
       original.add(23);
       original.add(51);
       original.add(14);
       original.add(50);
       ArrayList<Integer> duplicate = new ArrayList<Integer>();
       for (int i=0; i<original.size(); i++){
            duplicate.add(original.get(i));
        }
       System.out.print("\f");
       System.out.println("Original ="+original);
       System.out.println("Duplicate="+duplicate);
   }
}
```

```
Original =[23, 51, 14, 50]
Duplicate=[23, 51, 14, 50]
```

ArrayList copy (for-each loop)

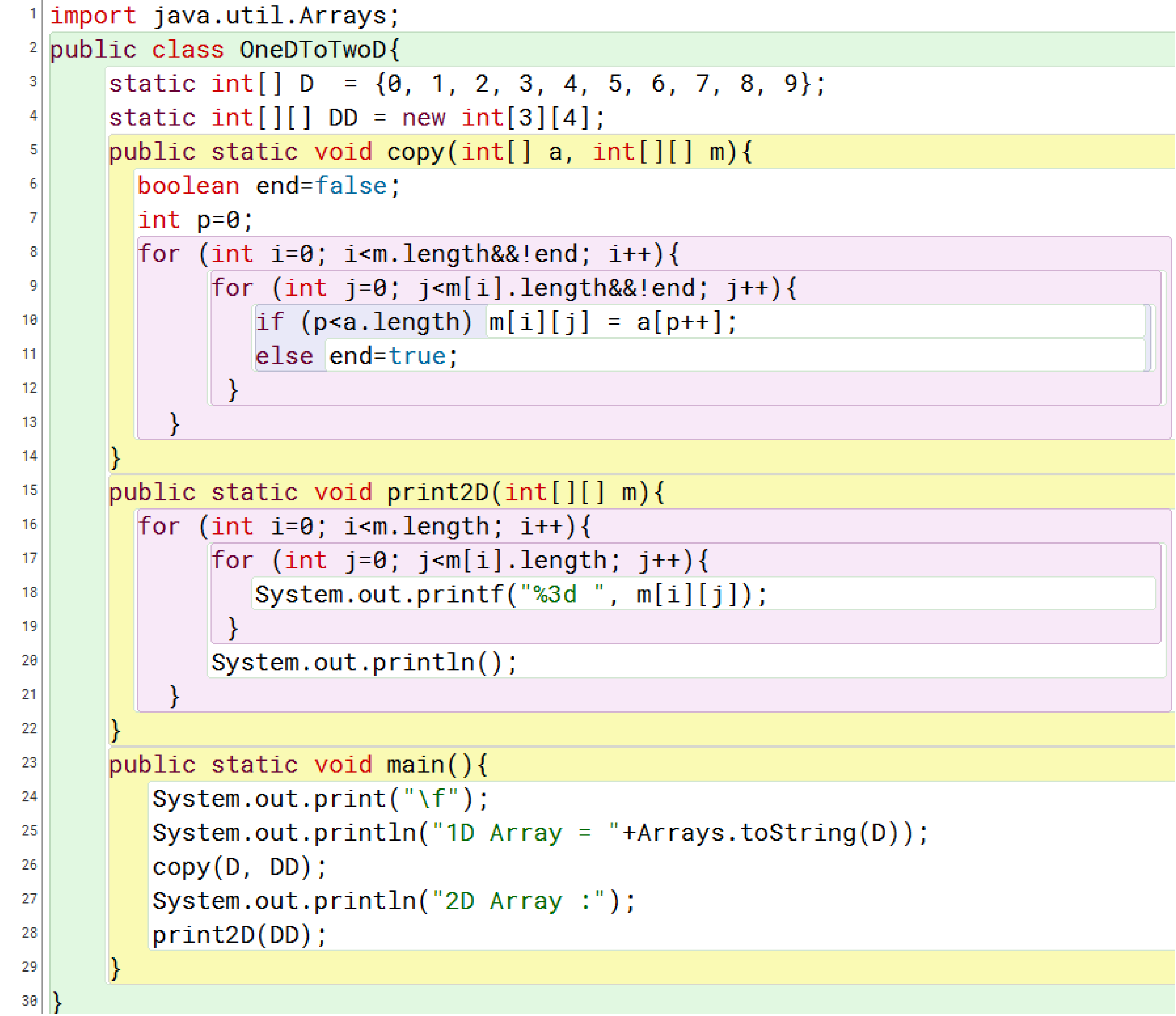


```java
import java.util.ArrayList;
public class ArrayListForEachCopy
{
  public static void main(String[] args){
        ArrayList<Integer> original = new ArrayList<Integer>();
        original.add(23);
        original.add(51);
        original.add(14);
        original.add(50);
        ArrayList<Integer> duplicate = new ArrayList<Integer>();
        for (Integer i: original){
            duplicate.add(i);
         }
        System.out.print("\f");
        System.out.println("Original ="+original);
        System.out.println("Duplicate="+duplicate);
    }
}
```

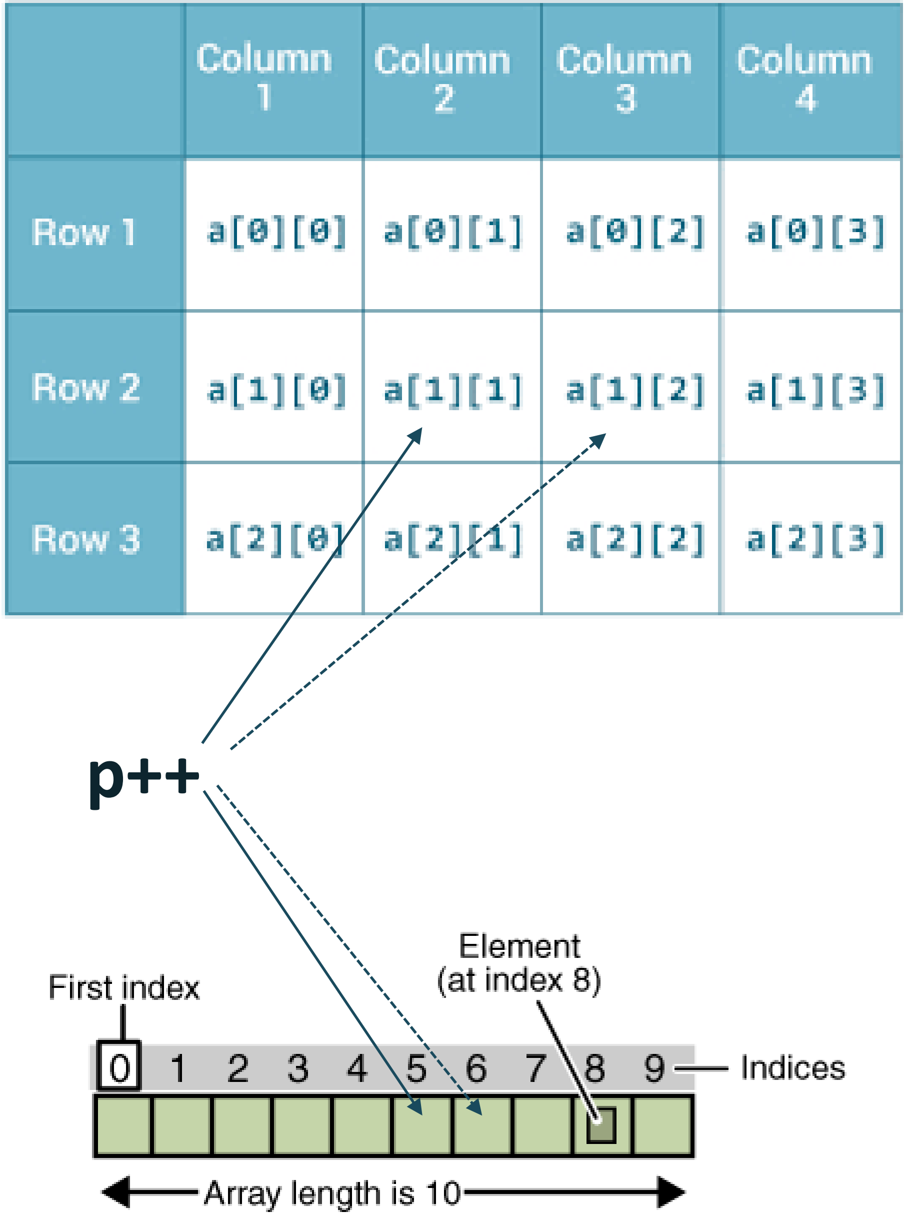
```
Original =[23, 51, 14, 50]
Duplicate=[23, 51, 14, 50]
```

Array copy from 1D to 2D

- The 2D use normal traversal.
- The 1D use a index pointer to traverse
  - If the 2D array has more elements, then ends the traversal when the 1D array is all copied.
  - If the 1D has more elements, ends the traversal when the 2D traversal is done.

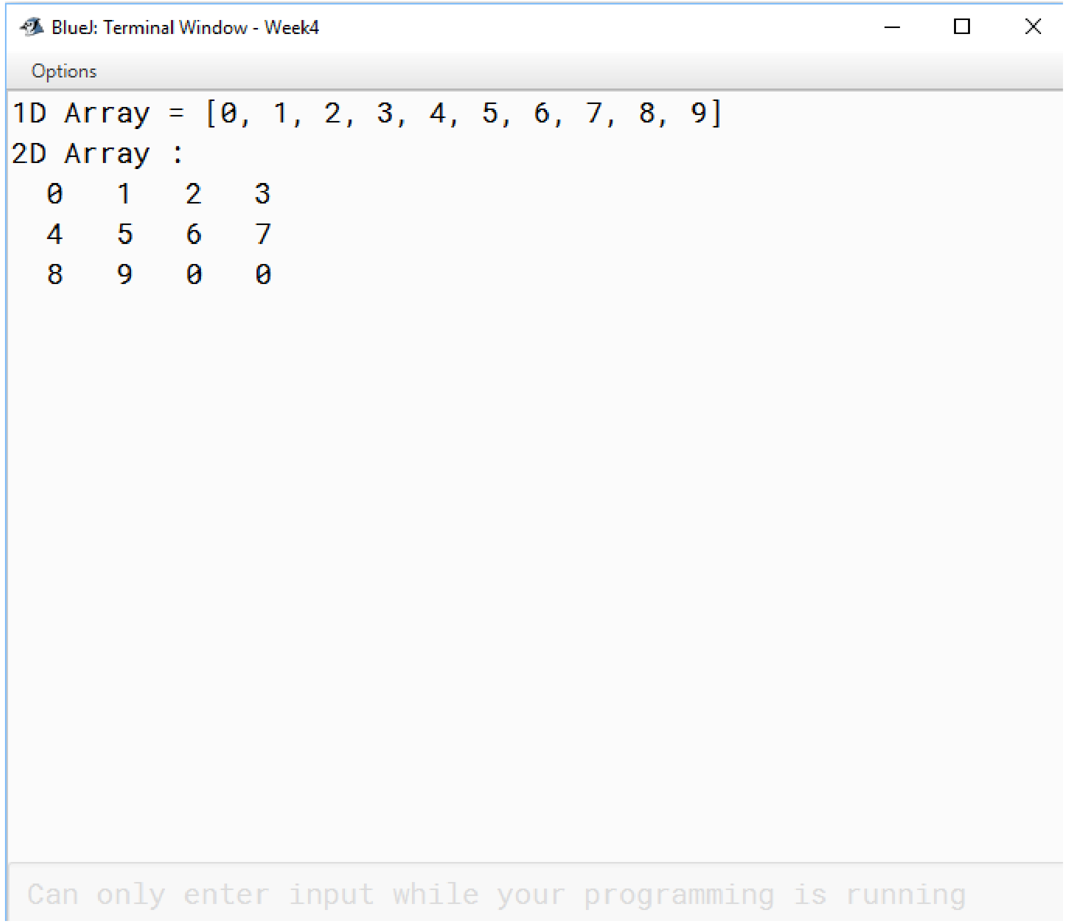


```java
import java.util.Arrays;
public class OneDToTwoD{
    static int[] D  = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    static int[][] DD = new int[3][4];
    public static void copy(int[] a, int[][] m){
      boolean end=false;
      int p=0;
      for (int i=0; i<m.length&&!end; i++){
           for (int j=0; j<m[i].length&&!end; j++){
               if (p<a.length) m[i][j] = a[p++];
               else end=true;
           }
       }
    }
    public static void print2D(int[][] m){
      for (int i=0; i<m.length; i++){
           for (int j=0; j<m[i].length; j++){
               System.out.printf("%3d ", m[i][j]);
           }
           System.out.println();
       }
    }
    public static void main(){
       System.out.print("\f");
       System.out.println("1D Array = "+Arrays.toString(D));
       copy(D, DD);
       System.out.println("2D Array :");
       print2D(DD);
    }
}
```



	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Result

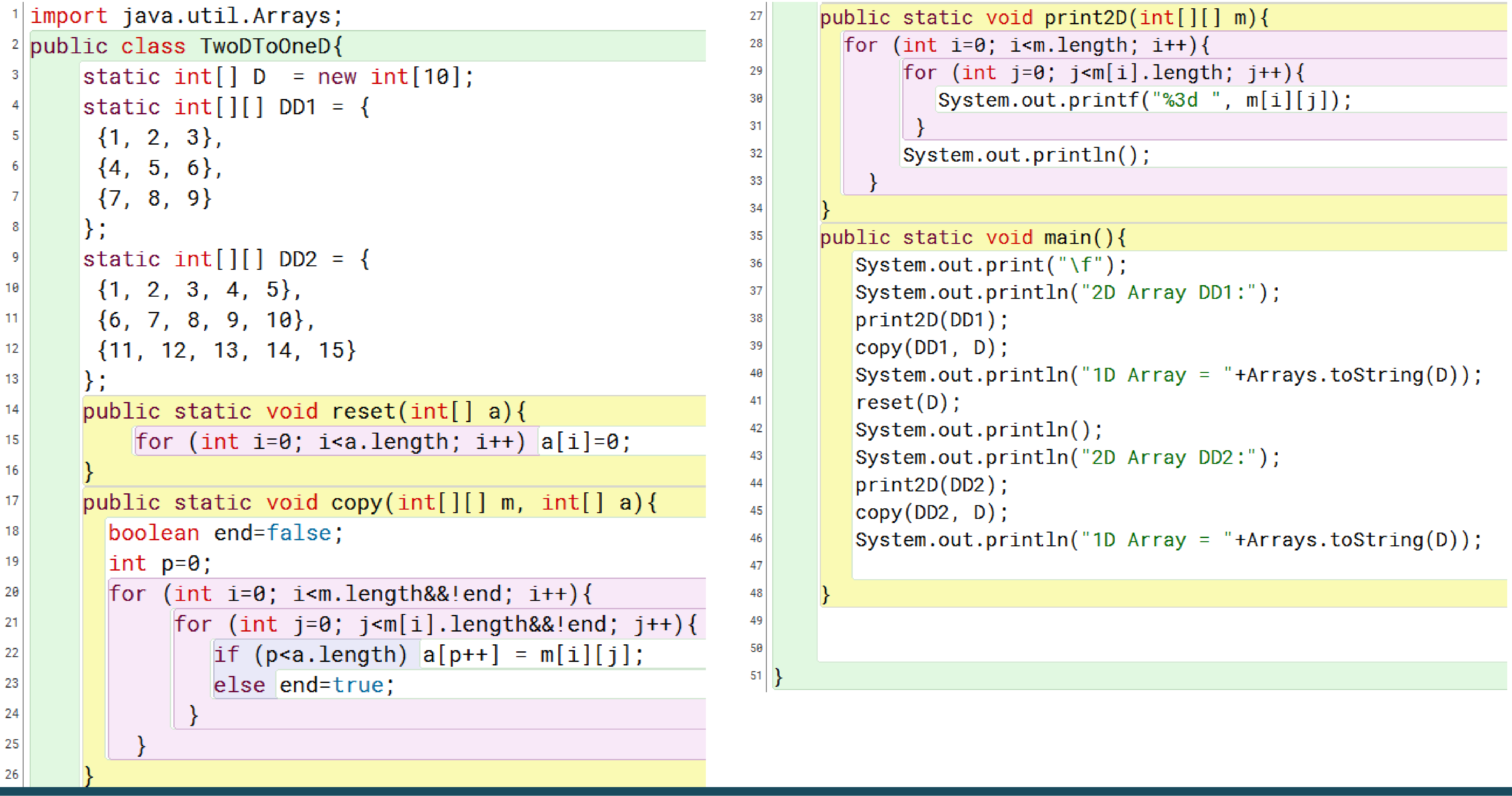

BlueJ: Terminal Window - Week4
Options

```
1D Array = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
2D Array :
  0   1   2   3
  4   5   6   7
  8   9   0   0
```

Can only enter input while your programming is running

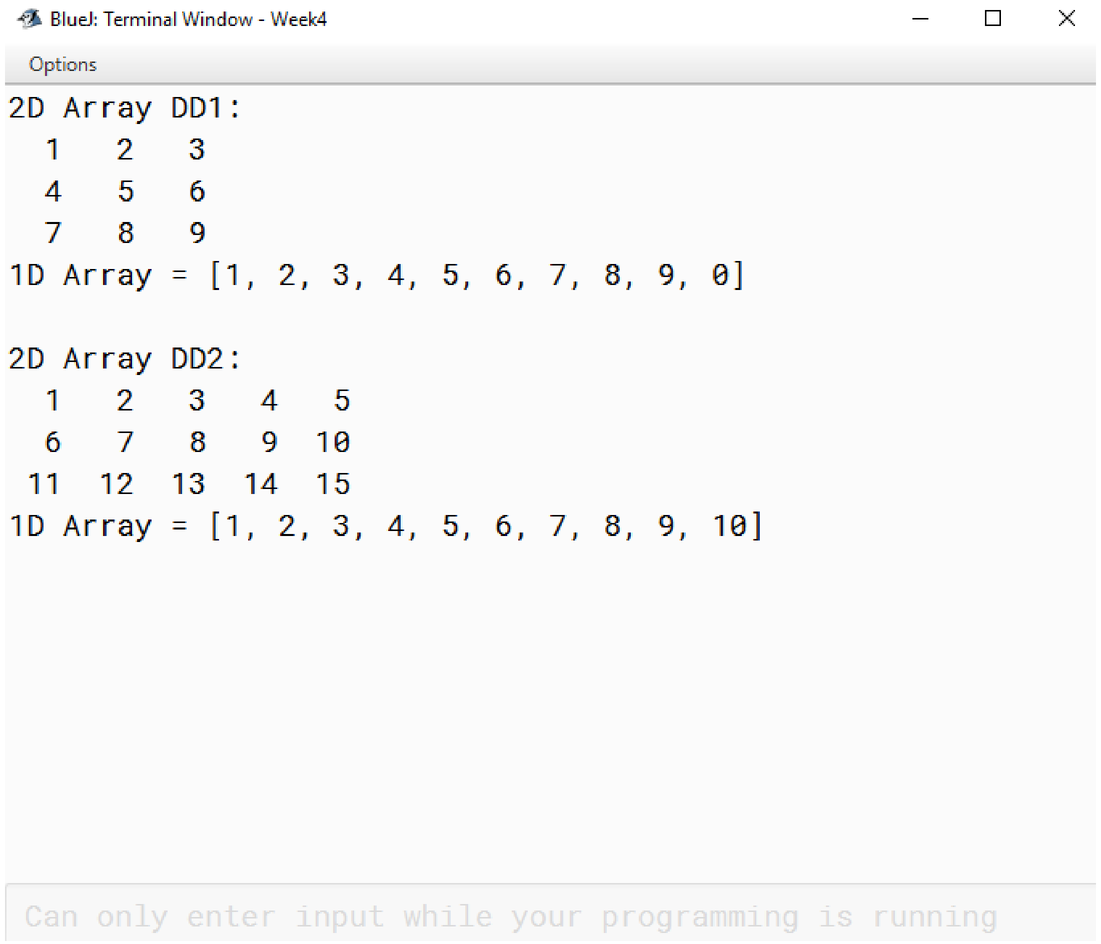
Array copy from 2D to 1D

- The 2D use normal traversal.
- The 1D use a index pointer to traverse
  - Assign value to 1D array only when the array is not all traversed through.



```
import java.util.Arrays;
public class TwoDToOneD{
    static int[] D  = new int[10];
    static int[][] DD1 = {
     {1, 2, 3},
     {4, 5, 6},
     {7, 8, 9}
    };
    static int[][] DD2 = {
     {1, 2, 3, 4, 5},
     {6, 7, 8, 9, 10},
     {11, 12, 13, 14, 15}
    };
    public static void reset(int[] a){
        for (int i=0; i<a.length; i++) a[i]=0;
    }
    public static void copy(int[][] m, int[] a){
      boolean end=false;
      int p=0;
      for (int i=0; i<m.length&&!end; i++){
           for (int j=0; j<m[i].length&&!end; j++){
              if (p<a.length) a[p++] = m[i][j];
              else end=true;
            }
        }
    }
    public static void print2D(int[][] m){
     for (int i=0; i<m.length; i++){
          for (int j=0; j<m[i].length; j++){
             System.out.printf("%3d ", m[i][j]);
           }
          System.out.println();
       }
    }
    public static void main(){
      System.out.print("\f");
      System.out.println("2D Array DD1:");
      print2D(DD1);
      copy(DD1, D);
      System.out.println("1D Array = "+Arrays.toString(D));
      reset(D);
      System.out.println();
      System.out.println("2D Array DD2:");
      print2D(DD2);
      copy(DD2, D);
      System.out.println("1D Array = "+Arrays.toString(D));

    }


}
```

Result


BlueJ: Terminal Window - Week4
Options

```
2D Array DD1:
  1   2   3
  4   5   6
  7   8   9
1D Array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]

2D Array DD2:
  1   2   3   4   5
  6   7   8   9  10
 11  12  13  14  15
1D Array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

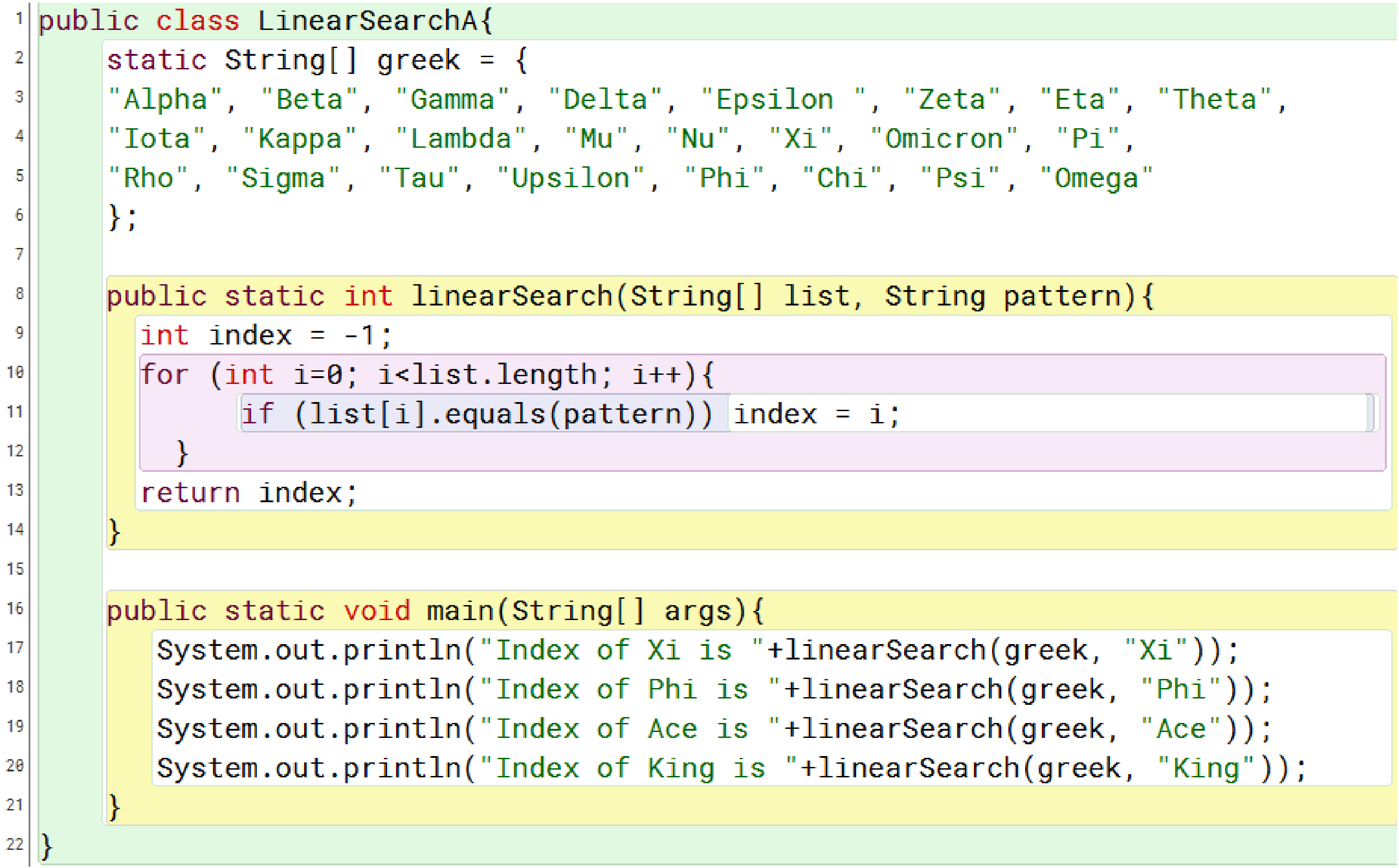
Can only enter input while your programming is running

SECTION 4

Sequential Algorithms

Look at every item in the list

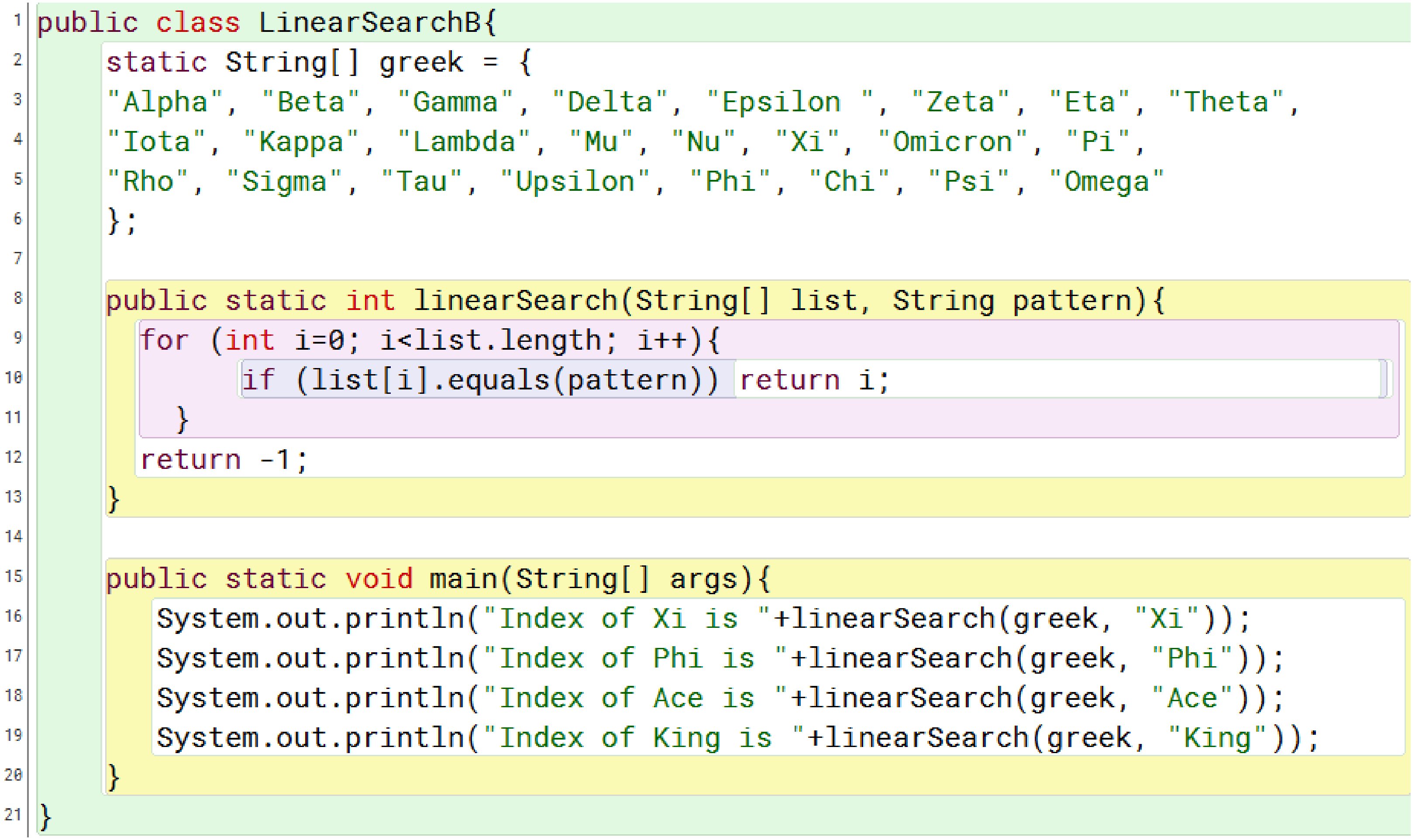
Demo Program: LinearSearchA.java



```java
public class LinearSearchA{
    static String[] greek = {
    "Alpha", "Beta", "Gamma", "Delta", "Epsilon ", "Zeta", "Eta", "Theta",
    "Iota", "Kappa", "Lambda", "Mu", "Nu", "Xi", "Omicron", "Pi",
    "Rho", "Sigma", "Tau", "Upsilon", "Phi", "Chi", "Psi", "Omega"
    };

    public static int linearSearch(String[] list, String pattern){
        int index = -1;
        for (int i=0; i<list.length; i++){
                if (list[i].equals(pattern)) index = i;
          }
        return index;
    }

    public static void main(String[] args){
         System.out.println("Index of Xi is "+linearSearch(greek, "Xi"));
         System.out.println("Index of Phi is "+linearSearch(greek, "Phi"));
         System.out.println("Index of Ace is "+linearSearch(greek, "Ace"));
         System.out.println("Index of King is "+linearSearch(greek, "King"));
    }
}
```

Stop looking if you find the search target

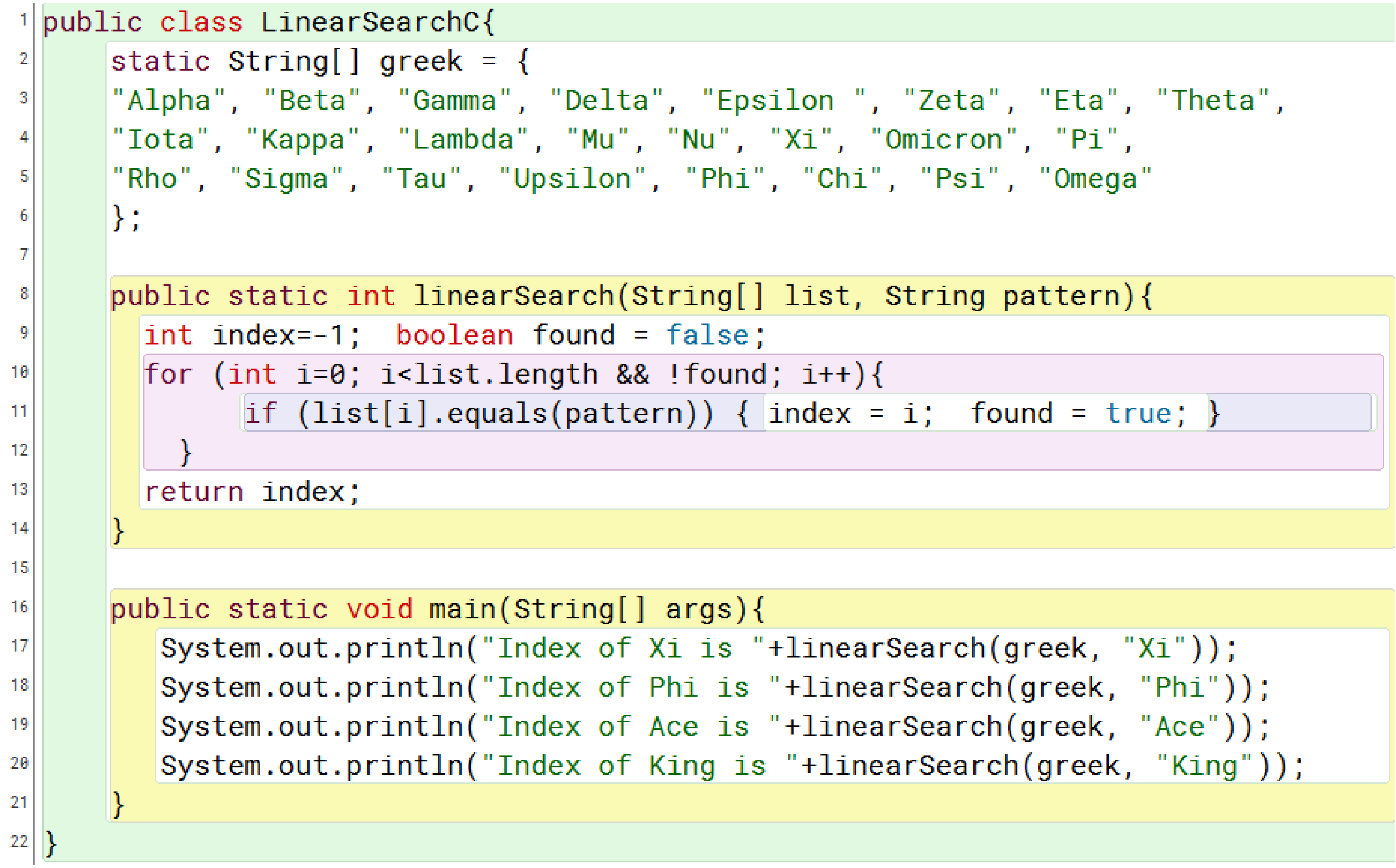
(Solution 1)



```
public class LinearSearchB{
    static String[] greek = {
    "Alpha", "Beta", "Gamma", "Delta", "Epsilon ", "Zeta", "Eta", "Theta",
    "Iota", "Kappa", "Lambda", "Mu", "Nu", "Xi", "Omicron", "Pi",
    "Rho", "Sigma", "Tau", "Upsilon", "Phi", "Chi", "Psi", "Omega"
    };

    public static int linearSearch(String[] list, String pattern){
      for (int i=0; i<list.length; i++){
            if (list[i].equals(pattern)) return i;
        }
      return -1;
    }

    public static void main(String[] args){
       System.out.println("Index of Xi is "+linearSearch(greek, "Xi"));
       System.out.println("Index of Phi is "+linearSearch(greek, "Phi"));
       System.out.println("Index of Ace is "+linearSearch(greek, "Ace"));
       System.out.println("Index of King is "+linearSearch(greek, "King"));
    }
}
```

Stop looking if you find the search target

(Solution 2)



```
public class LinearSearchC{
    static String[] greek = {
    "Alpha", "Beta", "Gamma", "Delta", "Epsilon ", "Zeta", "Eta", "Theta",
    "Iota", "Kappa", "Lambda", "Mu", "Nu", "Xi", "Omicron", "Pi",
    "Rho", "Sigma", "Tau", "Upsilon", "Phi", "Chi", "Psi", "Omega"
    };

    public static int linearSearch(String[] list, String pattern){
      int index=-1;  boolean found = false;
      for (int i=0; i<list.length && !found; i++){
             if (list[i].equals(pattern)) { index = i;  found = true; }
        }
      return index;
    }

    public static void main(String[] args){
       System.out.println("Index of Xi is "+linearSearch(greek, "Xi"));
       System.out.println("Index of Phi is "+linearSearch(greek, "Phi"));
       System.out.println("Index of Ace is "+linearSearch(greek, "Ace"));
       System.out.println("Index of King is "+linearSearch(greek, "King"));
    }
}
```

SECTION 5

Accumulate Algorithms

Accumulator

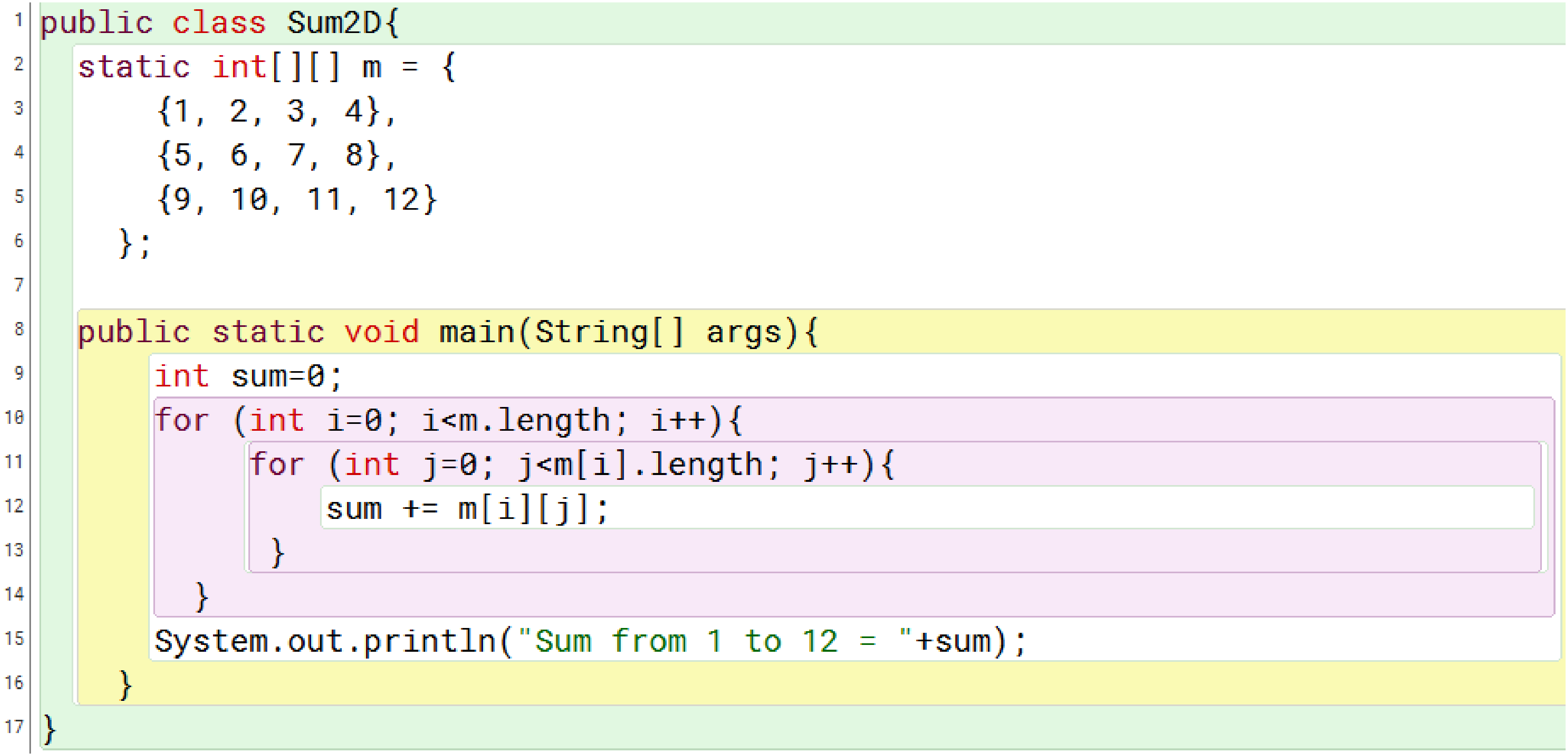
1. Traverse through an array and add(multiply) all elements of a list (array or arraylist) to a sum value (accumulator)

Application:

1. Sum or average value of a list.

2. Product of a list (n!, C(m, n))

Sum of 2D Array



```java
public class Sum2D{
 static int[][] m = {
   {1, 2, 3, 4},
   {5, 6, 7, 8},
   {9, 10, 11, 12}
  };

 public static void main(String[] args){
   int sum=0;
   for (int i=0; i<m.length; i++){
        for (int j=0; j<m[i].length; j++){
            sum += m[i][j];
         }
     }
   System.out.println("Sum from 1 to 12 = "+sum);
  }
}
```

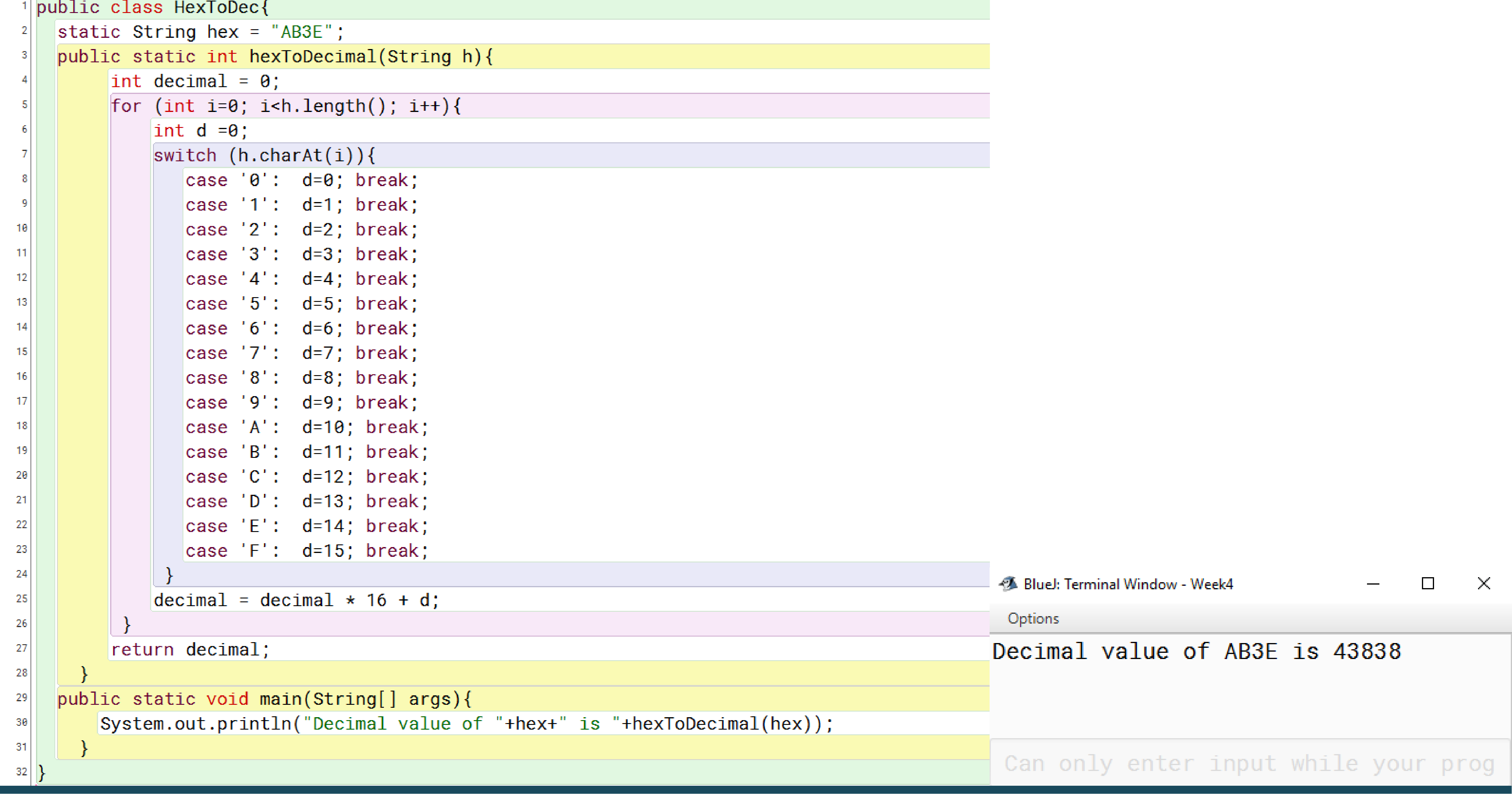
Accumulator with Shift and Add

Multiply and Add

Shift a value to the left(or right) and then add an new element to the right (or left)

Application:

1. Encoder
2. Hexadecimal to decimal
3. Reverse of decimal value



```java
public class HexToDec{
  static String hex = "AB3E";
  public static int hexToDecimal(String h){
        int decimal = 0;
        for (int i=0; i<h.length(); i++){
            int d =0;
            switch (h.charAt(i)){
               case '0':  d=0; break;
               case '1':  d=1; break;
               case '2':  d=2; break;
               case '3':  d=3; break;
               case '4':  d=4; break;
               case '5':  d=5; break;
               case '6':  d=6; break;
               case '7':  d=7; break;
               case '8':  d=8; break;
               case '9':  d=9; break;
               case 'A':  d=10; break;
               case 'B':  d=11; break;
               case 'C':  d=12; break;
               case 'D':  d=13; break;
               case 'E':  d=14; break;
               case 'F':  d=15; break;
             }
            decimal = decimal * 16 + d;
         }
        return decimal;
    }
  public static void main(String[] args){
      System.out.println("Decimal value of "+hex+" is "+hexToDecimal(hex));
    }
}
```

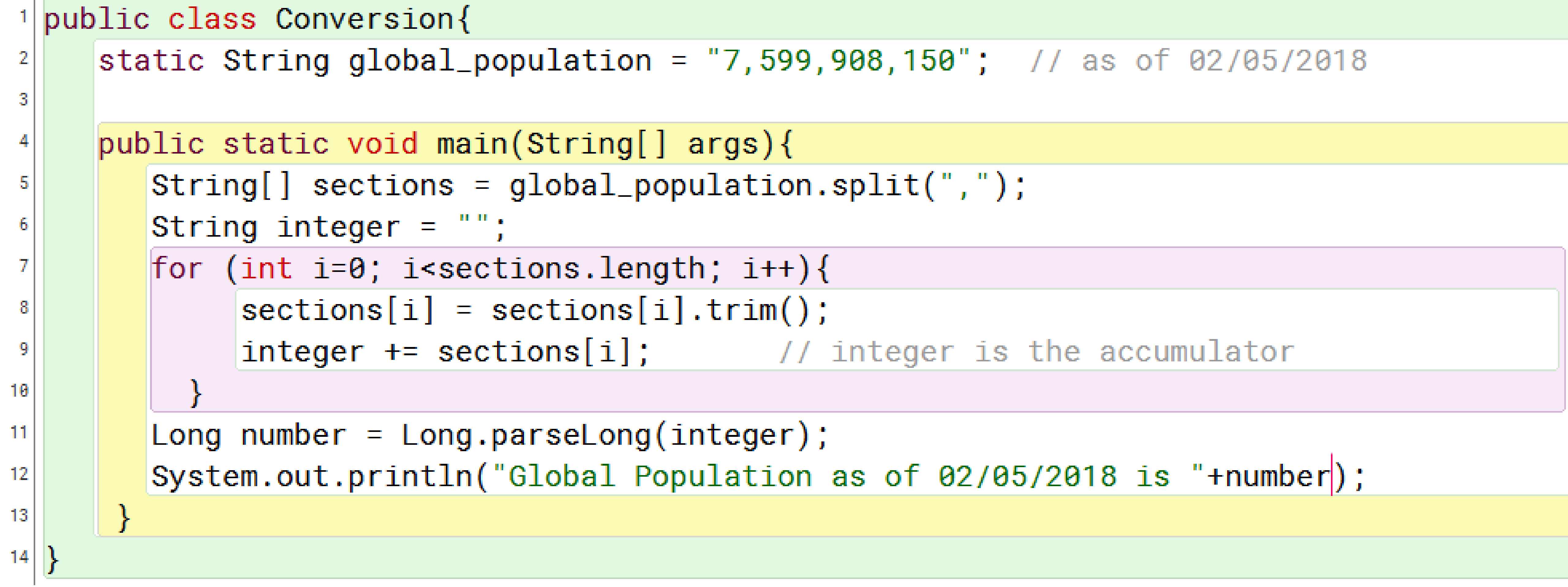
```
Decimal value of AB3E is 43838
```

Accumulator by Appending

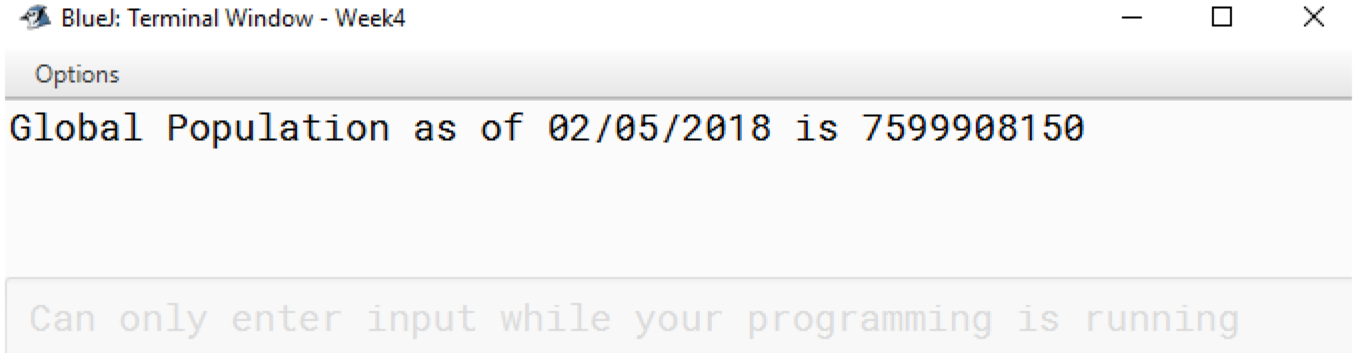
Accumulator using string

Application:

1. Reverse of String
2. Conversion of a number with “,” to integer



```java
public class Conversion{
    static String global_population = "7,599,908,150";  // as of 02/05/2018

    public static void main(String[] args){
        String[] sections = global_population.split(",");
        String integer = "";
        for (int i=0; i<sections.length; i++){
            sections[i] = sections[i].trim();
            integer += sections[i];       // integer is the accumulator
        }
        Long number = Long.parseLong(integer);
        System.out.println("Global Population as of 02/05/2018 is "+number);
    }
}
```



BlueJ: Terminal Window - Week4
Options
Can only enter input while your programming is running

```
Global Population as of 02/05/2018 is 7599908150
```

SECTION 6

Min/Max

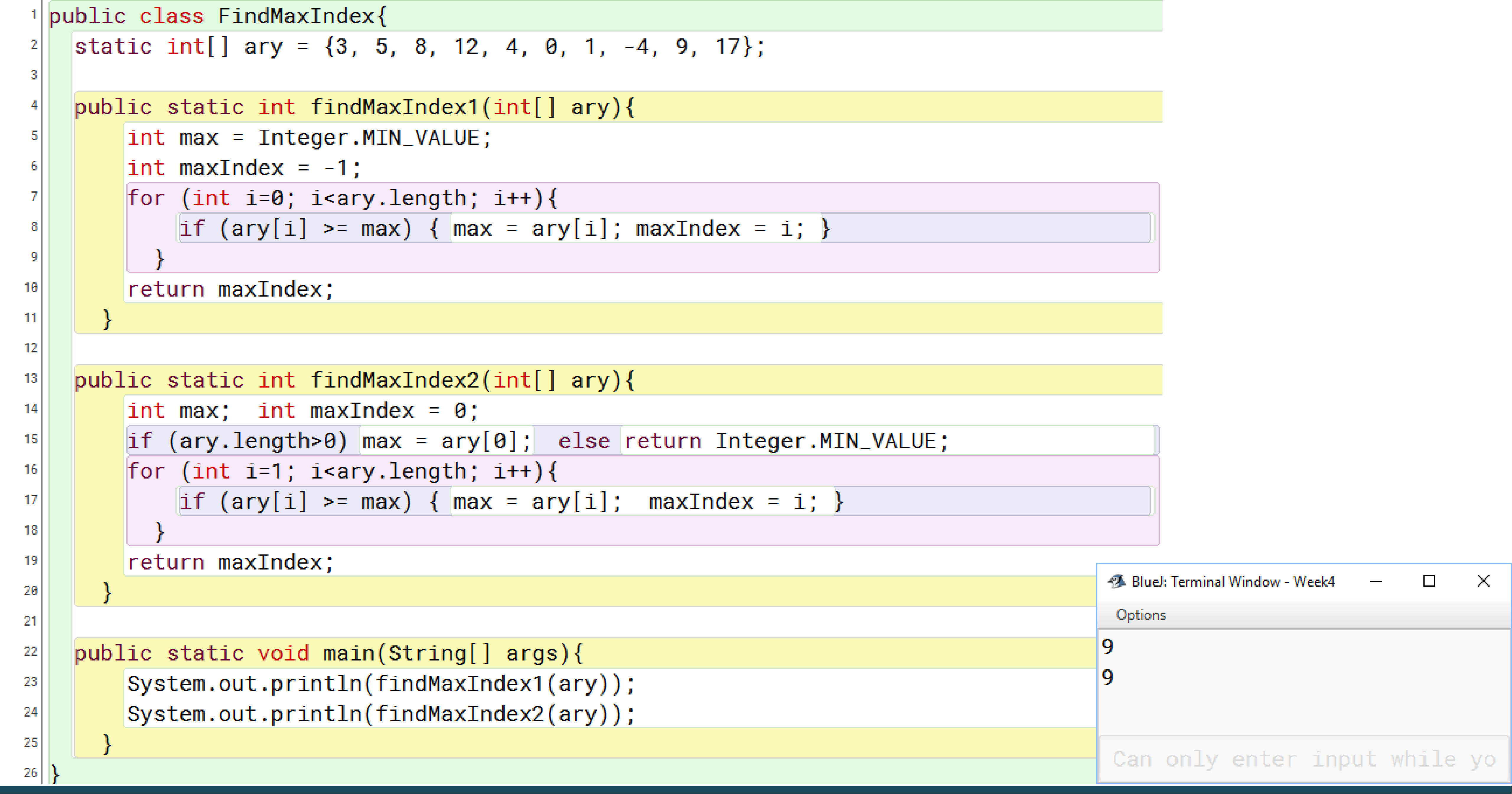
Two ways of Finding Maximum

- Using the low value (Integer.MIN_VALUE, Integer.MAX_VALUE)
- Using the first element of the list.

See Week 2 Program FindMax.java

Finding the MaxIndex

- Just keep track of another integer for the index of the maximum value.



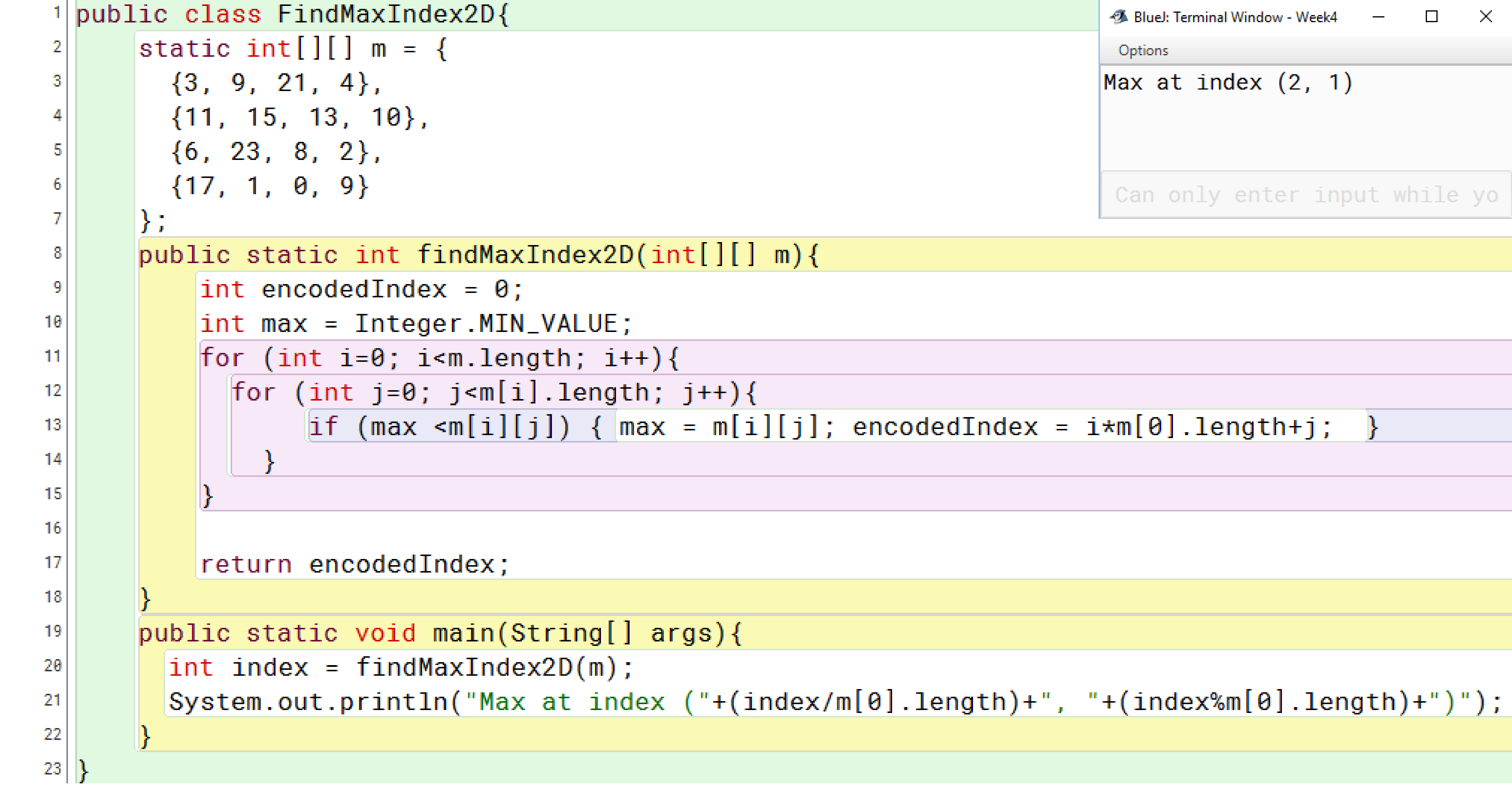
```java
public class FindMaxIndex{
 static int[] ary = {3, 5, 8, 12, 4, 0, 1, -4, 9, 17};

 public static int findMaxIndex1(int[] ary){
   int max = Integer.MIN_VALUE;
   int maxIndex = -1;
   for (int i=0; i<ary.length; i++){
     if (ary[i] >= max) { max = ary[i]; maxIndex = i; }
    }
   return maxIndex;
  }

 public static int findMaxIndex2(int[] ary){
   int max;  int maxIndex = 0;
   if (ary.length>0) max = ary[0];  else return Integer.MIN_VALUE;
   for (int i=1; i<ary.length; i++){
     if (ary[i] >= max) { max = ary[i];  maxIndex = i; }
    }
   return maxIndex;
  }

 public static void main(String[] args){
   System.out.println(findMaxIndex1(ary));
   System.out.println(findMaxIndex2(ary));
  }
}
```

```
9
9

Can only enter input while yo
```

Finding the Encoded 2D Index for the Maximum Value

- The encoded 2D index

```
encodedIndex = i * number_of_cols + j;
```

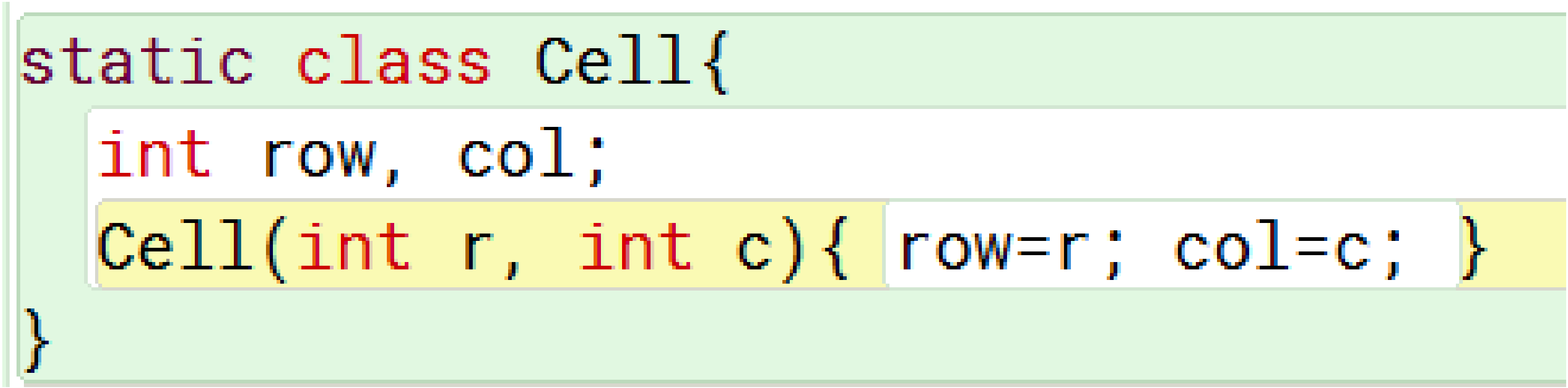


```java
public class FindMaxIndex2D{
    static int[][] m = {
      {3, 9, 21, 4},
      {11, 15, 13, 10},
      {6, 23, 8, 2},
      {17, 1, 0, 9}
    };
    public static int findMaxIndex2D(int[][] m){
        int encodedIndex = 0;
        int max = Integer.MIN_VALUE;
        for (int i=0; i<m.length; i++){
          for (int j=0; j<m[i].length; j++){
               if (max <m[i][j]) { max = m[i][j]; encodedIndex = i*m[0].length+j;  }
            }
        }

        return encodedIndex;
    }
    public static void main(String[] args){
      int index = findMaxIndex2D(m);
      System.out.println("Max at index ("+(index/m[0].length)+", "+(index%m[0].length)+")");
    }
}
```

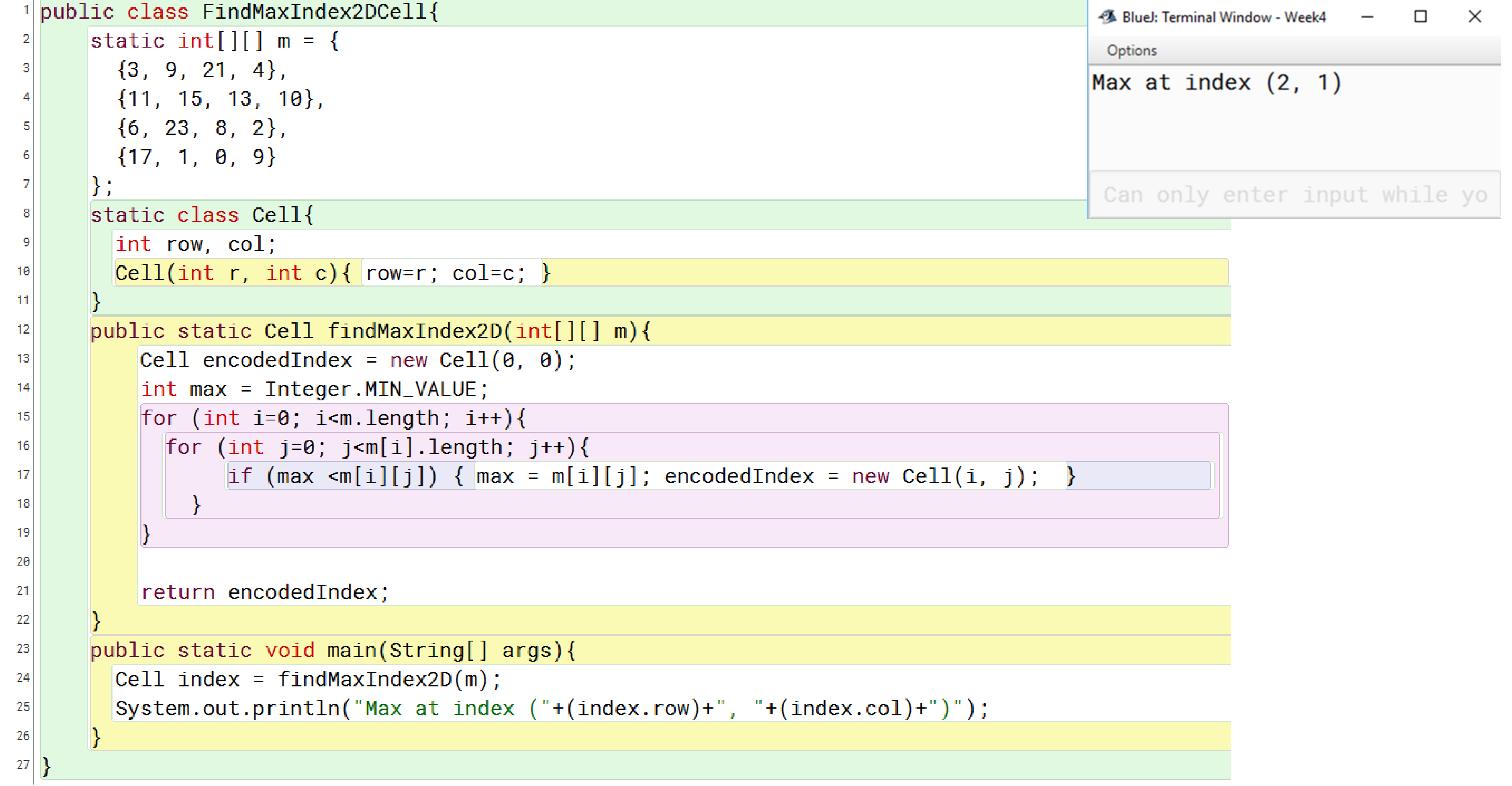
```
Max at index (2, 1)
```

Finding the 2D Cell Object with Maximum Value

- Create an easy 2D (row, col) object as a structure (record) as return data type.



```
static class Cell{
   int row, col;
   Cell(int r, int c){ row=r; col=c; }
}
```



```java
public class FindMaxIndex2DCell{
    static int[][] m = {
      {3, 9, 21, 4},
      {11, 15, 13, 10},
      {6, 23, 8, 2},
      {17, 1, 0, 9}
    };
    static class Cell{
      int row, col;
      Cell(int r, int c){ row=r; col=c; }
    }
    public static Cell findMaxIndex2D(int[][] m){
        Cell encodedIndex = new Cell(0, 0);
        int max = Integer.MIN_VALUE;
        for (int i=0; i<m.length; i++){
          for (int j=0; j<m[i].length; j++){
               if (max <m[i][j]) { max = m[i][j]; encodedIndex = new Cell(i, j);   }
            }
        }

        return encodedIndex;
    }
    public static void main(String[] args){
      Cell index = findMaxIndex2D(m);
      System.out.println("Max at index ("+(index.row)+", "+(index.col)+")");
    }
}
```

```
Max at index (2, 1)
```

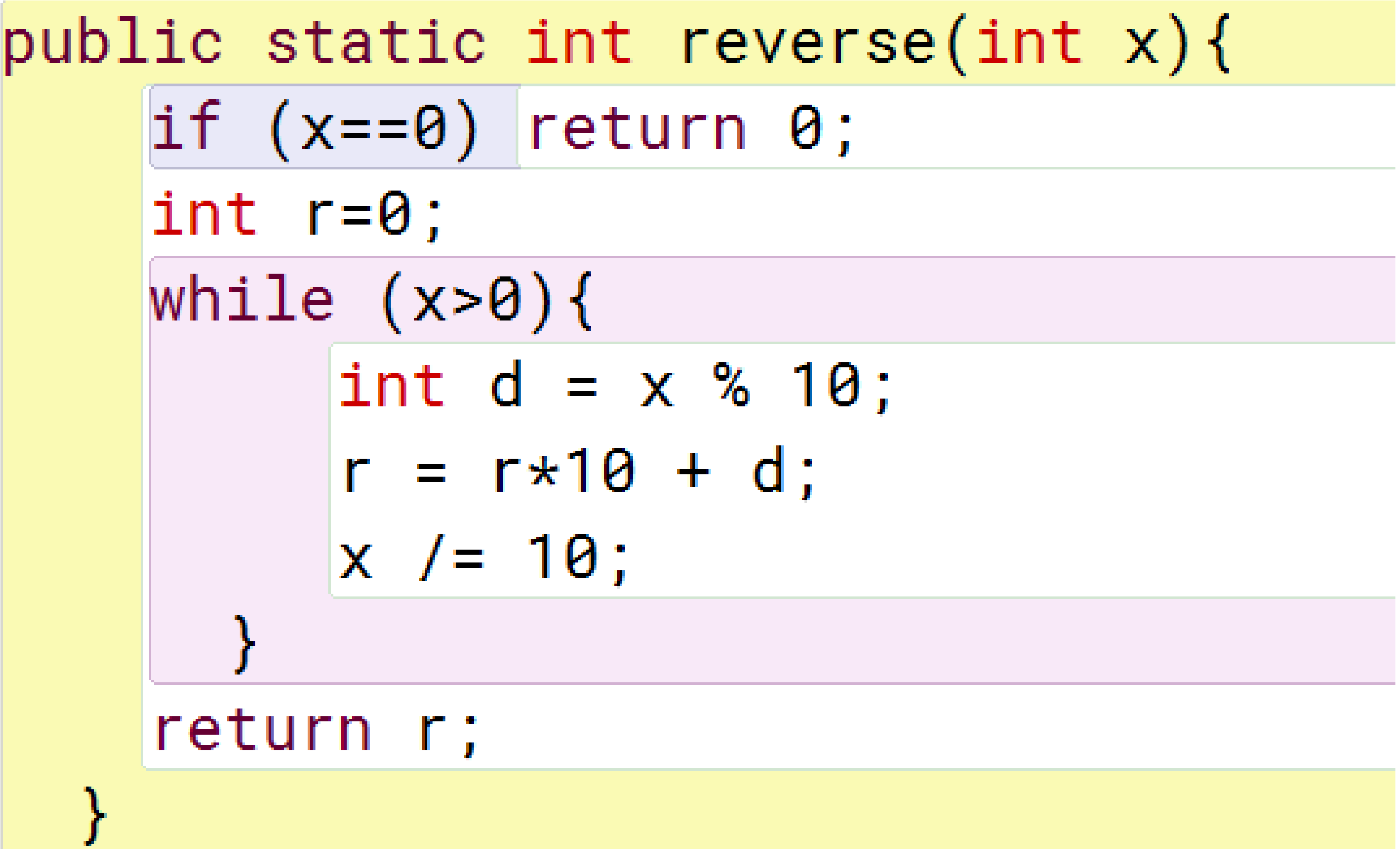
SECTION 7

Number Systems

Questions

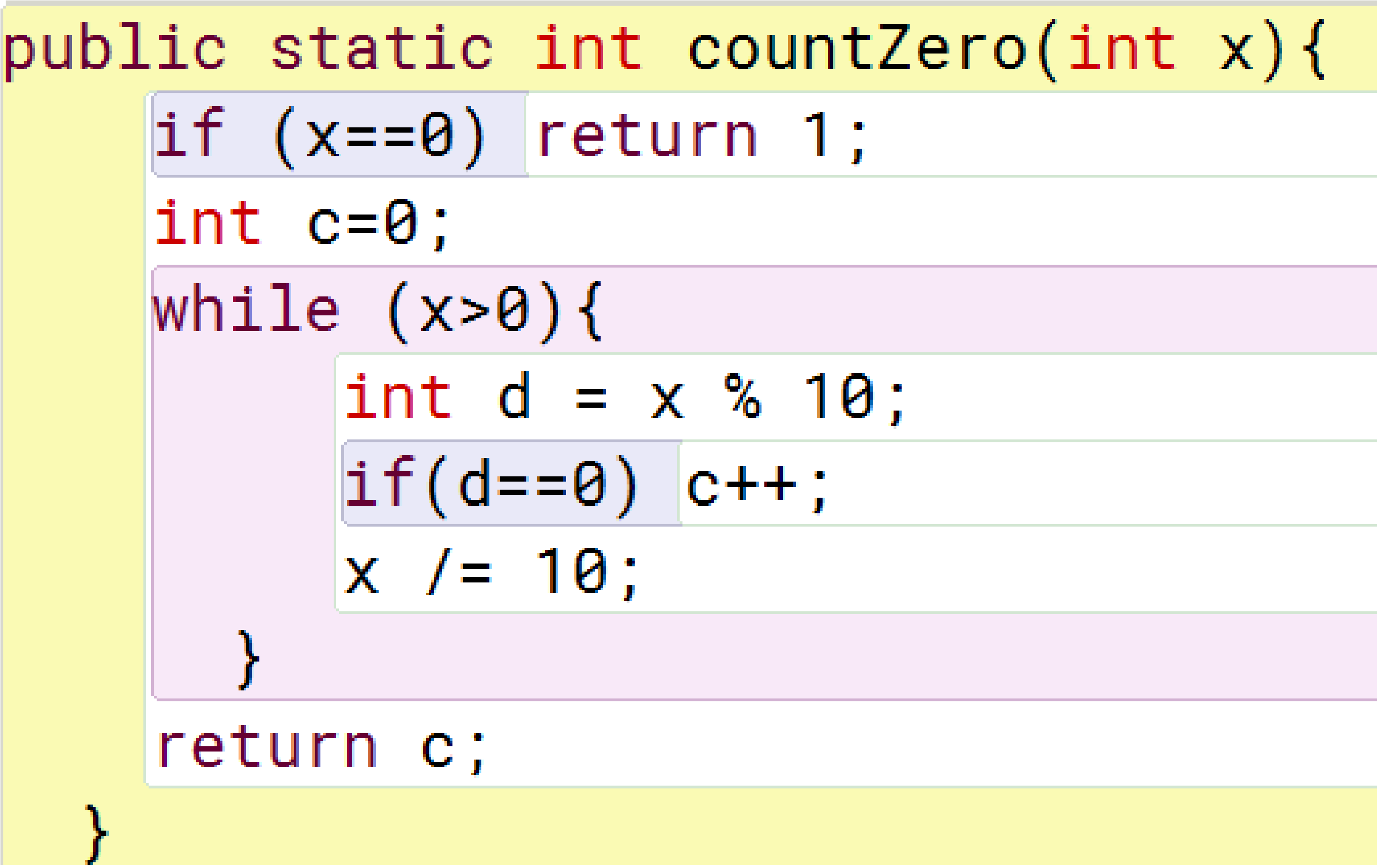
AP2017 – Q1(A)

C1(a) Reverse of Digits



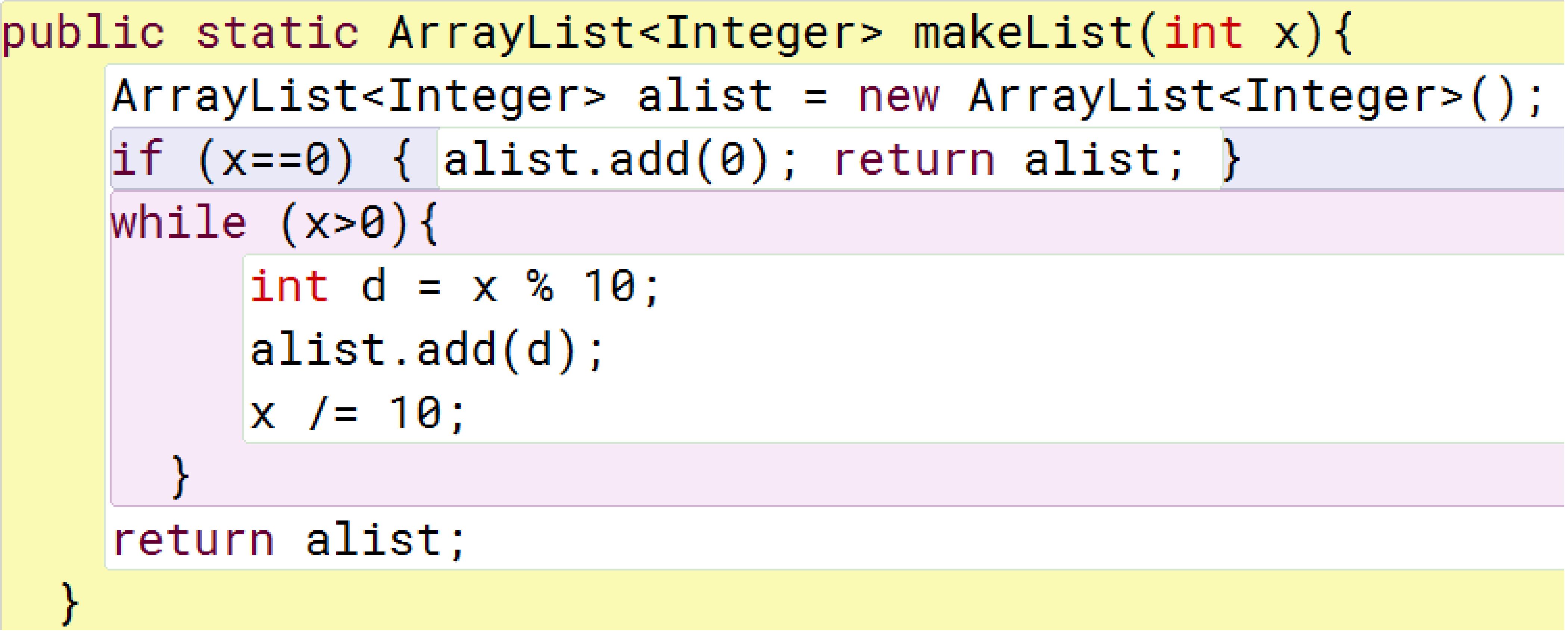
```java
public static int reverse(int x){
    if (x==0) return 0;
    int r=0;
    while (x>0){
        int d = x % 10;
        r = r*10 + d;
        x /= 10;
      }
    return r;
  }
```

C1(b) Number of Zero Digits



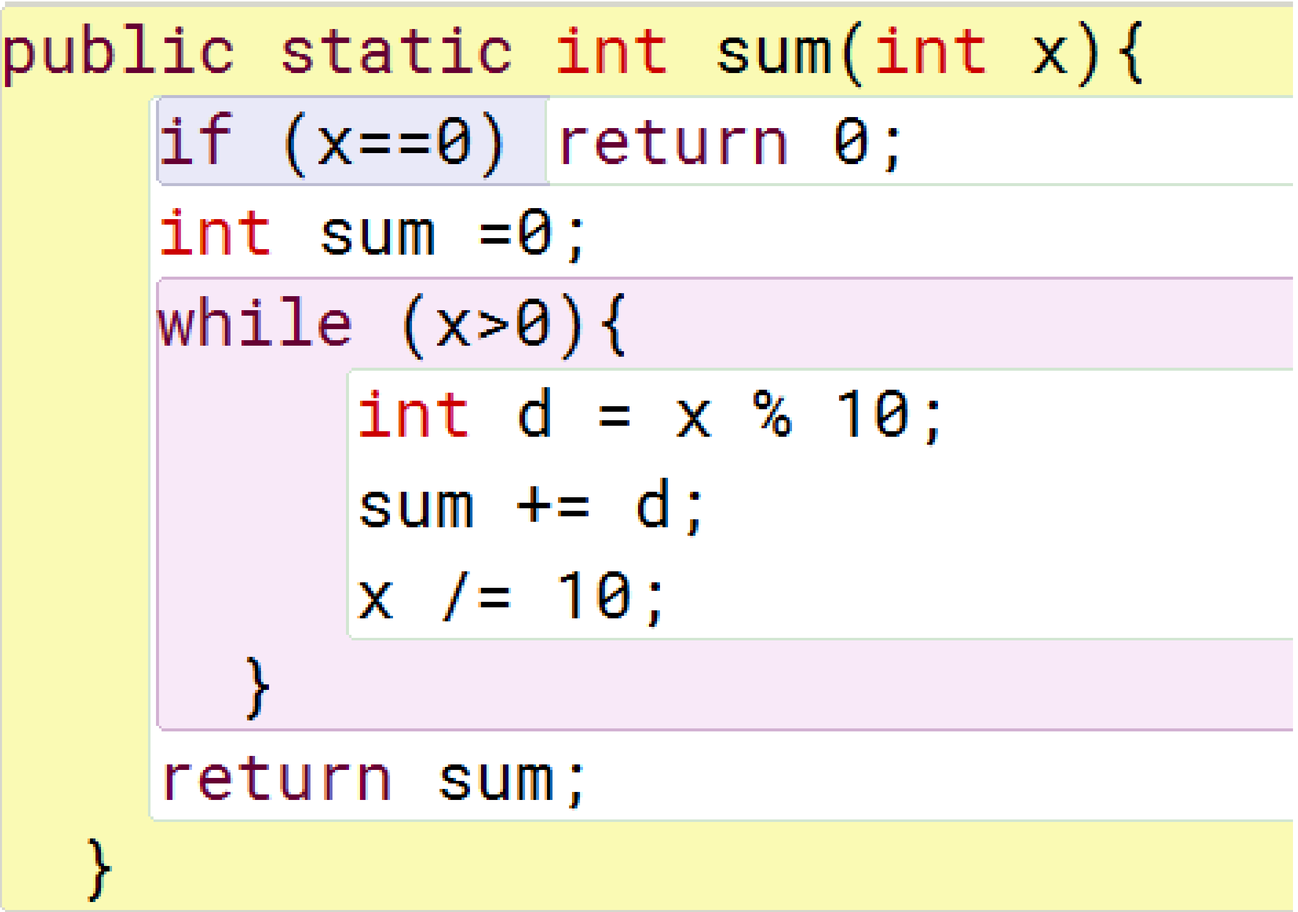
```
public static int countZero(int x){
    if (x==0) return 1;
    int c=0;
    while (x>0){
        int d = x % 10;
        if(d==0) c++;
        x /= 10;
      }
    return c;
  }
```

C1(c) Digit List

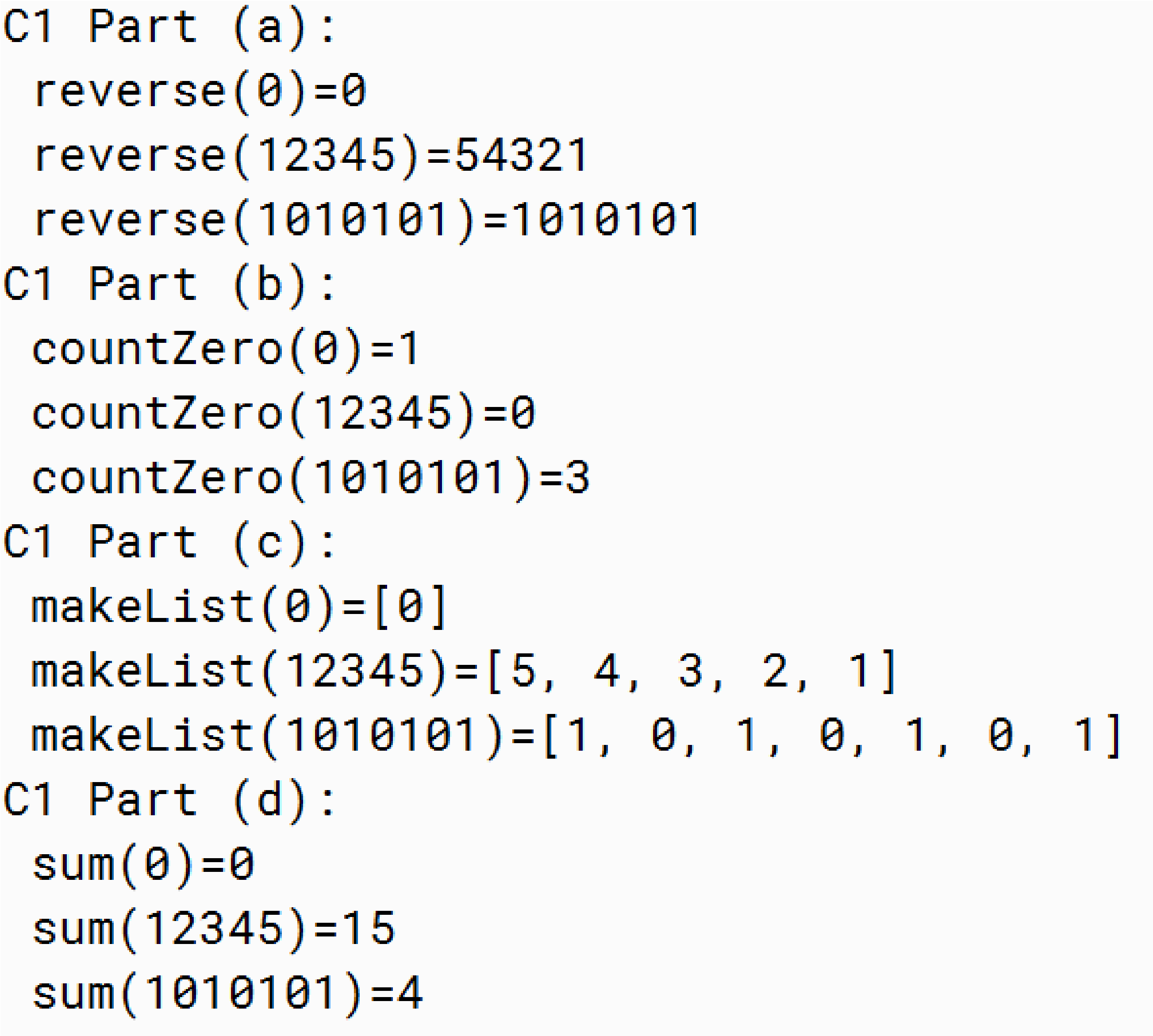


```
public static ArrayList<Integer> makeList(int x){
    ArrayList<Integer> alist = new ArrayList<Integer>();
    if (x==0) { alist.add(0); return alist; }
    while (x>0){
        int d = x % 10;
        alist.add(d);
        x /= 10;
     }
    return alist;
  }
```

C1(d) Sum of Digits



```
public static int sum(int x){
    if (x==0) return 0;
    int sum =0;
    while (x>0){
        int d = x % 10;
        sum += d;
        x /= 10;
    }
    return sum;
}
```



```
C1 Part (a):
 reverse(0)=0
 reverse(12345)=54321
 reverse(1010101)=1010101
C1 Part (b):
 countZero(0)=1
 countZero(12345)=0
 countZero(1010101)=3
C1 Part (c):
 makeList(0)=[0]
 makeList(12345)=[5, 4, 3, 2, 1]
 makeList(1010101)=[1, 0, 1, 0, 1, 0, 1]
C1 Part (d):
 sum(0)=0
 sum(12345)=15
 sum(1010101)=4
```

Key Points

- Take good care of **Zeros**! Zeros may need special care.
- Be aware of the direction (ascending or descending)

SECTION 8

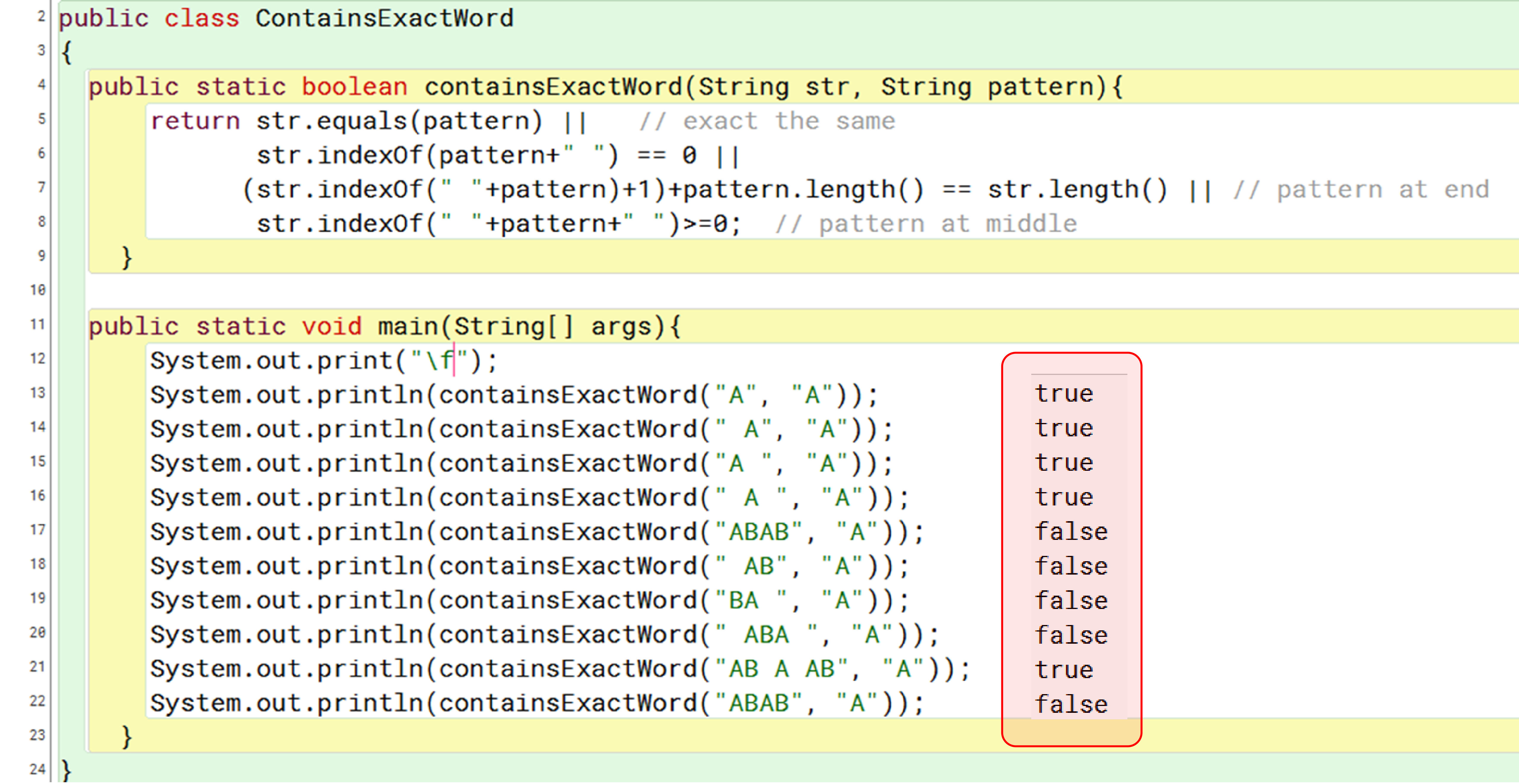
ContainsExactWord()

Questions

AP 2016 – Q2(B)

Observation

- str.equals(pattern);   // exact the same
- str.indexOf(pattern+” “) == 0;
- (str.indexOf(“ ”+pattern)+1)+pattern.length() == str.length(); // pattern at end
- str.indexOf(pattern)>0; // pattern at middle

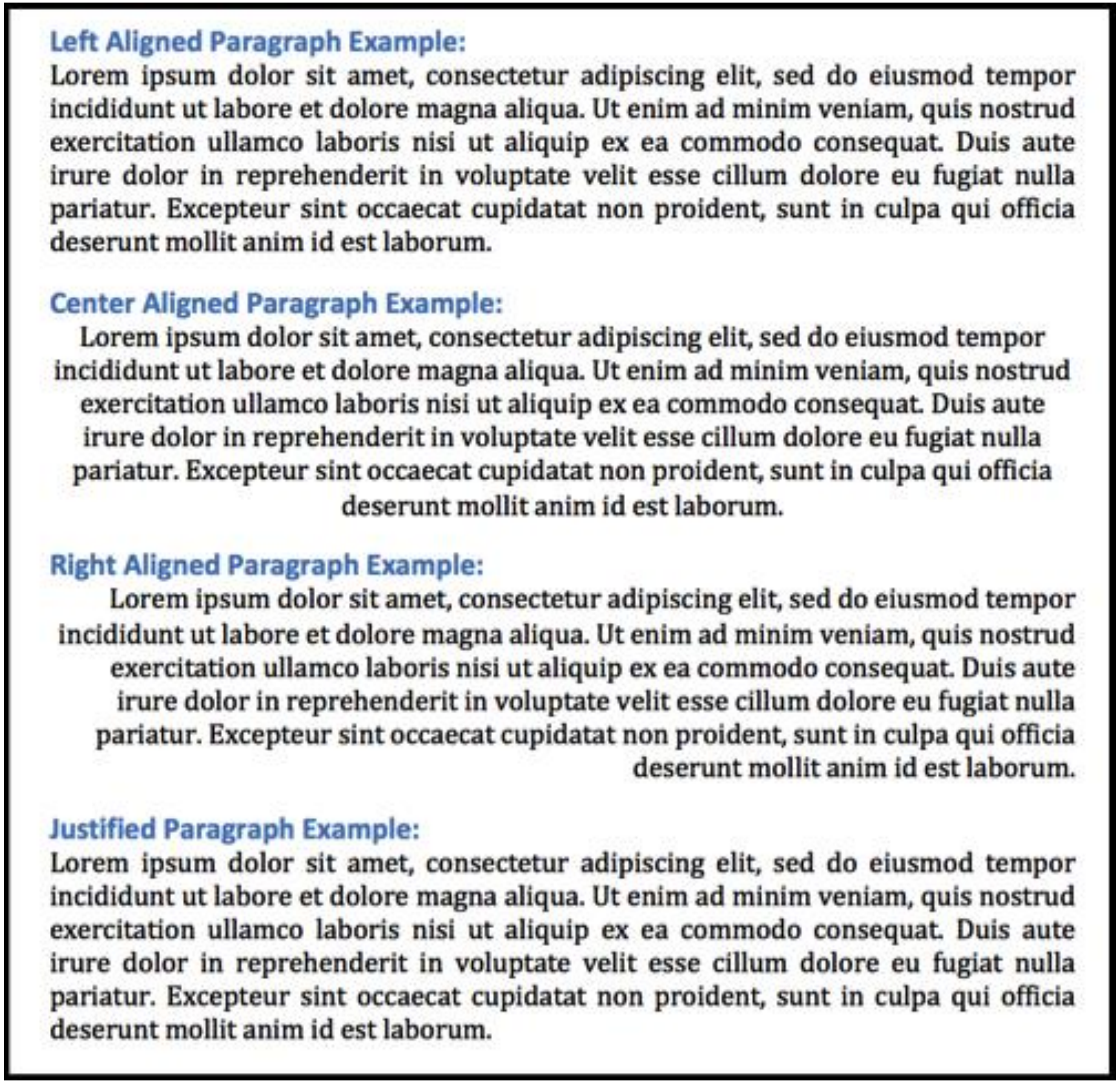


```java
public class ContainsExactWord
{
  public static boolean containsExactWord(String str, String pattern){
      return str.equals(pattern) ||   // exact the same
             str.indexOf(pattern+" ") == 0 ||
            (str.indexOf(" "+pattern)+1)+pattern.length() == str.length() || // pattern at end
             str.indexOf(" "+pattern+" ")>=0;  // pattern at middle
  }

  public static void main(String[] args){
      System.out.print("\f");
      System.out.println(containsExactWord("A", "A"));
      System.out.println(containsExactWord(" A", "A"));
      System.out.println(containsExactWord("A ", "A"));
      System.out.println(containsExactWord(" A ", "A"));
      System.out.println(containsExactWord("ABAB", "A"));
      System.out.println(containsExactWord(" AB", "A"));
      System.out.println(containsExactWord("BA ", "A"));
      System.out.println(containsExactWord(" ABA ", "A"));
      System.out.println(containsExactWord("AB A AB", "A"));
      System.out.println(containsExactWord("ABAB", "A"));
  }
}
```

```
true
true
true
true
false
false
false
false
true
false
```

SECTION 9

String Formatting



Left Aligned Paragraph Example:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Center Aligned Paragraph Example:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Right Aligned Paragraph Example:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Justified Paragraph Example:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Given a formatLength

Given a line of text which is shorter than the formatLength (line length).

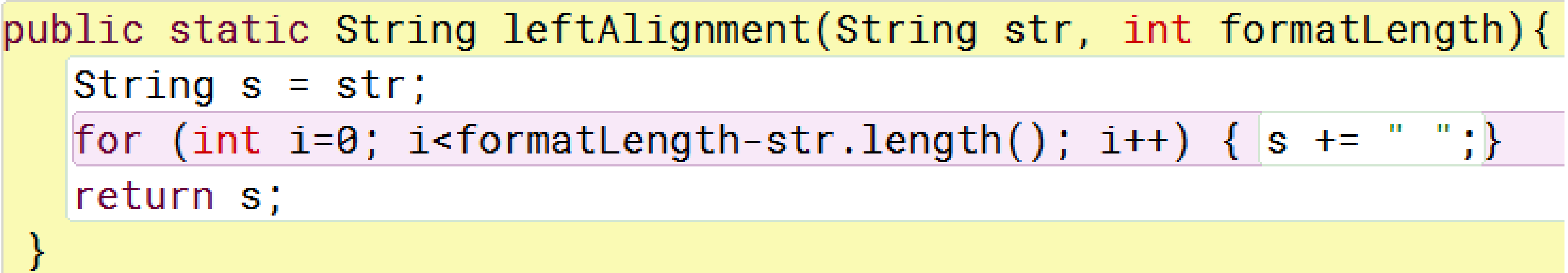
Adjust the text the following 4 formats:

- Left-Alignment
- Right-Alignment
- Centered
- Justified

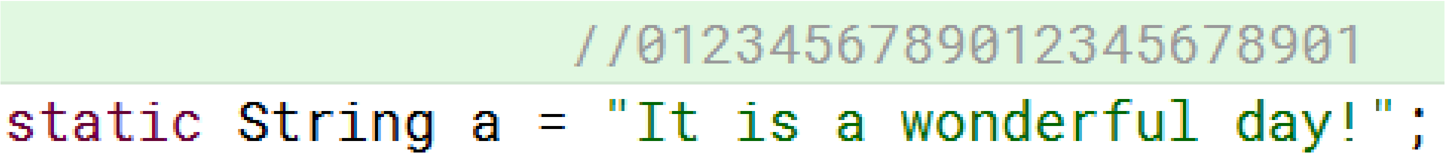
Questions

AP 2016-Q4

Left Alignment

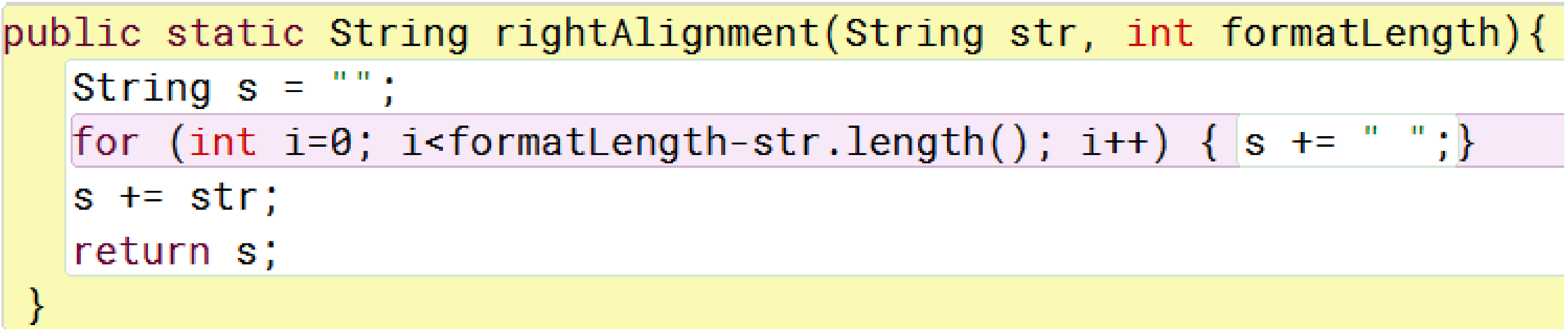


```
public static String leftAlignment(String str, int formatLength){
    String s = str;
    for (int i=0; i<formatLength-str.length(); i++) { s += " ";}
    return s;
 }
```



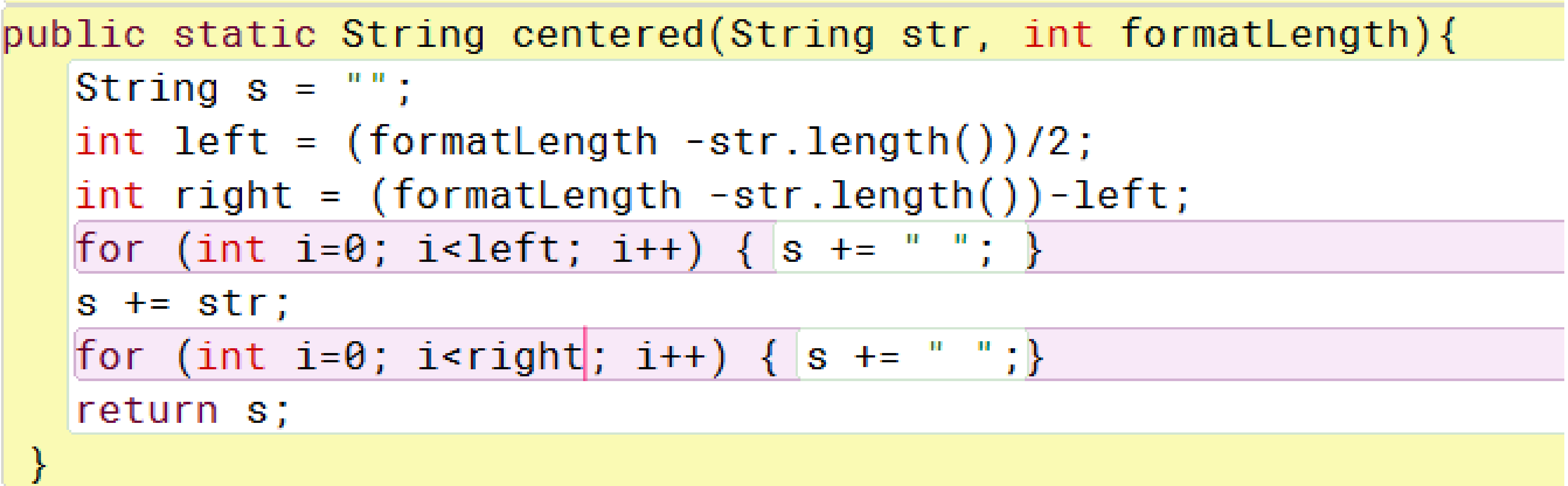
```
                 //0123456789012345678901
static String a = "It is a wonderful day!";
```

Right Alignment

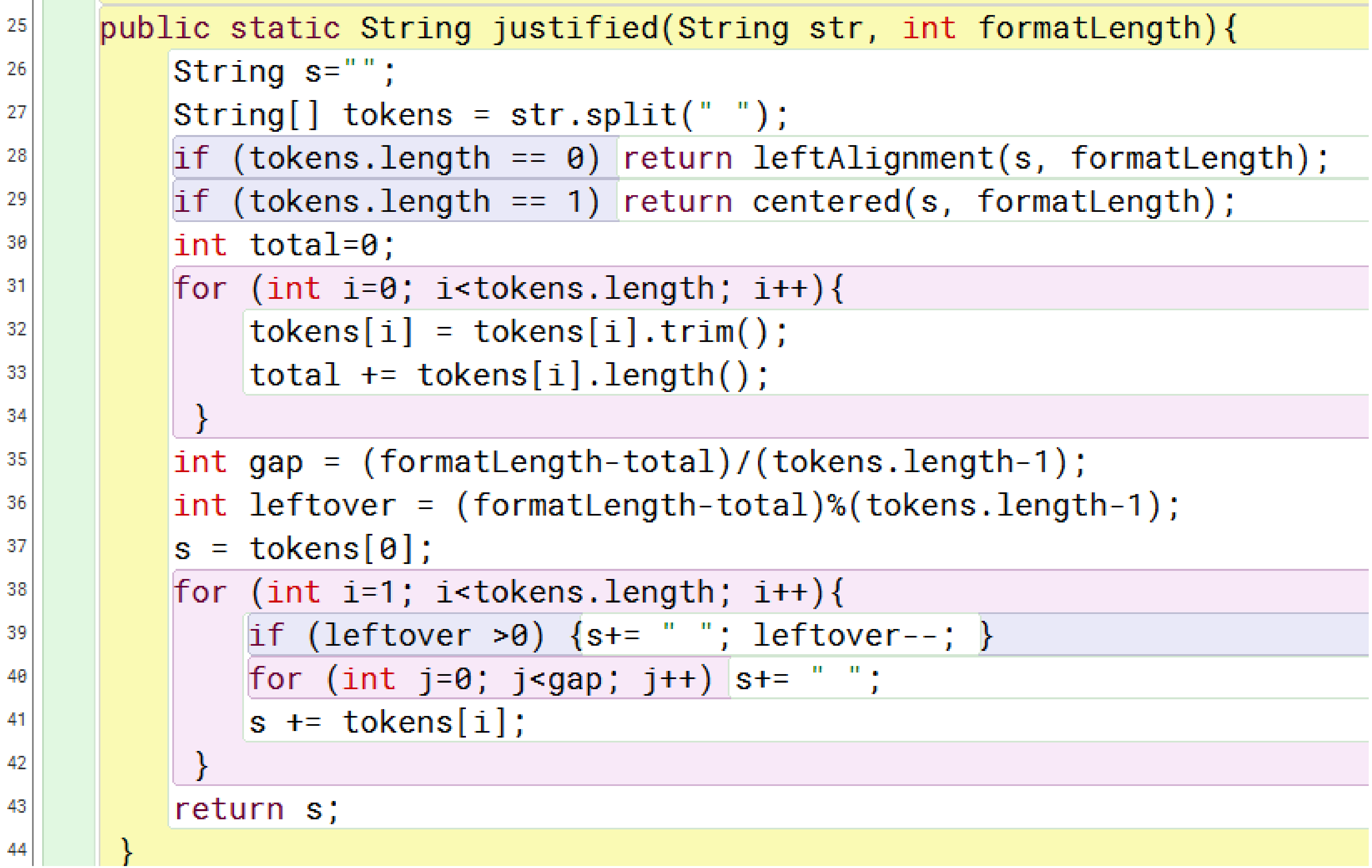


```java
public static String rightAlignment(String str, int formatLength){
    String s = "";
    for (int i=0; i<formatLength-str.length(); i++) { s += " ";}
    s += str;
    return s;
 }
```

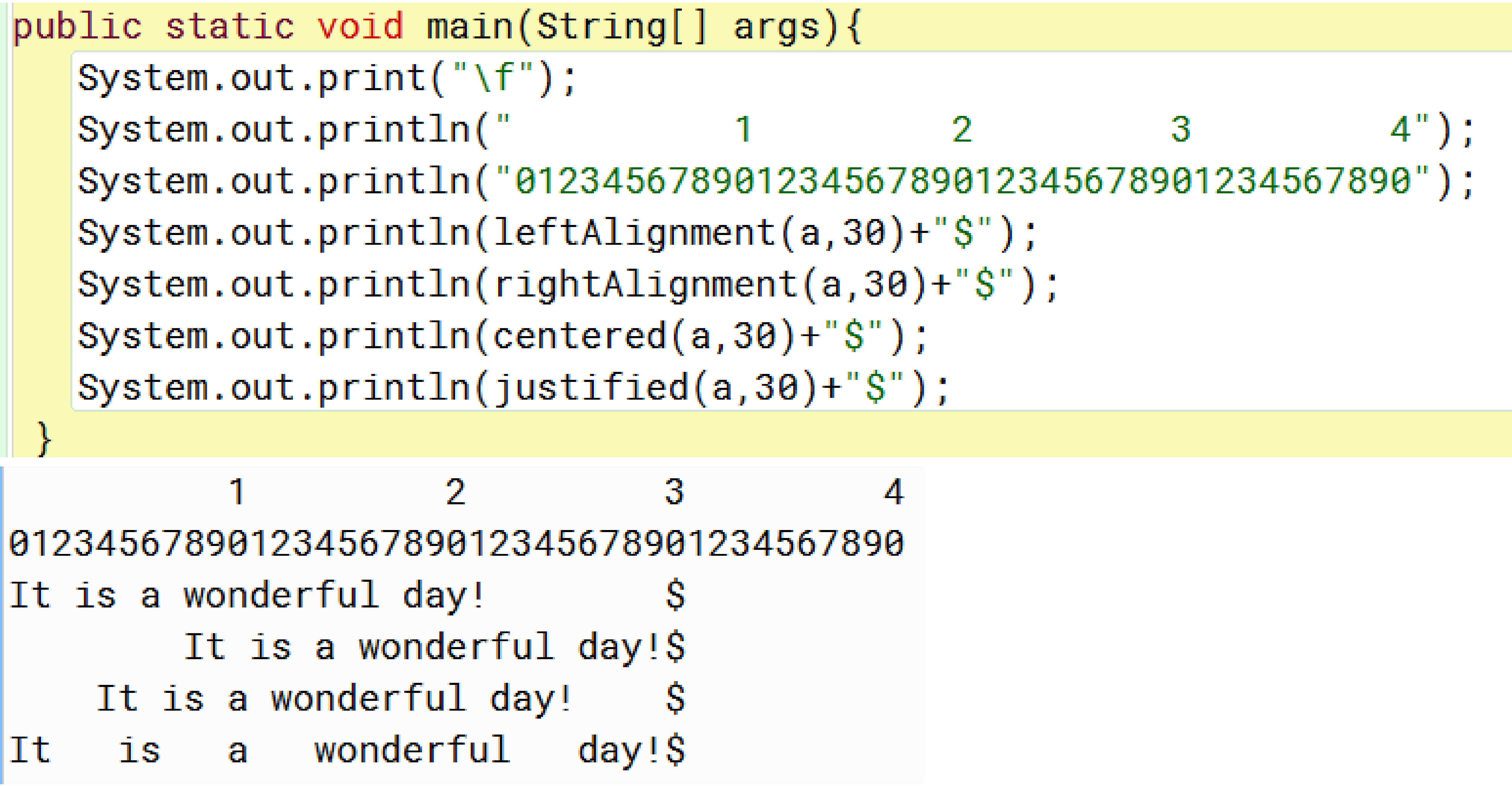
Centered



```
public static String centered(String str, int formatLength){
    String s = "";
    int left = (formatLength -str.length())/2;
    int right = (formatLength -str.length())-left;
    for (int i=0; i<left; i++) { s += " "; }
    s += str;
    for (int i=0; i<right; i++) { s += " ";}
    return s;
 }
```



```java
public static String justified(String str, int formatLength){
    String s="";
    String[] tokens = str.split(" ");
    if (tokens.length == 0) return leftAlignment(s, formatLength);
    if (tokens.length == 1) return centered(s, formatLength);
    int total=0;
    for (int i=0; i<tokens.length; i++){
        tokens[i] = tokens[i].trim();
        total += tokens[i].length();
     }
    int gap = (formatLength-total)/(tokens.length-1);
    int leftover = (formatLength-total)%(tokens.length-1);
    s = tokens[0];
    for (int i=1; i<tokens.length; i++){
        if (leftover >0) {s+= " "; leftover--; }
        for (int j=0; j<gap; j++) s+= " ";
        s += tokens[i];
     }
    return s;
 }
```



```
public static void main(String[] args){
    System.out.print("\f");
    System.out.println("          1         2         3         4");
    System.out.println("01234567890123456789012345678901234567890");
    System.out.println(leftAlignment(a,30)+"$");
    System.out.println(rightAlignment(a,30)+"$");
    System.out.println(centered(a,30)+"$");
    System.out.println(justified(a,30)+"$");
 }
```

```
          1         2         3         4
01234567890123456789012345678901234567890
It is a wonderful day!        $
        It is a wonderful day!$
    It is a wonderful day!    $
It   is   a   wonderful   day!$
```

SECTION 10

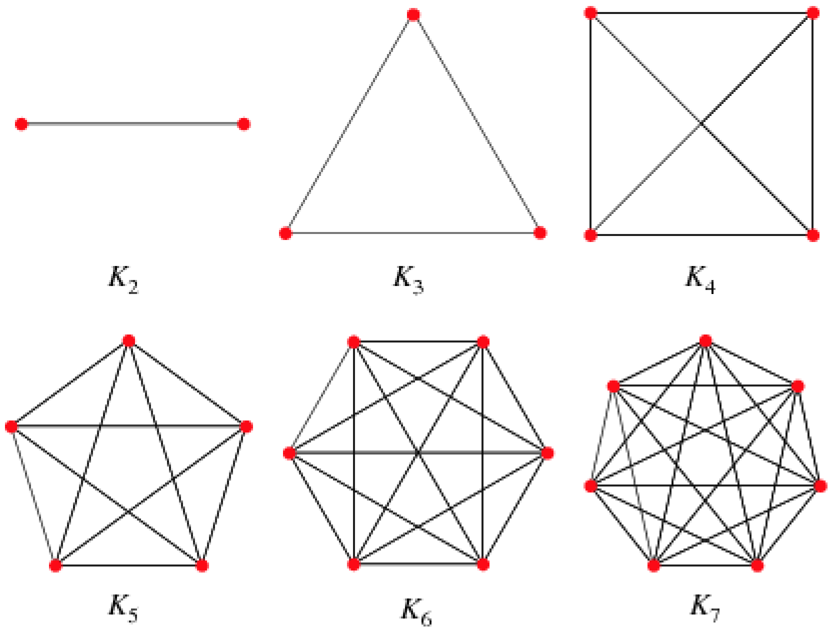
K-Graph

Questions

AP2015-Q1(A, B, C)

Mutual Relationship

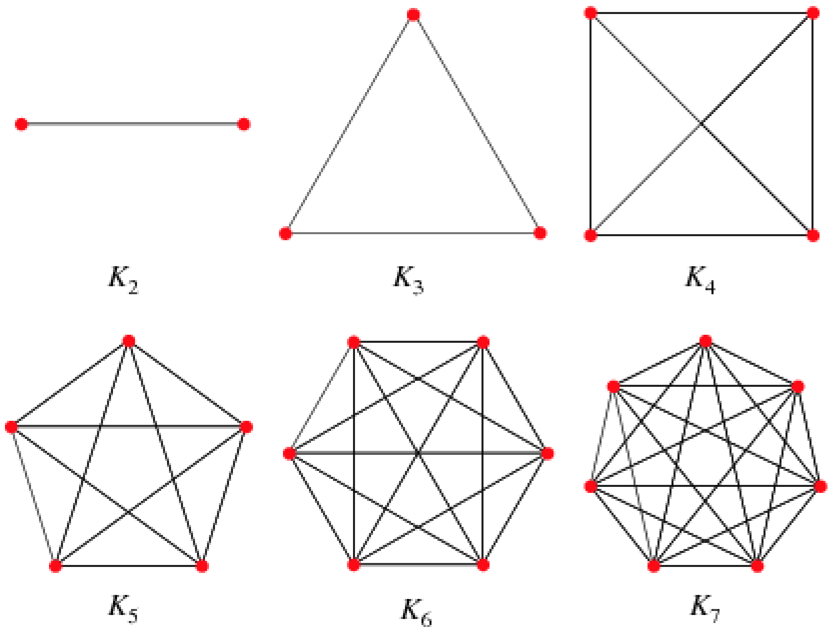
- Equality (Equality Check)
- Shortest Distance (Finding Max/Min)
- Inverse Order Count (Comparison)
- Friendships (Two-way checking)
- Matrix Transpose (Swap)



```
for (int i=1; i< arr.length; i++){
        for (int j=0; j<i; j++){
            cell[i][j] = ‘*’;
            /* for all solution space
               (i, j) */
        }
    }
```

K-graph is for C(n, 2).

	0	**1**	**2**
0			
1	*		
2	*	*	



```
for (int i=0; i< arr.length-1; i++){
        for (int j=i+1; j<arr.length; j++){
            cell[i][j] = '*';
            /* for all solution space
                (i, j) */
        }
    }
```

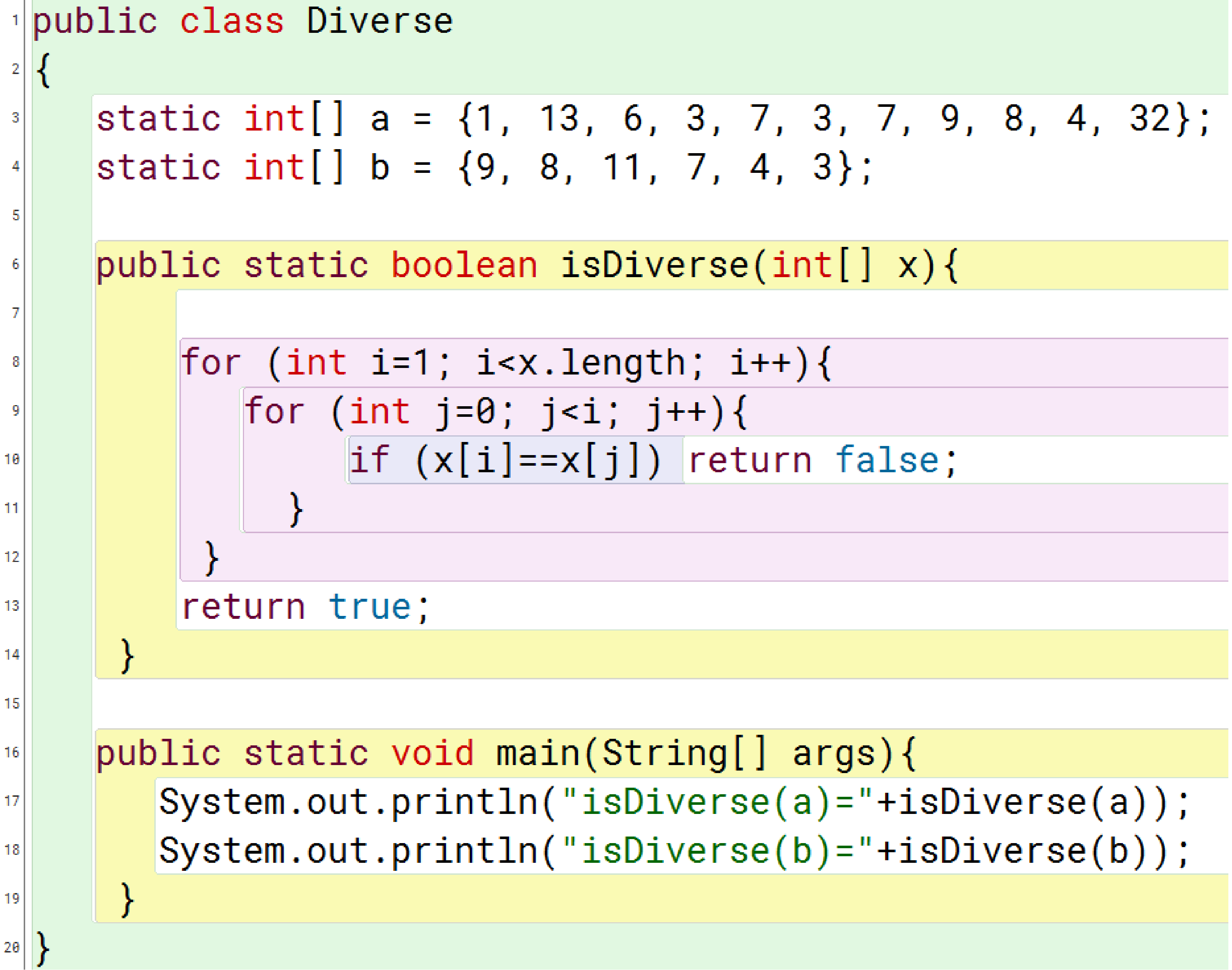
K-graph is for C(n, 2).

	0	**1**	**2**
0		*	*
1			*
2			

Diversity

Check if a group of n numbers are diverse

- A group of number are called diverse if none of its members repeats.
- Use partial 2-D traversal for all one-to-one mapping from one member to another. If there is any equality, then return false. Otherwise, if there is no identical matching members, we return true.

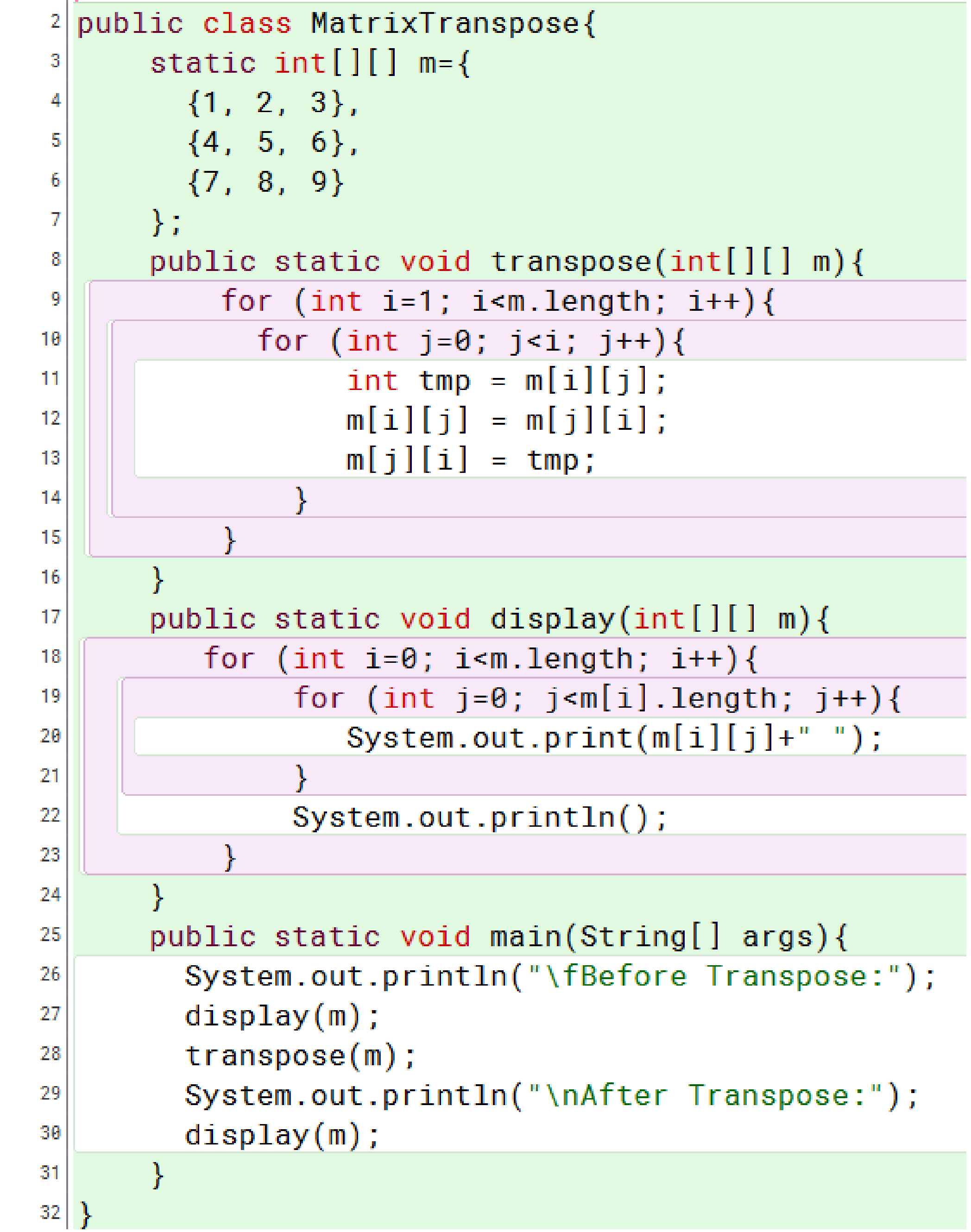


```
public class Diverse
{
    static int[] a = {1, 13, 6, 3, 7, 3, 7, 9, 8, 4, 32};
    static int[] b = {9, 8, 11, 7, 4, 3};

    public static boolean isDiverse(int[] x){

        for (int i=1; i<x.length; i++){
            for (int j=0; j<i; j++){
                if (x[i]==x[j]) return false;
            }
        }
        return true;
    }

    public static void main(String[] args){
        System.out.println("isDiverse(a)="+isDiverse(a));
        System.out.println("isDiverse(b)="+isDiverse(b));
    }
}
```

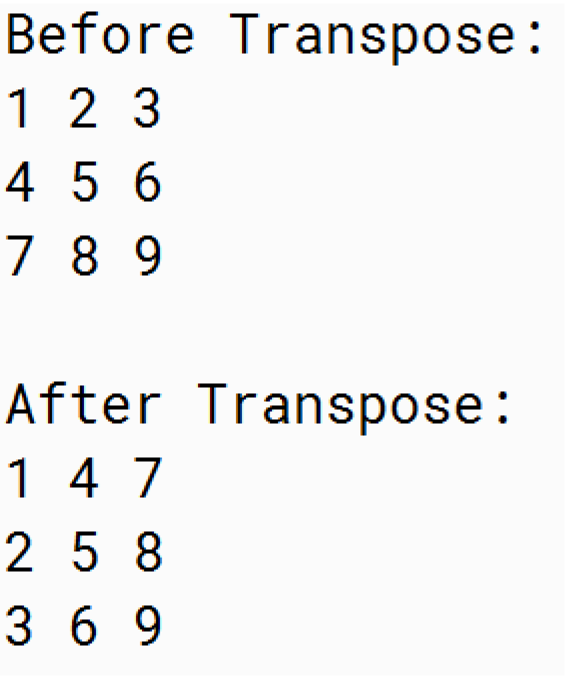
```
isDiverse(a)=false
isDiverse(b)=true
```

Matrix Transpose

- Each pair of elements will be swapped only once.
- Use K-map traversal. (upper half or lower half)
- Swap arr[r][c] with arr[c][r]



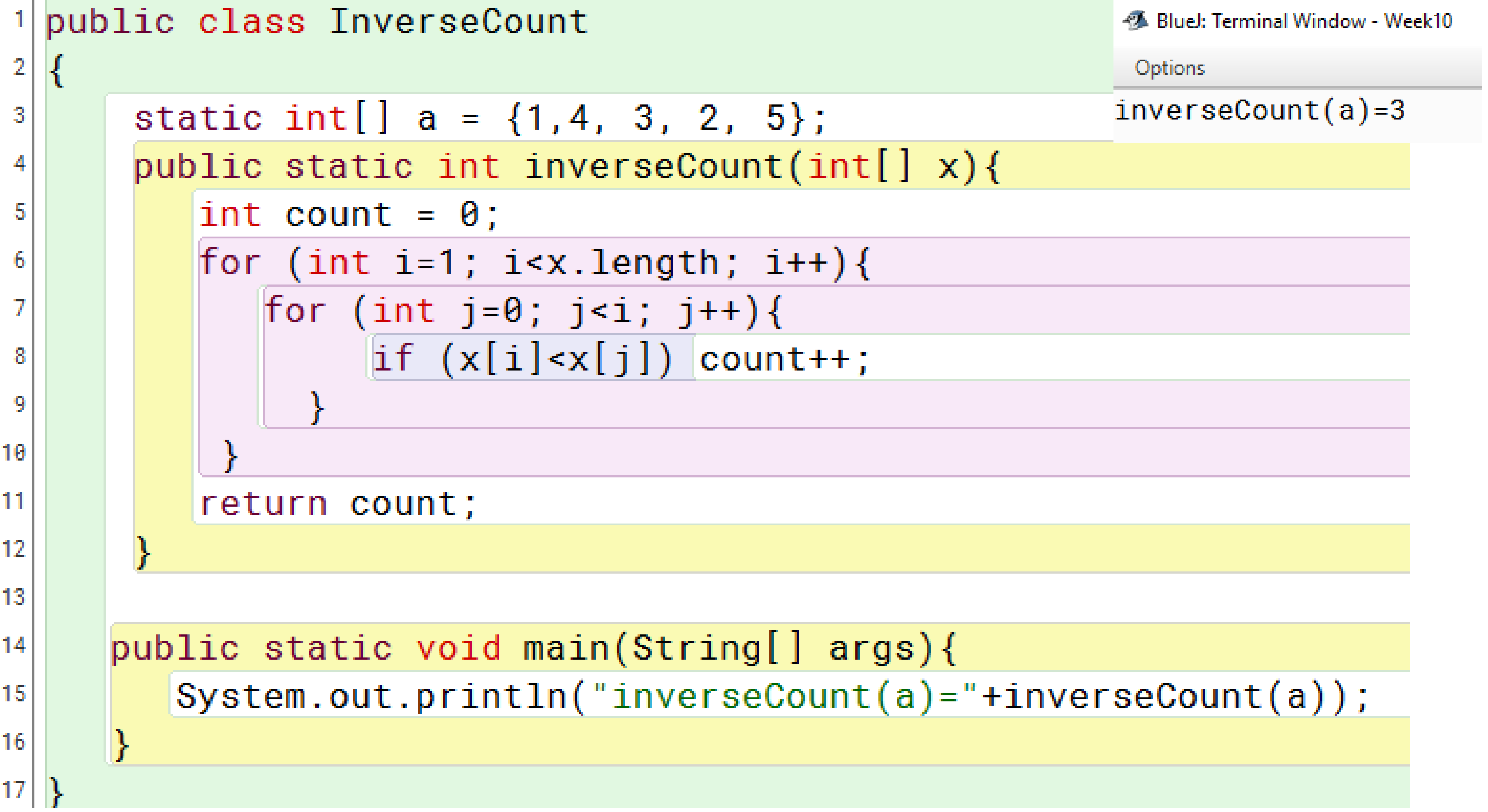
```java
public class MatrixTranspose{
    static int[][] m={
      {1, 2, 3},
      {4, 5, 6},
      {7, 8, 9}
    };
    public static void transpose(int[][] m){
        for (int i=1; i<m.length; i++){
          for (int j=0; j<i; j++){
               int tmp = m[i][j];
               m[i][j] = m[j][i];
               m[j][i] = tmp;
            }
        }
    }
    public static void display(int[][] m){
       for (int i=0; i<m.length; i++){
            for (int j=0; j<m[i].length; j++){
               System.out.print(m[i][j]+" ");
            }
            System.out.println();
        }
    }
    public static void main(String[] args){
      System.out.println("\fBefore Transpose:");
      display(m);
      transpose(m);
      System.out.println("\nAfter Transpose:");
      display(m);
    }
}
```



```
Before Transpose:
1 2 3
4 5 6
7 8 9

After Transpose:
1 4 7
2 5 8
3 6 9
```

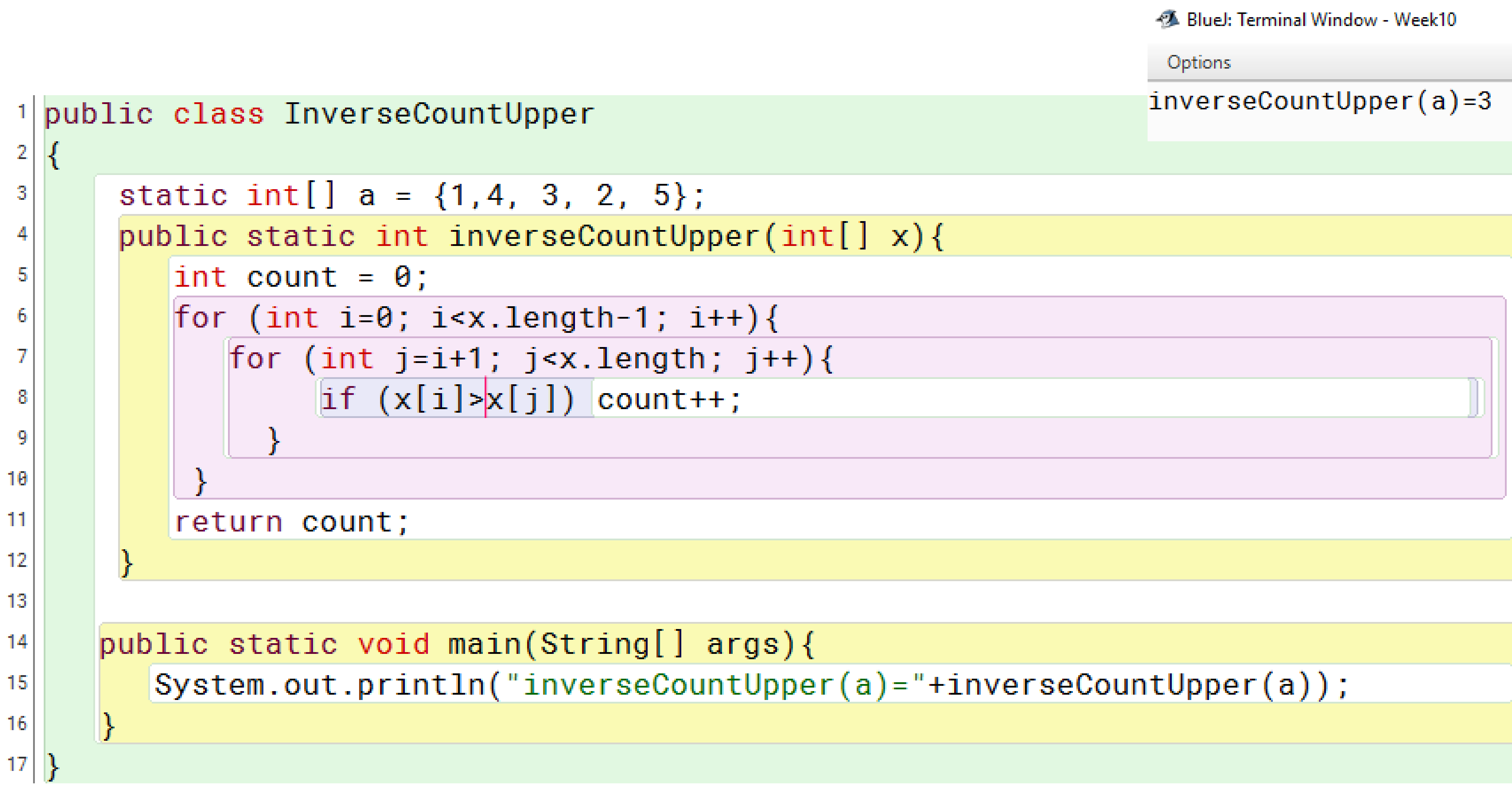
Inverse Order Count

- Given an array, see how many inverse relationship among the numbers
- Example: a = {1, 4, 3, 2, 5};
- Totally 3 inverse.



```java
public class InverseCount
{
    static int[] a = {1,4, 3, 2, 5};
    public static int inverseCount(int[] x){
       int count = 0;
       for (int i=1; i<x.length; i++){
          for (int j=0; j<i; j++){
               if (x[i]<x[j]) count++;
            }
        }
       return count;
    }

   public static void main(String[] args){
      System.out.println("inverseCount(a)="+inverseCount(a));
   }
}
```

```
inverseCount(a)=3
```

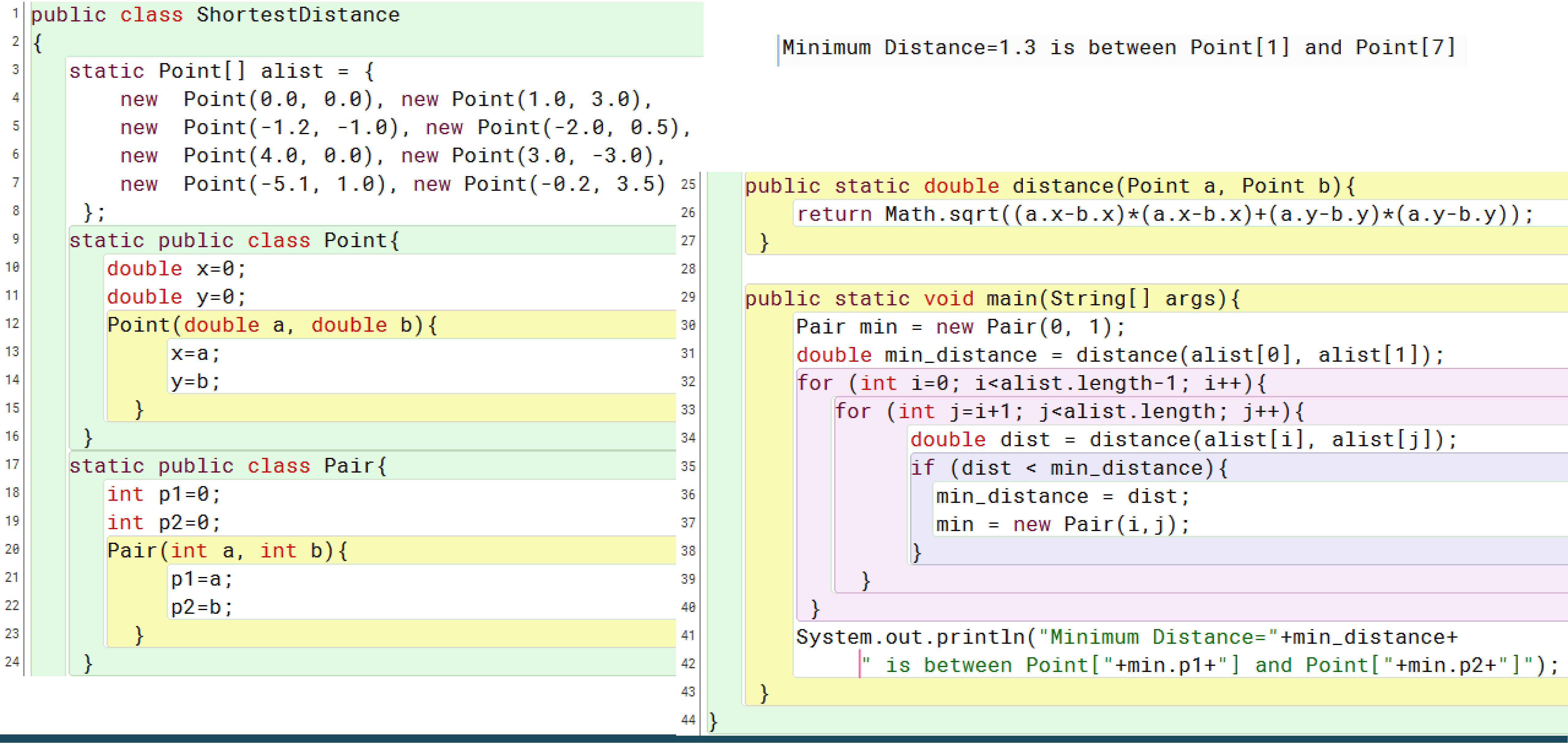


```java
public class InverseCountUpper
{
    static int[] a = {1,4, 3, 2, 5};
    public static int inverseCountUpper(int[] x){
        int count = 0;
        for (int i=0; i<x.length-1; i++){
            for (int j=i+1; j<x.length; j++){
                if (x[i]>x[j]) count++;
            }
        }
        return count;
    }

    public static void main(String[] args){
        System.out.println("inverseCountUpper(a)="+inverseCountUpper(a));
    }
}
```

```
inverseCountUpper(a)=3
```

Shortest Distance

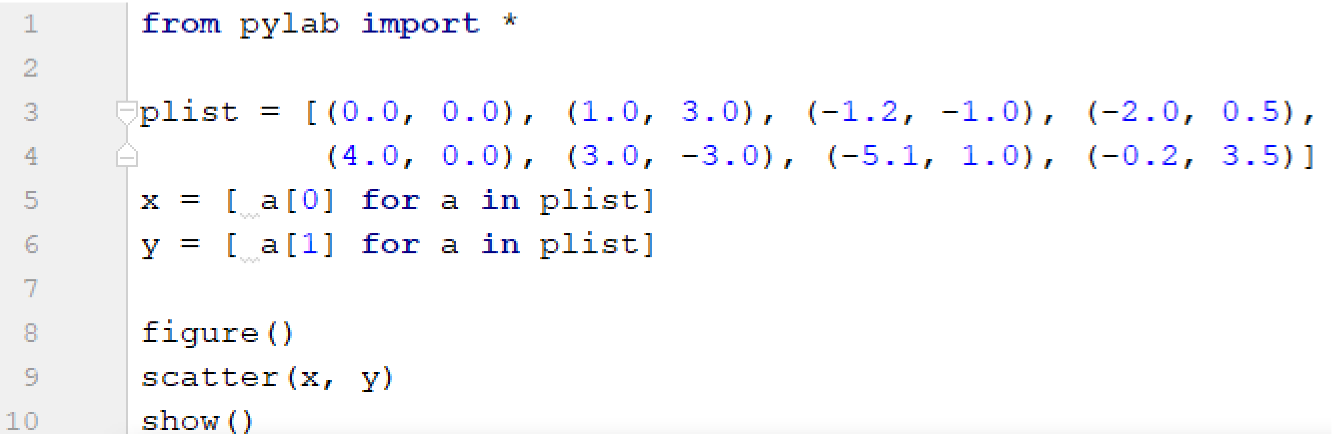
- Find the two points in a set of 8 points. These two points have the shortest distance between them.
- Using K-map upper-half 2-D Traversal.
- Compare and identify the pair of points which have the minimum distance.
- Use Pair(x, y) as the two tuples return value.
- Use Point(a, b) as the two tuples parameters.



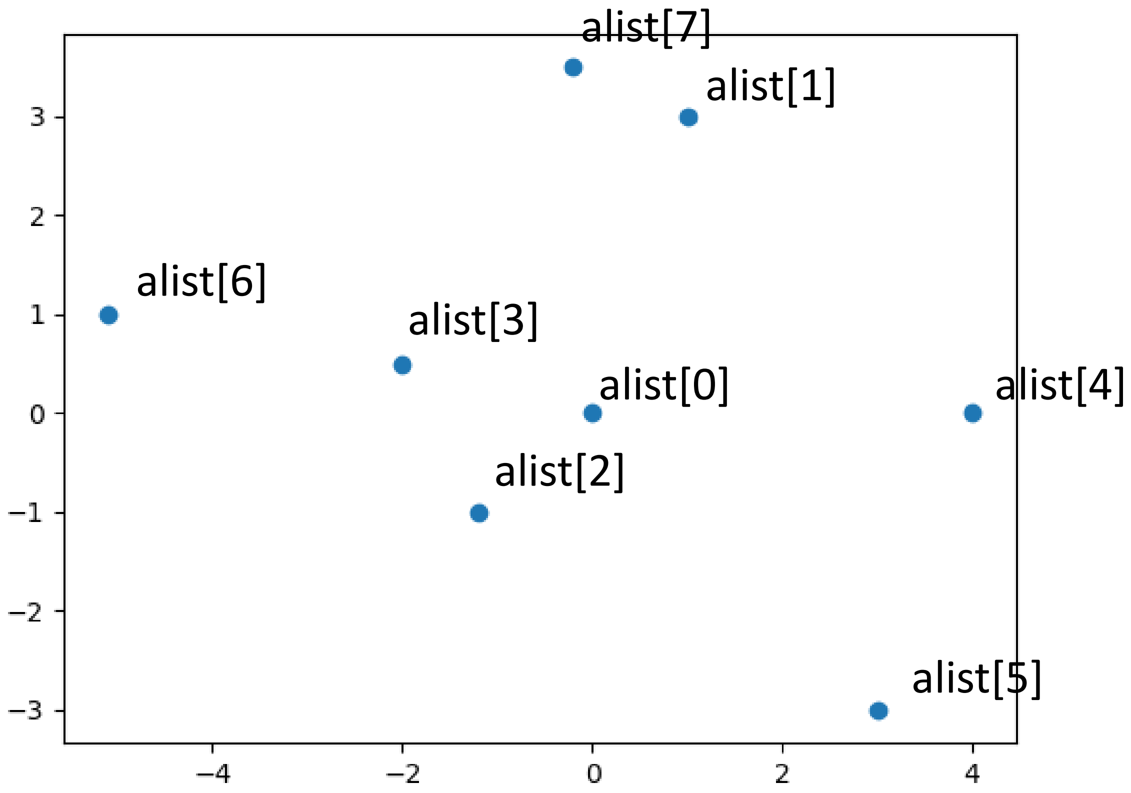
```java
public class ShortestDistance
{
    static Point[] alist = {
        new  Point(0.0, 0.0), new Point(1.0, 3.0),
        new  Point(-1.2, -1.0), new Point(-2.0, 0.5),
        new  Point(4.0, 0.0), new Point(3.0, -3.0),
        new  Point(-5.1, 1.0), new Point(-0.2, 3.5)
    };
    static public class Point{
        double x=0;
        double y=0;
        Point(double a, double b){
            x=a;
            y=b;
        }
    }
    static public class Pair{
        int p1=0;
        int p2=0;
        Pair(int a, int b){
            p1=a;
            p2=b;
        }
    }
    public static double distance(Point a, Point b){
        return Math.sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
    }

    public static void main(String[] args){
        Pair min = new Pair(0, 1);
        double min_distance = distance(alist[0], alist[1]);
        for (int i=0; i<alist.length-1; i++){
            for (int j=i+1; j<alist.length; j++){
                double dist = distance(alist[i], alist[j]);
                if (dist < min_distance){
                    min_distance = dist;
                    min = new Pair(i,j);
                }
            }
        }
        System.out.println("Minimum Distance="+min_distance+
            " is between Point["+min.p1+"] and Point["+min.p2+"]");
    }
}
```

```
Minimum Distance=1.3 is between Point[1] and Point[7]
```

Showing Minimum Distance of Points Using PyLab

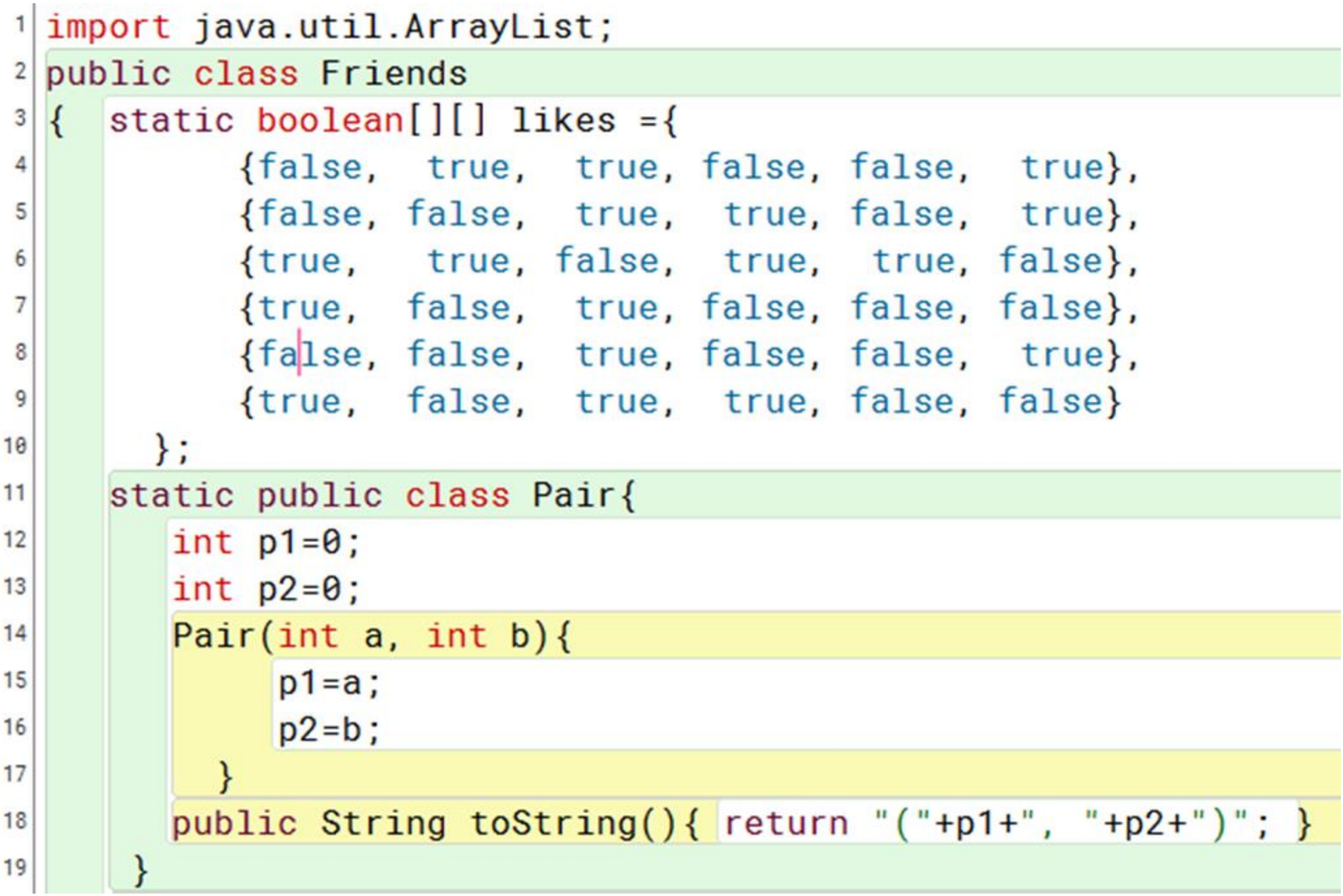


```
from pylab import *

plist = [(0.0, 0.0), (1.0, 3.0), (-1.2, -1.0), (-2.0, 0.5),
         (4.0, 0.0), (3.0, -3.0), (-5.1, 1.0), (-0.2, 3.5)]
x = [ a[0] for a in plist]
y = [ a[1] for a in plist]

figure()
scatter(x, y)
show()
```

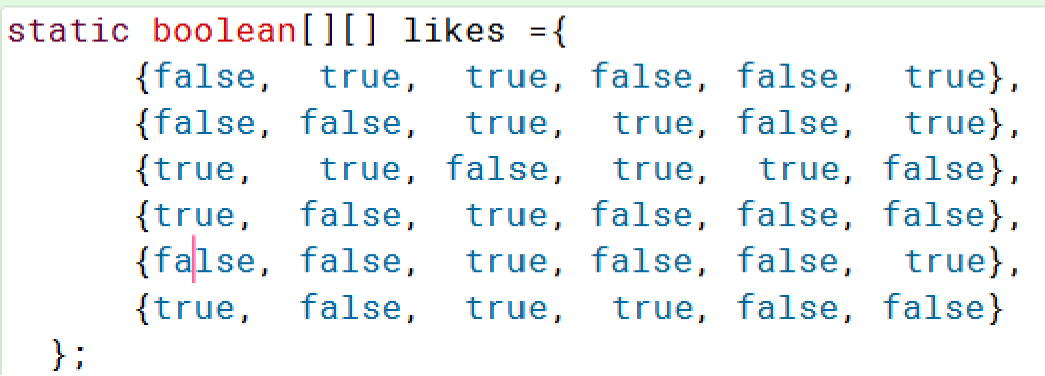


Friendship Checking

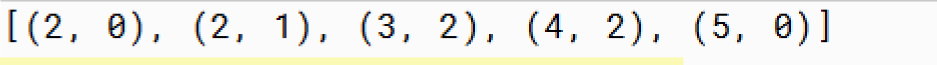
- Mutual Checking for friendship relationship.
- We have an matrix of likes[][]. If a person who id is i likes another person j, then like[i][j] = true.
- If (likes[i][j] && likes[j][i]), then the function isFriends(i, j) will return true.
- Find out all the pair of the people who are friends. Using the same pair class from the previous example.



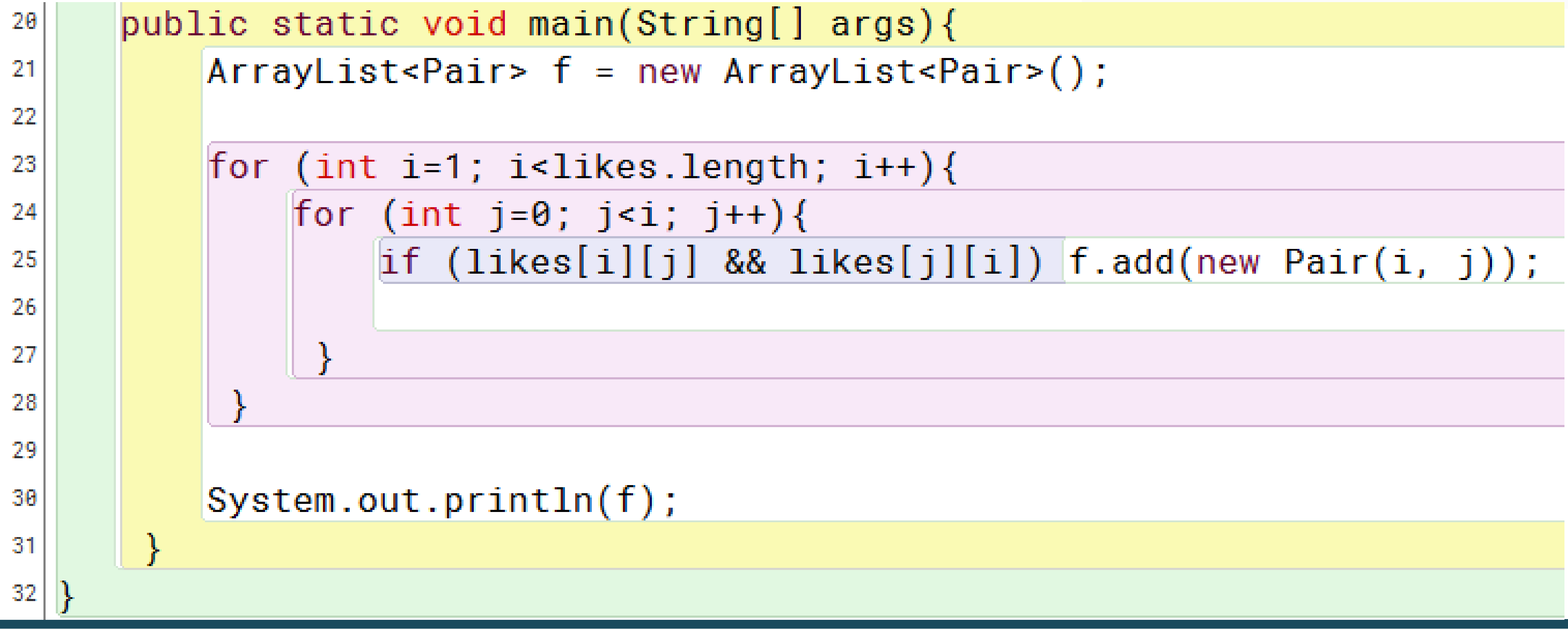
```java
import java.util.ArrayList;
public class Friends
{  static boolean[][] likes ={
         {false,  true,  true, false, false,  true},
         {false, false,  true,  true, false,  true},
         {true,    true, false,  true,  true, false},
         {true,  false,  true, false, false, false},
         {false, false,  true, false, false,  true},
         {true,  false,  true,  true, false, false}
     };
   static public class Pair{
      int p1=0;
      int p2=0;
      Pair(int a, int b){
           p1=a;
           p2=b;
        }
      public String toString(){ return "("+p1+", "+p2+")"; }
    }
```



```
static boolean[][] likes ={
        {false,  true,  true, false, false,  true},
        {false, false,  true,  true, false,  true},
        {true,   true, false,  true,  true, false},
        {true,  false,  true, false, false, false},
        {false, false,  true, false, false,  true},
        {true,  false,  true,  true, false, false}
  };
```



```
[(2, 0), (2, 1), (3, 2), (4, 2), (5, 0)]
```



```
    public static void main(String[] args){
        ArrayList<Pair> f = new ArrayList<Pair>();

        for (int i=1; i<likes.length; i++){
            for (int j=0; j<i; j++){
                if (likes[i][j] && likes[j][i]) f.add(new Pair(i, j));

            }
        }

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