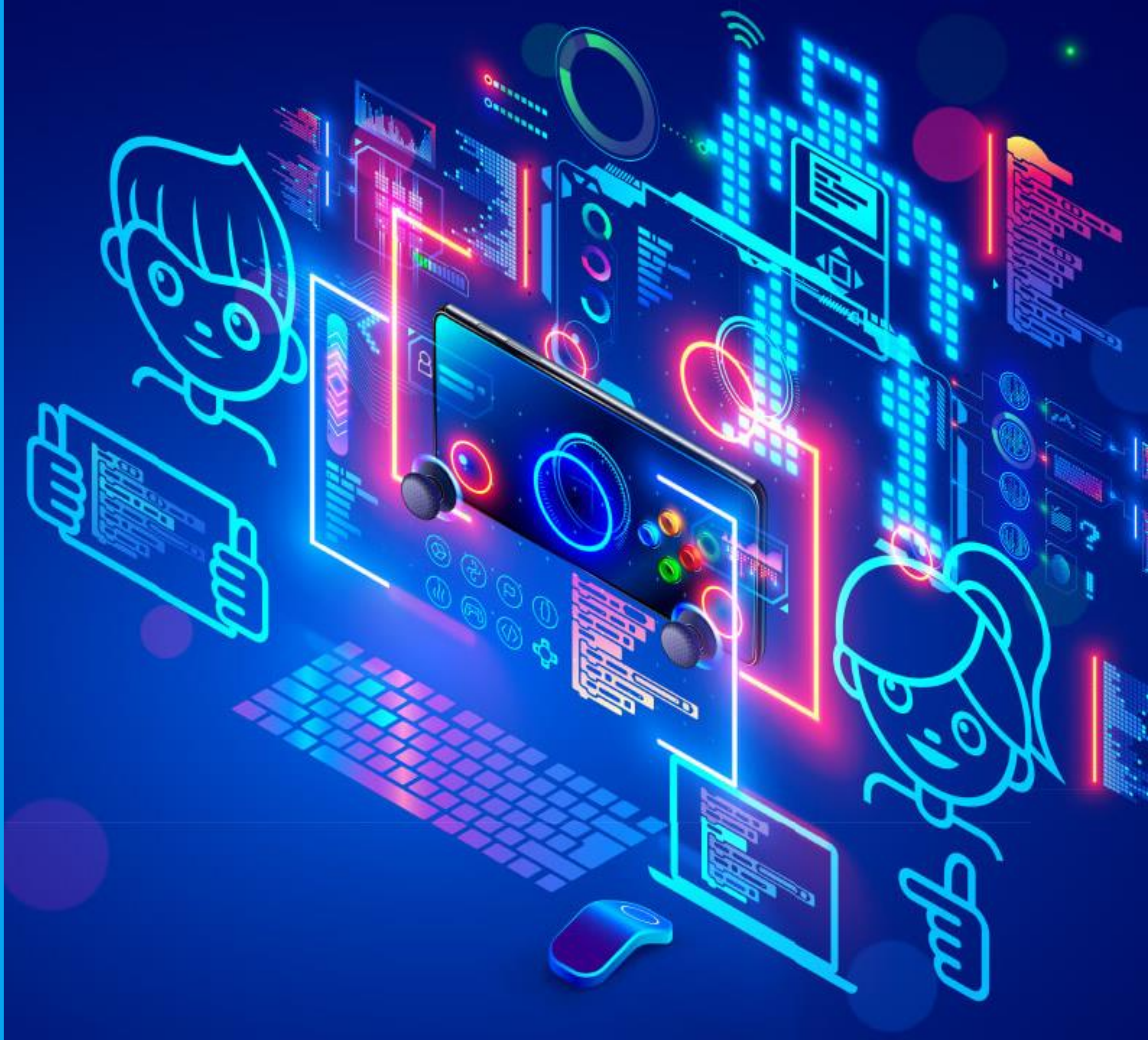
CS 24 AP Computer Science A Review

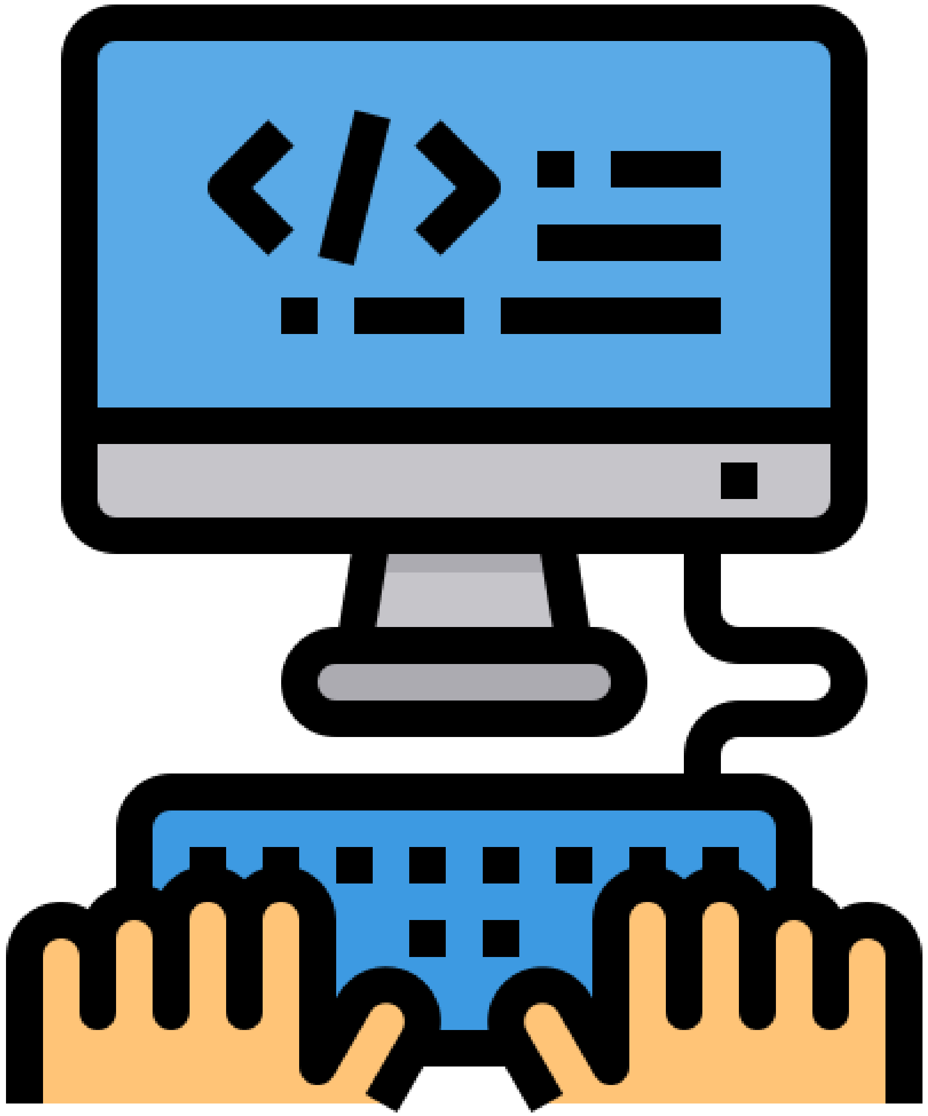
Week 6: 2D Array and ArrayList

DR. ERIC CHOU
IEEE SENIOR MEMBER



Topics

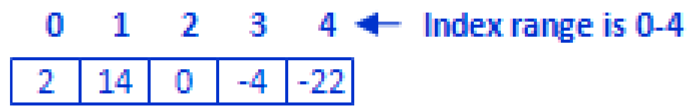
- ArrayList, ArrayList to Array, Array to ArrayList
- ArrayList Processing (Non-recurring List, Frequency List, Priority List, Interval List, Gap List, Available List)
- Removal of Elements from an ArrayList
- 2D Arrays
- 2D Array Processing



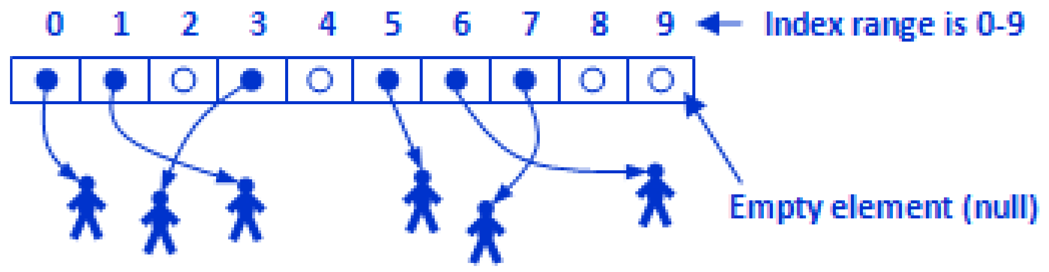
ArrayList

Section 1

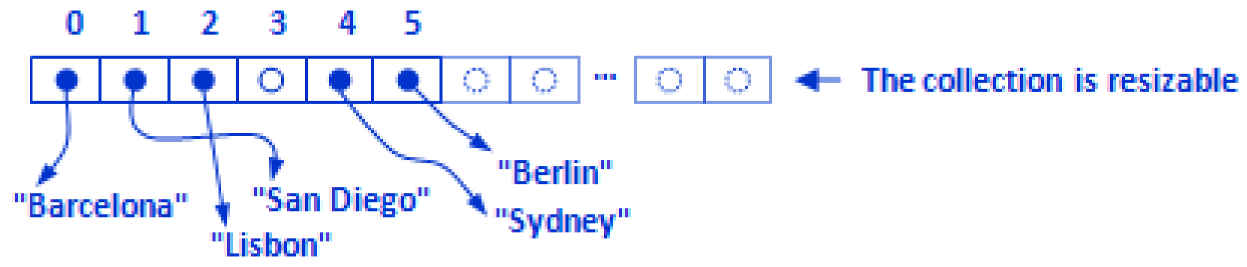
Array of 5 integers



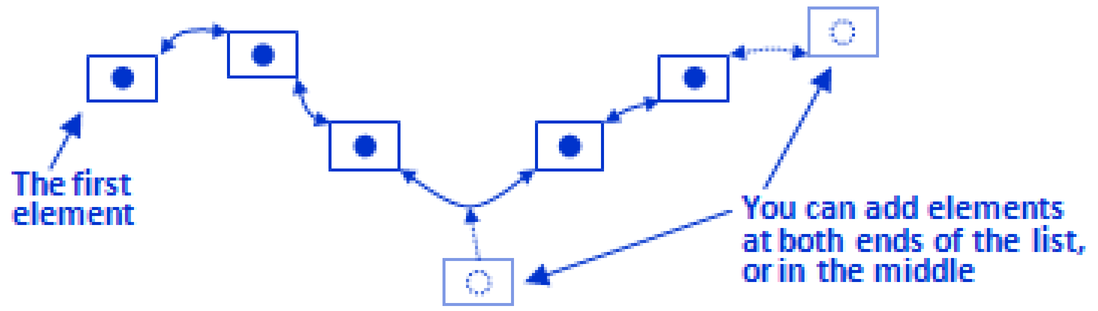
Array of 10 agents



ArrayList (collection) of strings, currently contains 6 elements



LinkedList (collection)


The first element
You can add elements at both ends of the list, or in the middle

Declaration of ArrayList

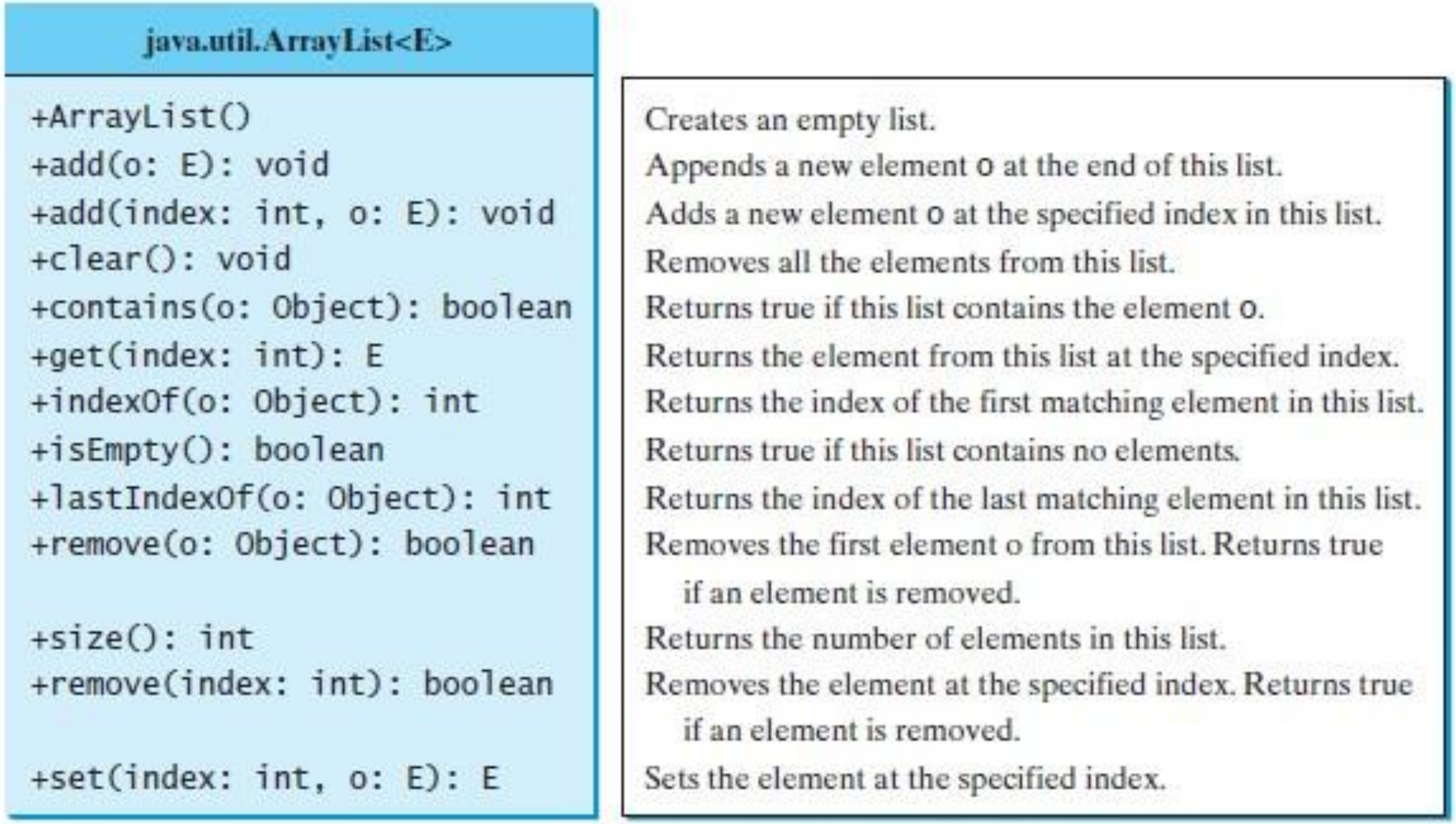
like a train. The datatype it carries is like the cargo.

- In declaring a variable of type **ArrayList** we use a statement like this:
  - **ArrayList<String> aList;**
- The word between the *angle brackets*, **<…>**, indicates the data type of the elements that the **ArrayList** will store. In this case, **aList** is declared to be an **ArrayList** each of whose elements will be a **String**.

Declaration of ArrayList

like a train. The datatype it carries is like the cargo.

- The following statement creates an **ArrayList** of **String**s and then assigns it to **aList**:
  aList = new ArrayList<String>();
- We can also declare and assign to the variable in a single statement:
  ArrayList<String> aList = new ArrayList<String>();



java.util.ArrayList<E>	
+ArrayList()	Creates an empty list.
+add(o: E): void	Appends a new element o at the end of this list.
+add(index: int, o: E): void	Adds a new element o at the specified index in this list.
+clear(): void	Removes all the elements from this list.
+contains(o: Object): boolean	Returns true if this list contains the element o.
+get(index: int): E	Returns the element from this list at the specified index.
+indexOf(o: Object): int	Returns the index of the first matching element in this list.
+isEmpty(): boolean	Returns true if this list contains no elements.
+lastIndexOf(o: Object): int	Returns the index of the last matching element in this list.
+remove(o: Object): boolean	Removes the first element o from this list. Returns true if an element is removed.
+size(): int	Returns the number of elements in this list.
+remove(index: int): boolean	Removes the element at the specified index. Returns true if an element is removed.
+set(index: int, o: E): E	Sets the element at the specified index.

ArrayList

- An arraylist alwasys starts out empty.
- An arraylist is resizable.
- An arraylist requires an import statement.
- An arraylist can only store objects.

ArrayList Methods

ADD, REMOVE, SET, INDEXOF

Differences and Similarities between Arrays and ArrayList

Operation	*Array*	*ArrayList*
Creating an array/ArrayList	`String[] a = new String[10]`	`ArrayList<String> list = new ArrayList<>();`
Accessing an element	`a[index]`	`list.get(index);`
Updating an element	`a[index] = "London";`	`list.set(index, "London");`
Returning size	`a.length`	`list.size();`
Adding a new element		`list.add("London");`
Inserting a new element		`list.add(index, "London");`
Removing an element		`list.remove(index);`
Removing an element		`list.remove(Object);`
Removing all elements		`list.clear();`

Important Methods

ArrayList<E>

- void add(E object)
- void add(int index, E Object)
- int size()
- E remove(int index)
- E get(int index)
- E set(int index, E object)

Creation of ArrayList:

Constructor, Loop Instantiation with add()

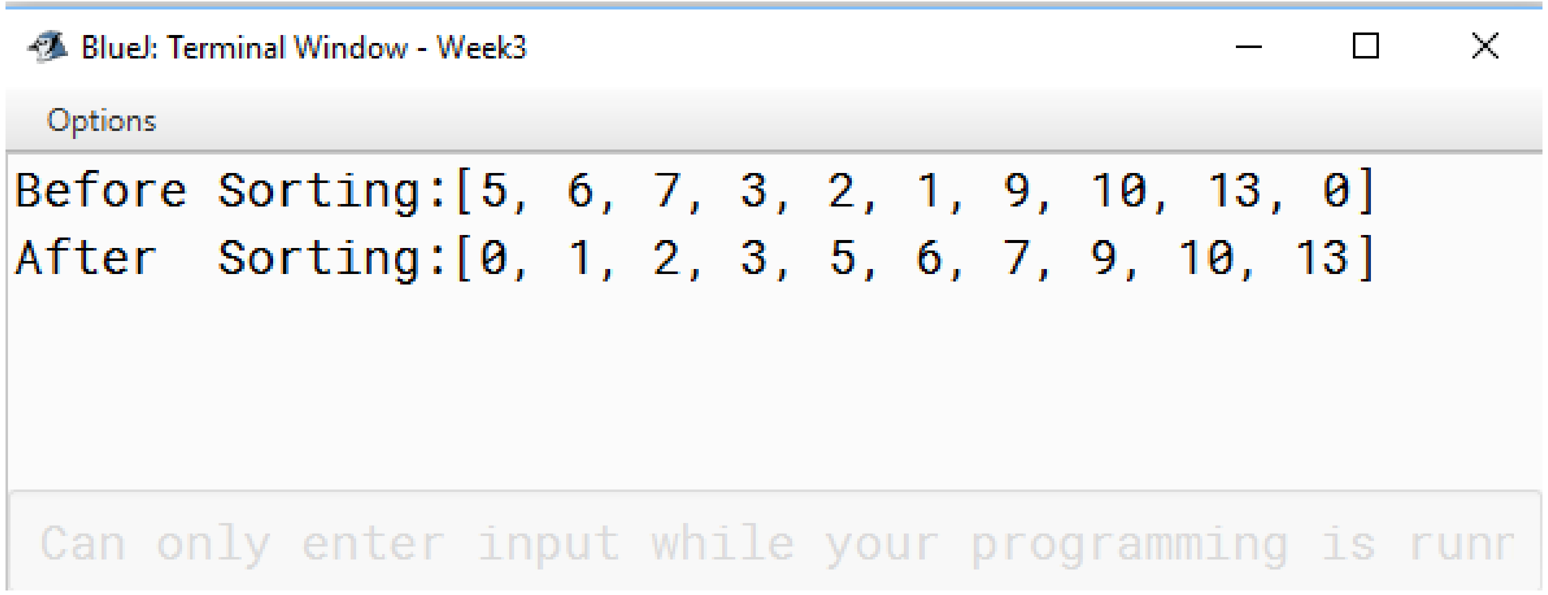
```
public static void main(String[] args){
  int count =5;
  ArrayList<Integer> aList= new ArrayList<Integer>();
  for (int i=0; i<5; i++){
    aList.add((int)(Math.random()*8));
  }
  System.out.println(“Loop Creation of an ArrayList: ”+aList);
}
```

Guideline

- Getter and Setter Methods using index are fine.
- Remove, add with index need extra cautions.

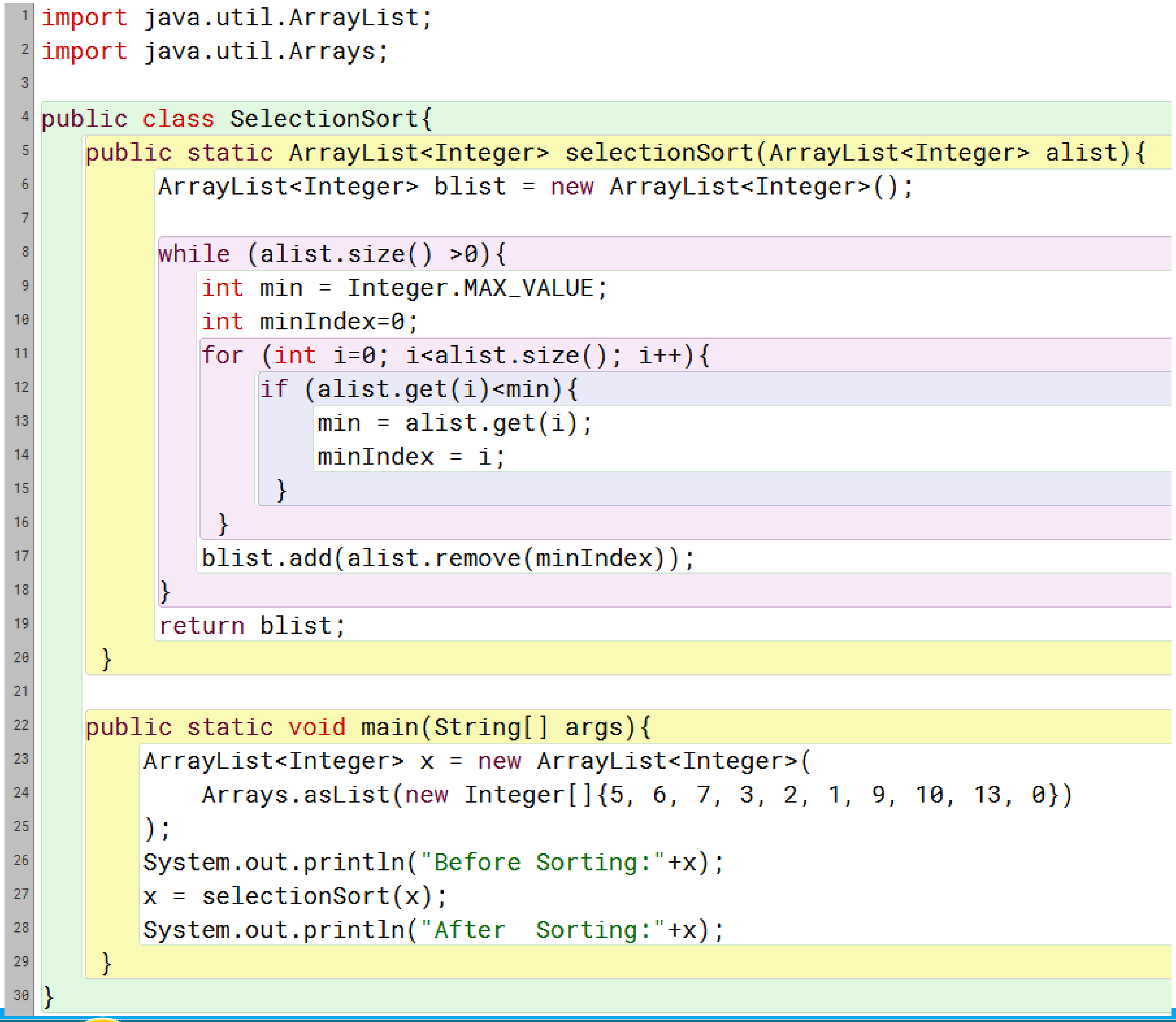
Selection Sort by ArrayList

- If insertion sort or selection sort is required, arraylist is a better option than array.
- It is just a finding maximum with a outer loop.



```
Before Sorting:[5, 6, 7, 3, 2, 1, 9, 10, 13, 0]
After  Sorting:[0, 1, 2, 3, 5, 6, 7, 9, 10, 13]
```

```
Can only enter input while your programming is runr
```

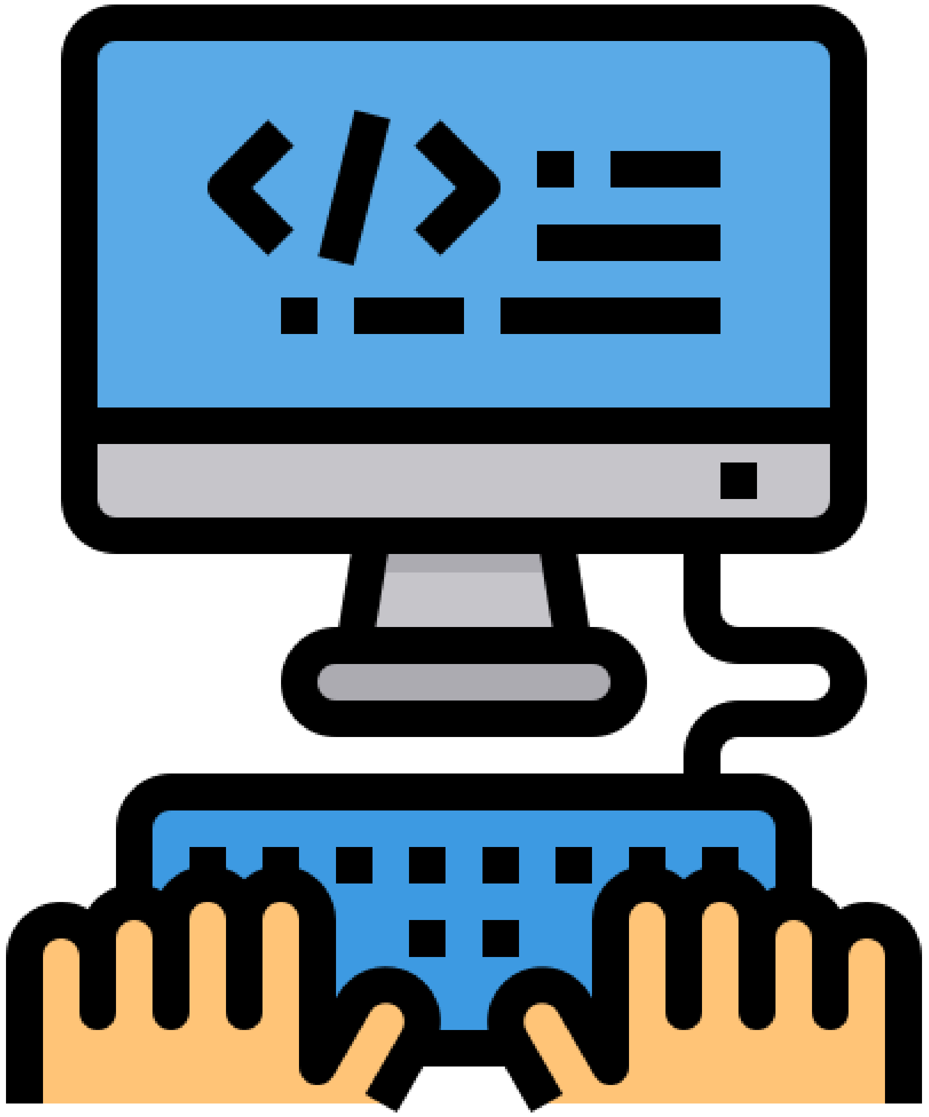


```java
import java.util.ArrayList;
import java.util.Arrays;

public class SelectionSort{
    public static ArrayList<Integer> selectionSort(ArrayList<Integer> alist){
        ArrayList<Integer> blist = new ArrayList<Integer>();

        while (alist.size() >0){
            int min = Integer.MAX_VALUE;
            int minIndex=0;
            for (int i=0; i<alist.size(); i++){
                if (alist.get(i)<min){
                    min = alist.get(i);
                    minIndex = i;
                }
            }
            blist.add(alist.remove(minIndex));
        }
        return blist;
    }

    public static void main(String[] args){
        ArrayList<Integer> x = new ArrayList<Integer>(
            Arrays.asList(new Integer[]{5, 6, 7, 3, 2, 1, 9, 10, 13, 0})
        );
        System.out.println("Before Sorting:"+x);
        x = selectionSort(x);
        System.out.println("After  Sorting:"+x);
    }
}
```



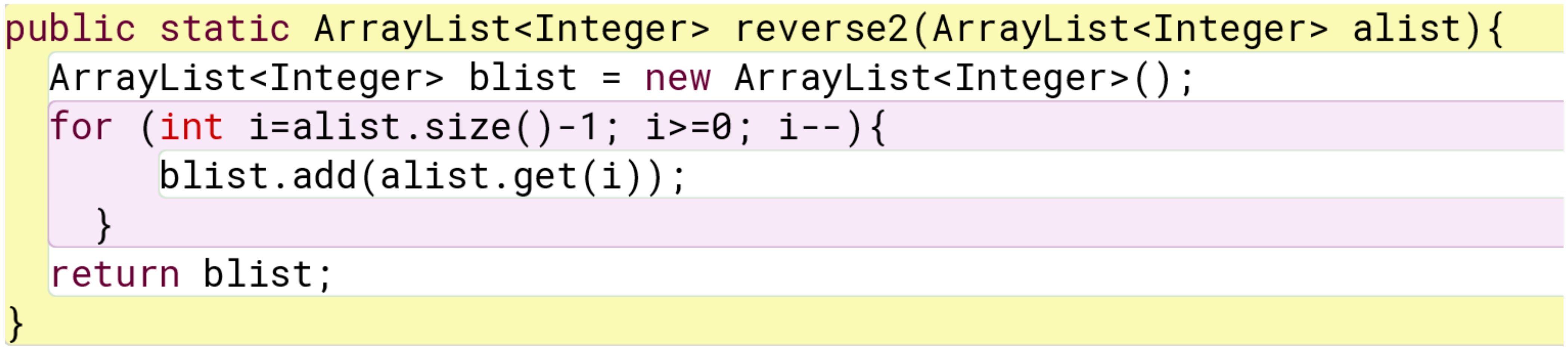
ArrayList Processing

Section 2

Reverse List 1

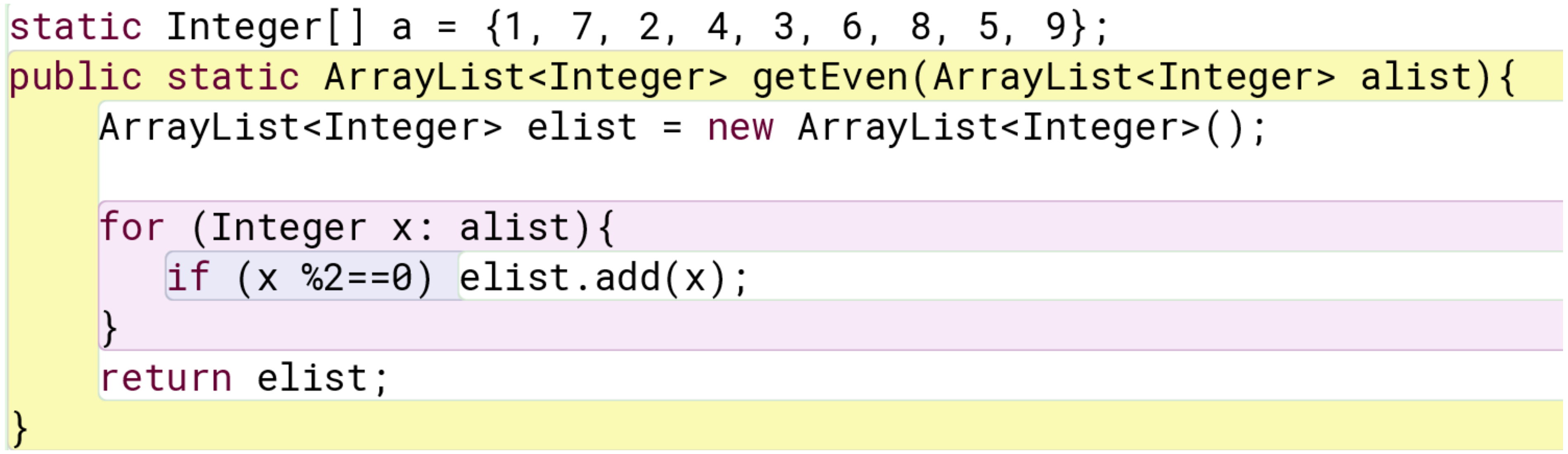
```
static Integer[] a = {3, 5, 8, 1, 4};
public static ArrayList<Integer> reverse(ArrayList<Integer> alist){
   int N = alist.size();
   ArrayList<Integer> blist = new ArrayList<Integer>();
   while (N>0){
        blist.add(0, alist.remove(0));
        N--;
     }
   return blist;
}
```

Reverse List 2



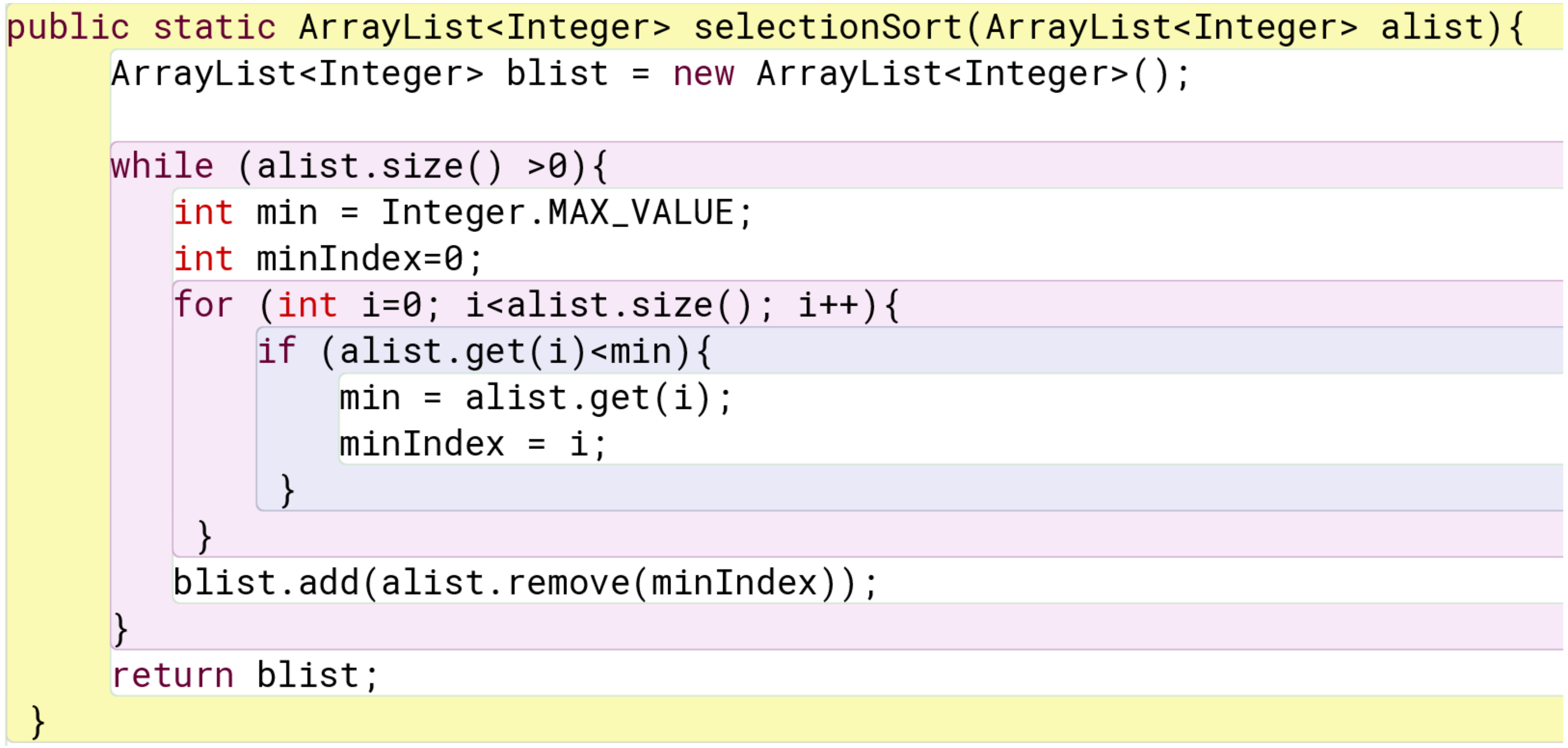
```
public static ArrayList<Integer> reverse2(ArrayList<Integer> alist){
    ArrayList<Integer> blist = new ArrayList<Integer>();
    for (int i=alist.size()-1; i>=0; i--){
          blist.add(alist.get(i));
      }
    return blist;
}
```

Filtering (Get Even Numbers)


```
static Integer[] a = {1, 7, 2, 4, 3, 6, 8, 5, 9};
public static ArrayList<Integer> getEven(ArrayList<Integer> alist){
    ArrayList<Integer> elist = new ArrayList<Integer>();

    for (Integer x: alist){
        if (x %2==0) elist.add(x);
    }
    return elist;
}
```

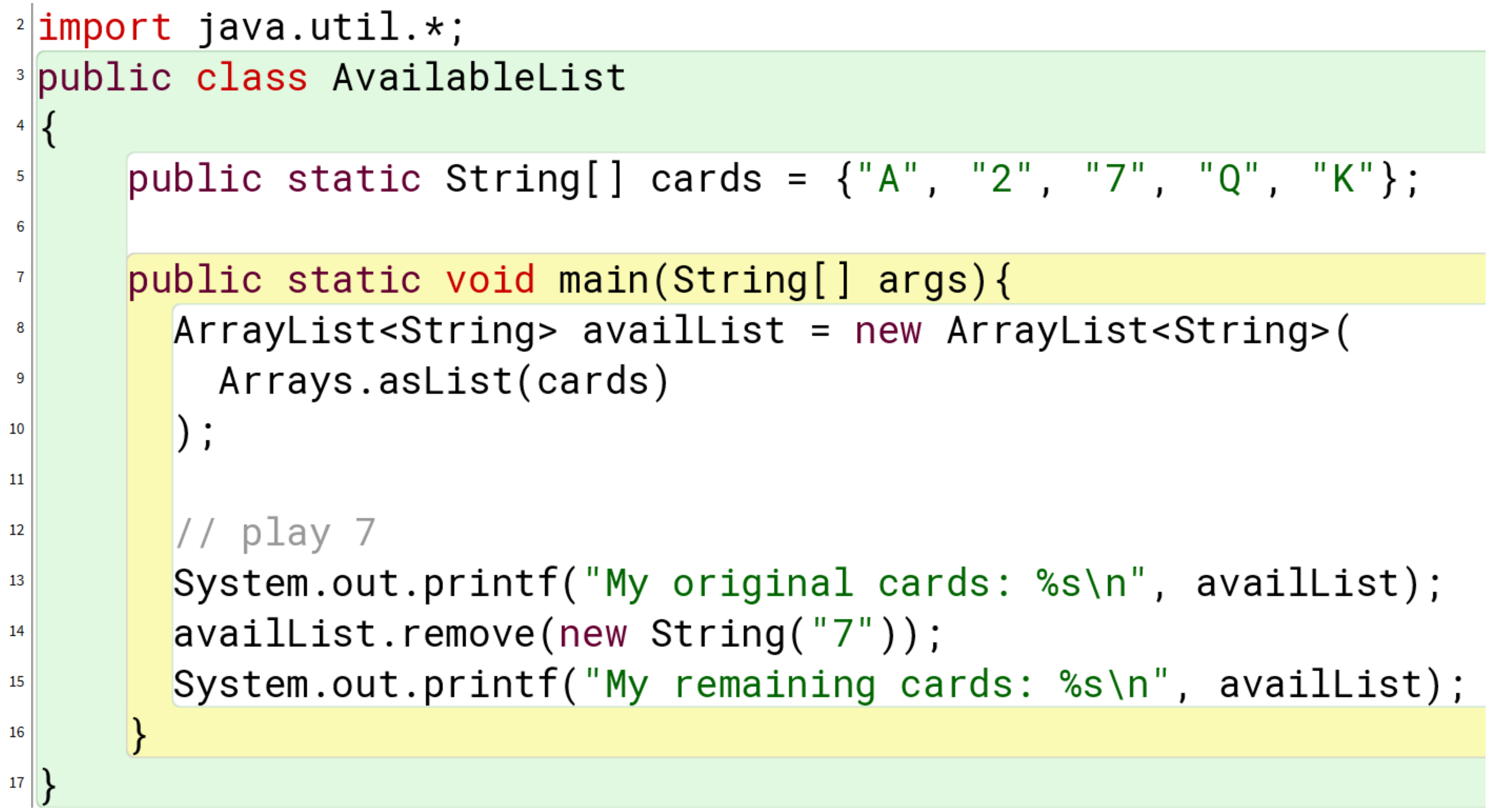
Selection Sort



```java
public static ArrayList<Integer> selectionSort(ArrayList<Integer> alist){
     ArrayList<Integer> blist = new ArrayList<Integer>();

     while (alist.size() >0){
        int min = Integer.MAX_VALUE;
        int minIndex=0;
        for (int i=0; i<alist.size(); i++){
            if (alist.get(i)<min){
                min = alist.get(i);
                minIndex = i;
             }
         }
        blist.add(alist.remove(minIndex));
     }
     return blist;
 }
```

Available List



```java
import java.util.*;
public class AvailableList
{
    public static String[] cards = {"A", "2", "7", "Q", "K"};

    public static void main(String[] args){
        ArrayList<String> availList = new ArrayList<String>(
            Arrays.asList(cards)
        );

        // play 7
        System.out.printf("My original cards: %s\n", availList);
        availList.remove(new String("7"));
        System.out.printf("My remaining cards: %s\n", availList);
    }
}
```

Frequency List

```
public static void findFreq(Integer[] a){
    set.clear();
    freq.clear();
    for (int i=0; i<a.length; i++){
        int idx = set.indexOf(a[i]);
        if (idx <0){
            set.add(a[i]);
            freq.add(0);
            idx = set.size()-1;
        }
        freq.set(idx, freq.get(idx)+1);
    }
}
```

Interval List

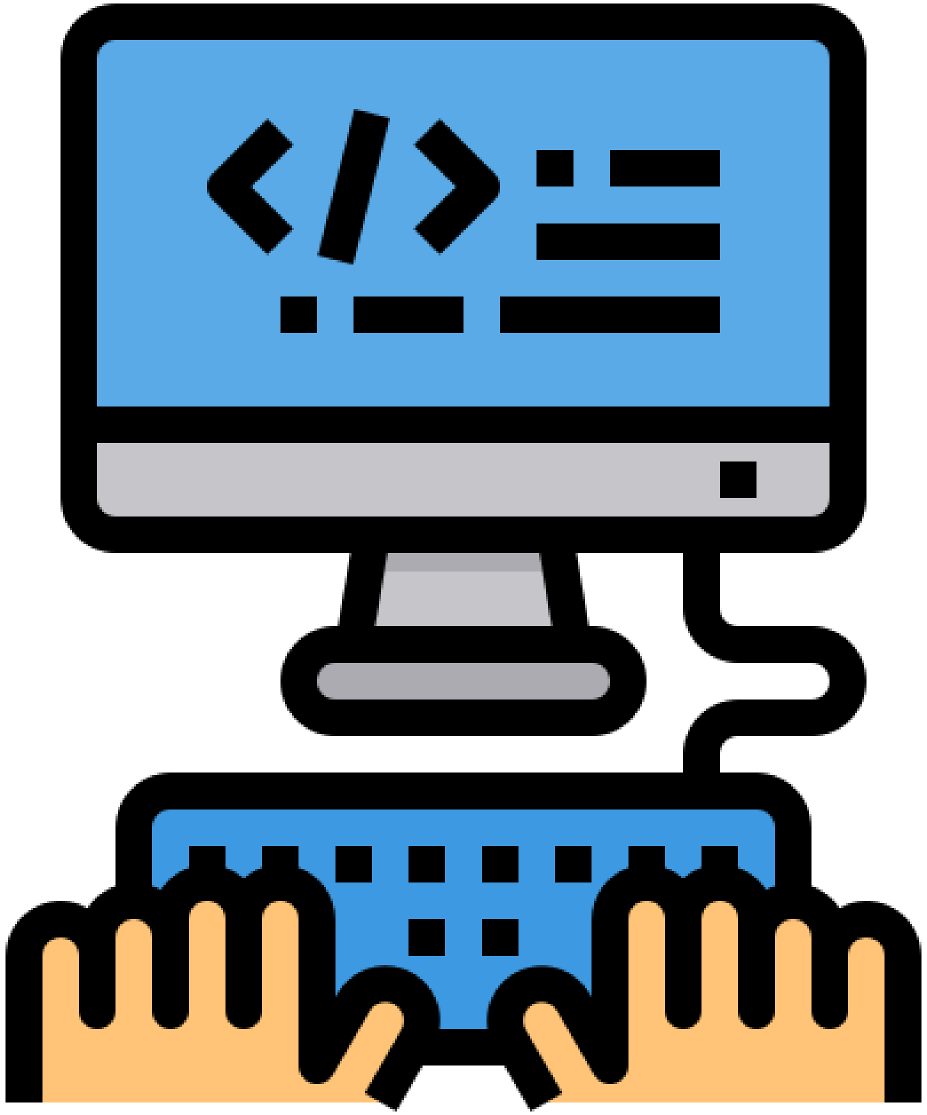
```java
public static int longestRun(String data){
    ArrayList<Integer> cuts = new ArrayList<Integer>();

    cuts.add(0);
    for (int i=0; i<data.length()-1; i++){
        if (data.charAt(i) != data.charAt(i+1)) cuts.add(i+1);
     }
    cuts.add(data.length());
    System.out.println(cuts);
    int m = 0;
    int idx =-1;
    for (int i=0; i<cuts.size()-1; i++){
        int run = cuts.get(i+1)-cuts.get(i);
        if (run > m){
            m = run; idx = cuts.get(i);
          }
     }
    System.out.printf("%c has the longest run of %d\n", data.charAt(idx), m);
    return m;
}
```

Difference List

```
public static ArrayList<Integer> difference(Integer[] nike){
    ArrayList<Integer> diff = new ArrayList<Integer>();

    for (int i=0; i<nike.length-1; i++){
      Integer d = nike[i+1] - nike[i];
      diff.add(d);
    }
    return diff;
}
```

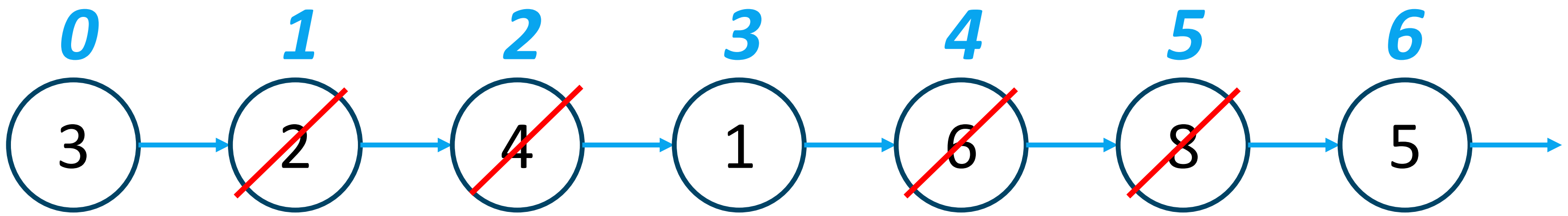


Removal of Elements from an ArrayList

Section 3

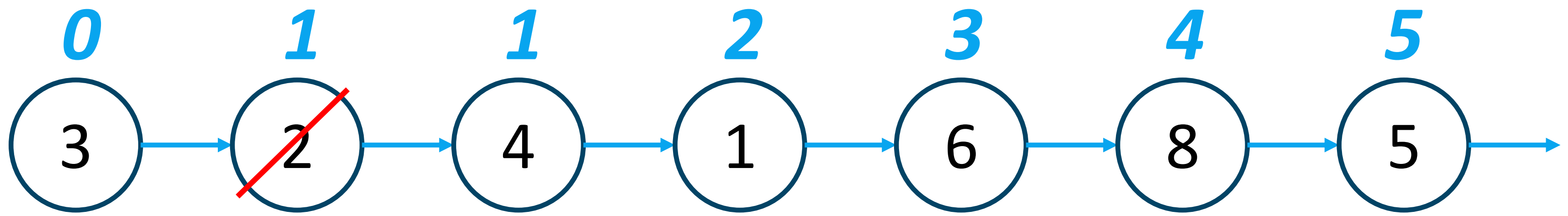
Bypass Problem (Delete Even Numbers)

Our purpose:



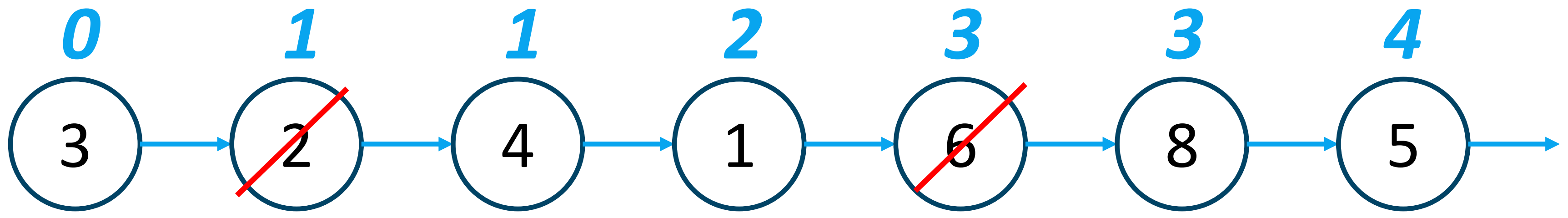
Bypass Problem (Delete Even Numbers)

After removal the index changes

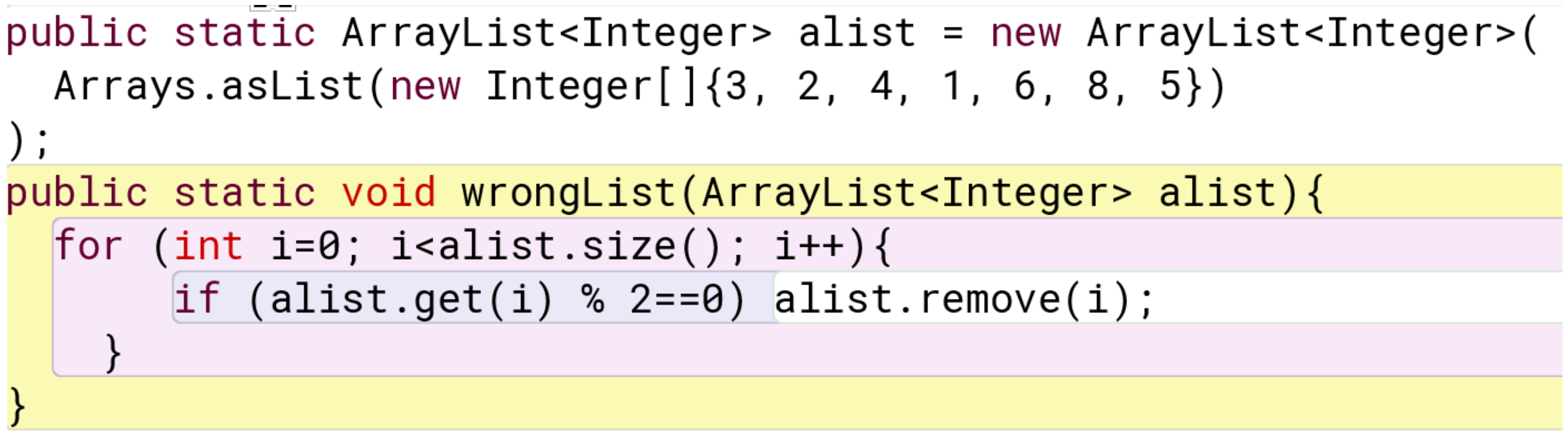


Bypass Problem (Delete Even Numbers)

Missing two spots



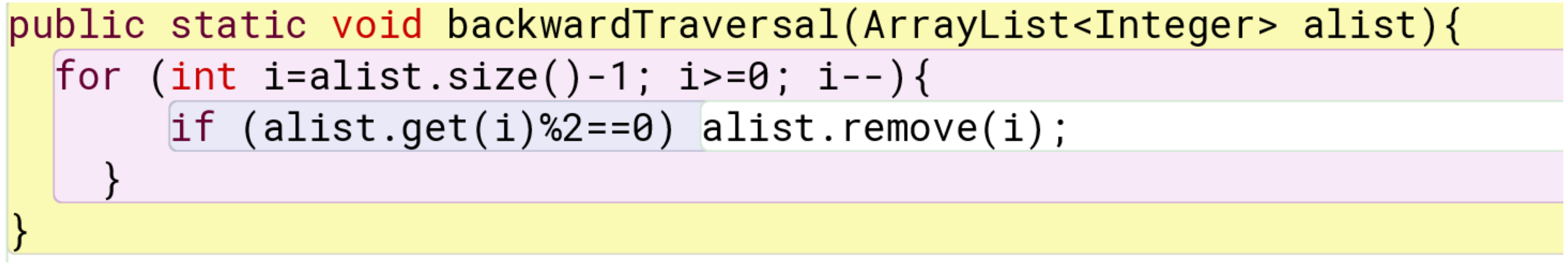
Wrong Index



```java
public static ArrayList<Integer> alist = new ArrayList<Integer>(
  Arrays.asList(new Integer[]{3, 2, 4, 1, 6, 8, 5})
);
public static void wrongList(ArrayList<Integer> alist){
  for (int i=0; i<alist.size(); i++){
       if (alist.get(i) % 2==0) alist.remove(i);
    }
}
```

```
alist before removal: [3, 2, 4, 1, 6, 8, 5]
alist after removal: [3, 4, 1, 8, 5]
```

Backward Traversal



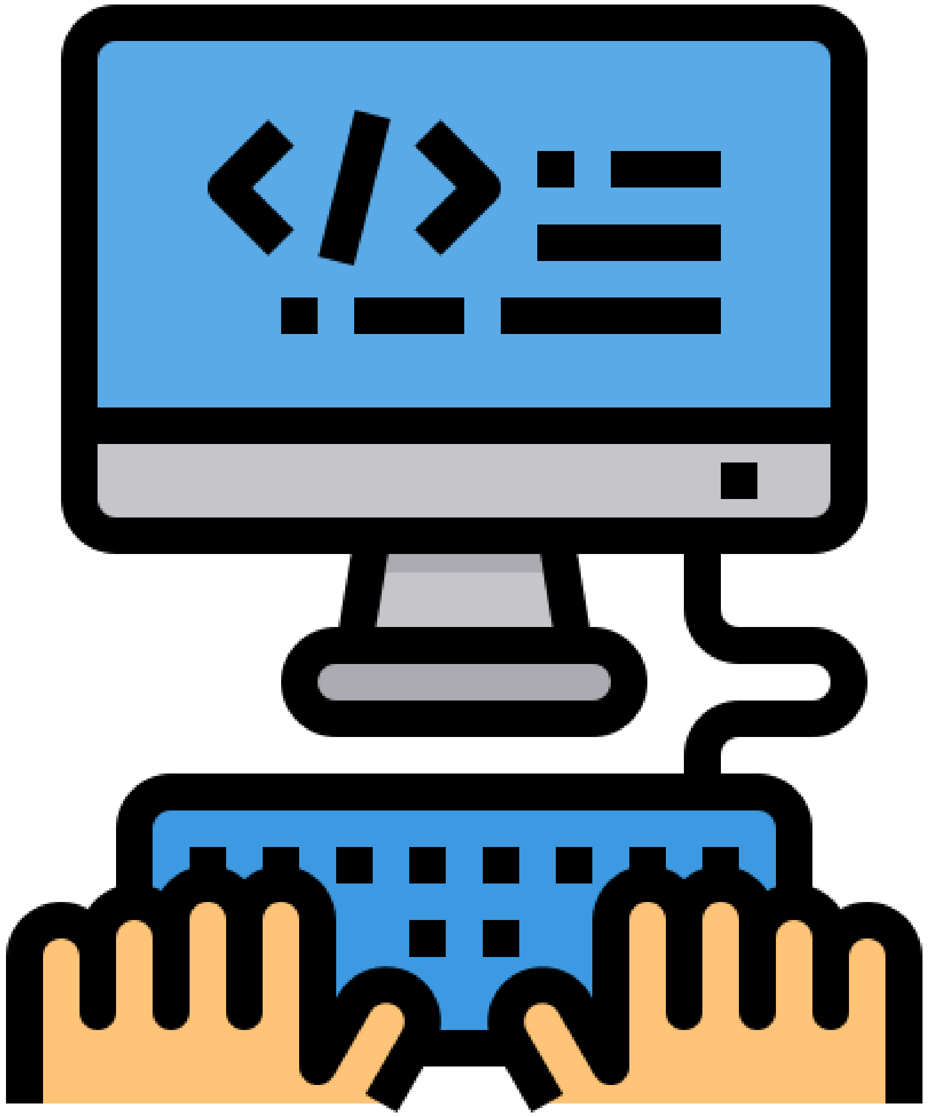
```java
public static void backwardTraversal(ArrayList<Integer> alist){
   for (int i=alist.size()-1; i>=0; i--){
        if (alist.get(i)%2==0) alist.remove(i);
    }
}
```

```
alist before backward traversal: [3, 2, 4, 1, 6, 8, 5]
alist after backward traversal: [3, 1, 5]
```

Delete Without Advancing Index

```
public static void forwardTraversalWithoutAdvancing(ArrayList<Integer> alist){
    for (int i=0; i<alist.size(); ){
        if (alist.get(i)%2==0) alist.remove(i);
        else i++;
    }
}
```

```
alist before forward no advancing traversal: [3, 2, 4, 1, 6, 8, 5]
alist after forward no advancing traversal: [3, 1, 5]
```



2D Arrays

Section 4

General Form for Creating 2-D Array Using a Pre-defined List of Data

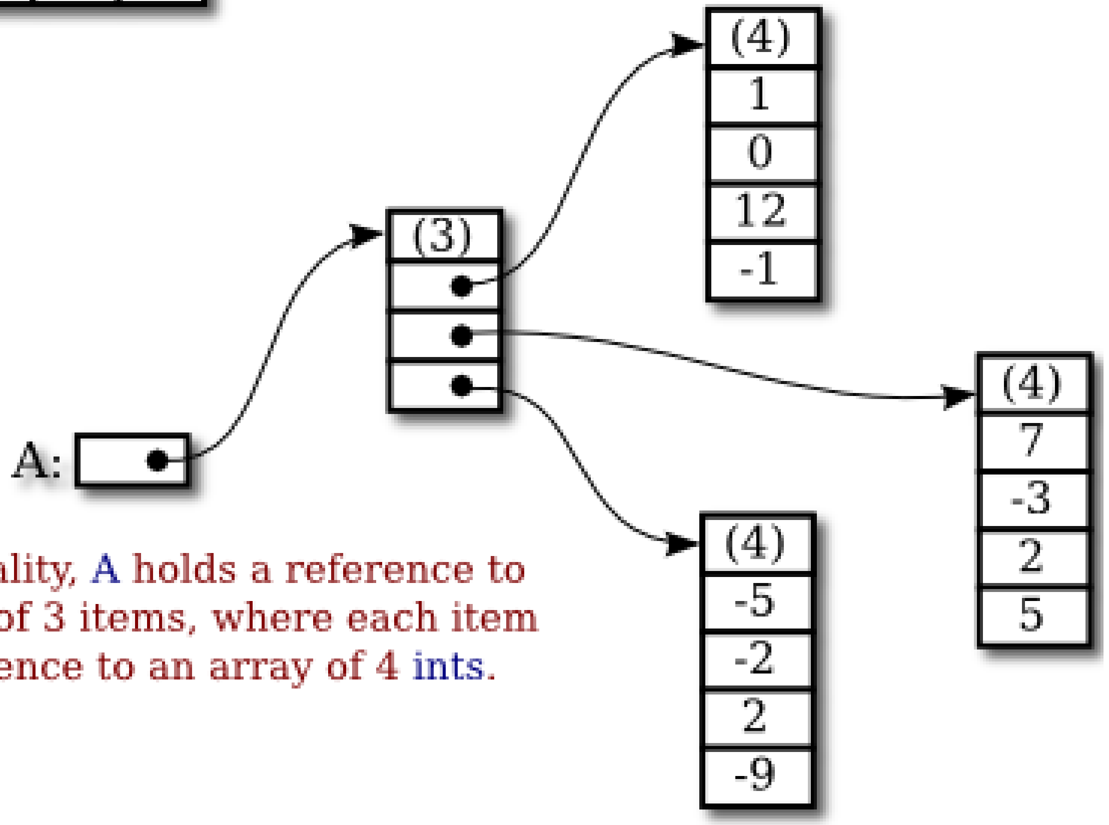
```
datatype[][] nameOf2DArray = {
   {value1, value2, value3},
   {value4, value5, value6},
   { …, …, …}
};
```

```
String[][] candyBoard = {
 {“Jelly Bean”, “Lozenge”, “Lemon Drop”},
 {“Gum Square”, “Lollipop Head”, “Jujube Cluster”},
 {“Lozenge”, “Lollipop Head”, “Lemon Drop”},
 {“Jelly Bean”, “Lollipop Head”, “Lozenge”}
};
```

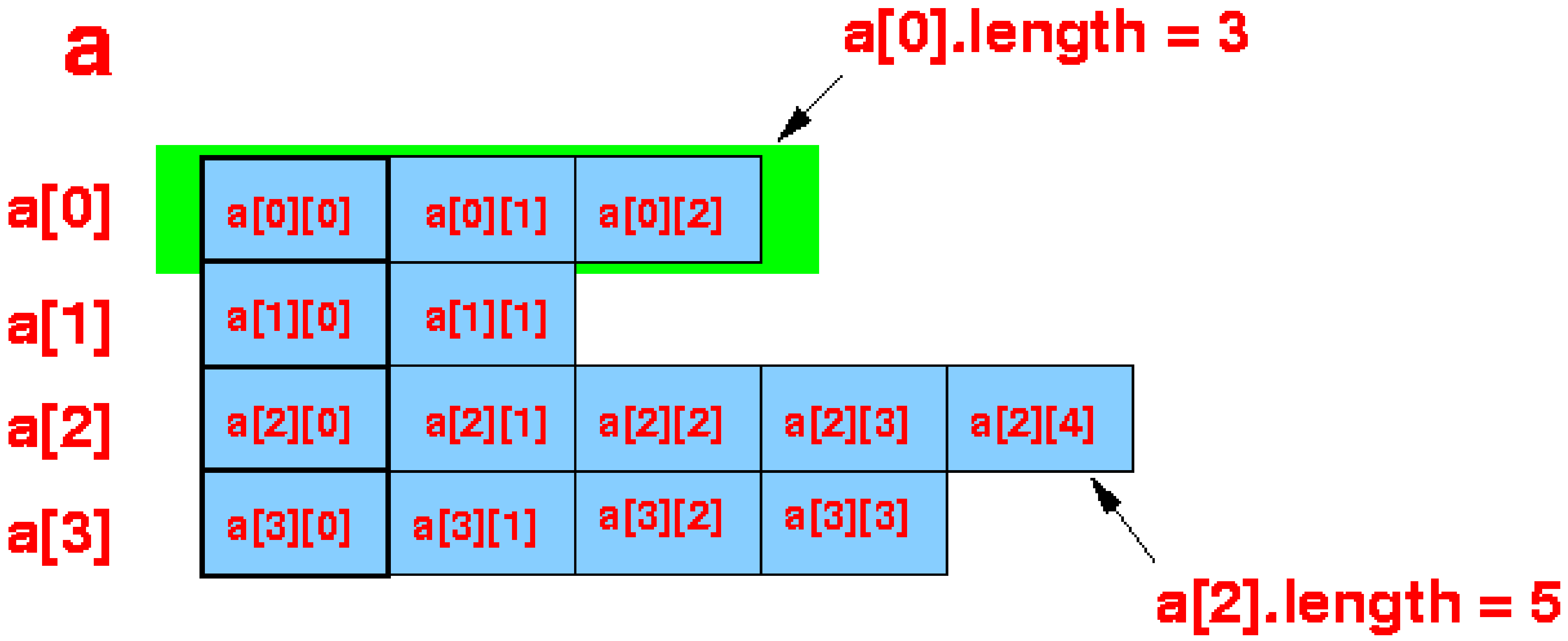
A:

1	0	12	-1
7	-3	2	5
-5	-2	2	-9

If you create an array A = new int[3][4], you should think of it as a "matrix" with 3 rows and 4 columns.



But in reality, A holds a reference to an array of 3 items, where each item is a reference to an array of 4 ints.


a
a[0].length = 3
a[0]
a[1]
a[2]
a[3]
a[0][0]
a[0][1]
a[0][2]
a[1][0]
a[1][1]
a[2][0]
a[2][1]
a[2][2]
a[2][3]
a[2][4]
a[3][0]
a[3][1]
a[3][2]
a[3][3]
a[2].length = 5

Using the length Field to Find the Number of Rows and Columns

Retrieve the number of rows from a 2-D array:

```
double[][] myBoard = new double[8][3];
int result = myBoard.length;
```

Retrieve the number of columns from a 2-D array:

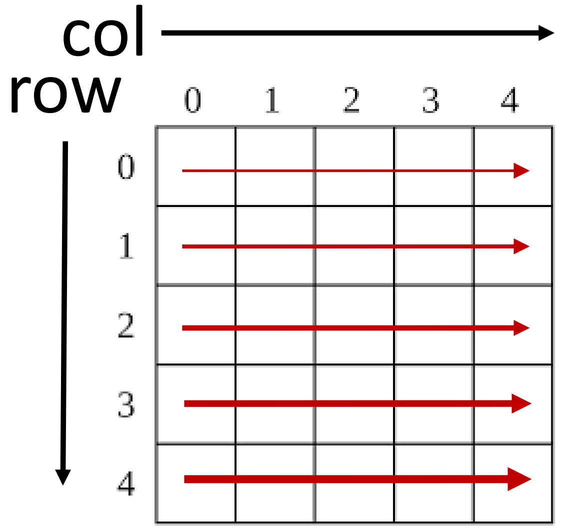
```
double[][] myBoard = new double[8][3];
int result1 = myBoard[0].length;
int result2 = myBoard[5].length;
```

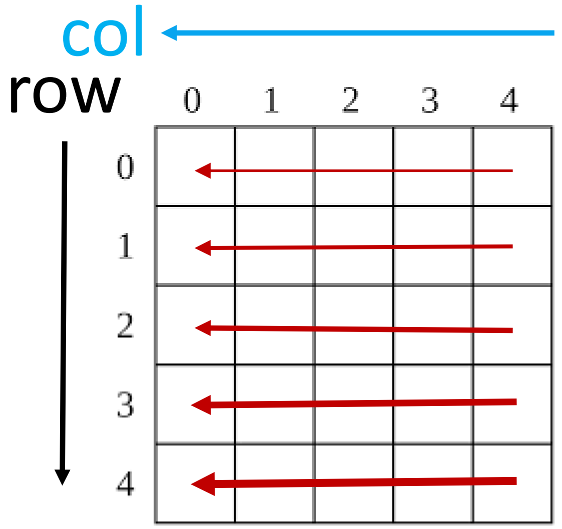
2D Traversal

ROW/COLUMN MANAGEMENT

2-D Array Indexing for Traversal

int row, col; int[][] m=new int[5][5];

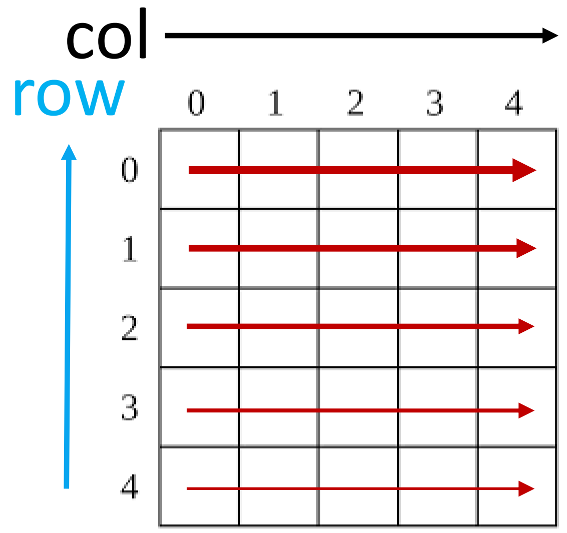


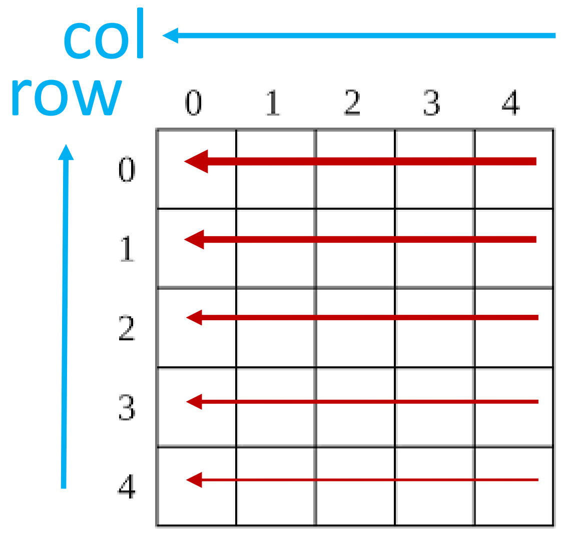


```
for(row=0; row<m.length; row++)
  for(col=0; col<m[0].length; col++)
      System.out.println(m[row][col]);
```

```
for(row=0; row<m.length; row++)
  for(col=m[0].length-1; col>=0; col--)
      System.out.println(m[row][col]);
```

2-D Array Indexing for Traversal
int row, col; int[][] m=new int[5][5];



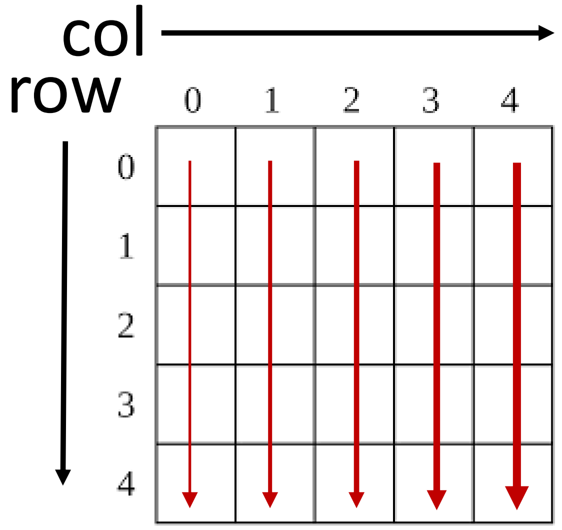


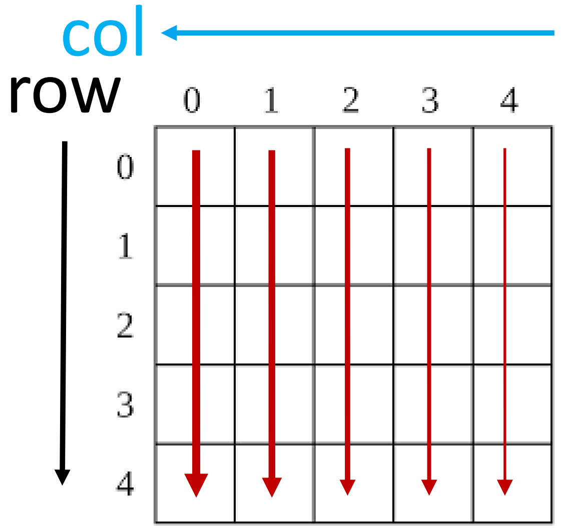
```
for(row=m.length-1; row>=0; row--)
  for(col=0; col<m[0].length; col++)
    System.out.println(m[row][col]);
```

```
for(row=m.length-1; row>=0; row--)
  for(col=m[0].length-1; col>=0; col--)
    System.out.println(m[row][col]);
```

2-D Array Indexing for Traversal

int row, col; int[][] m=new int[5][5];

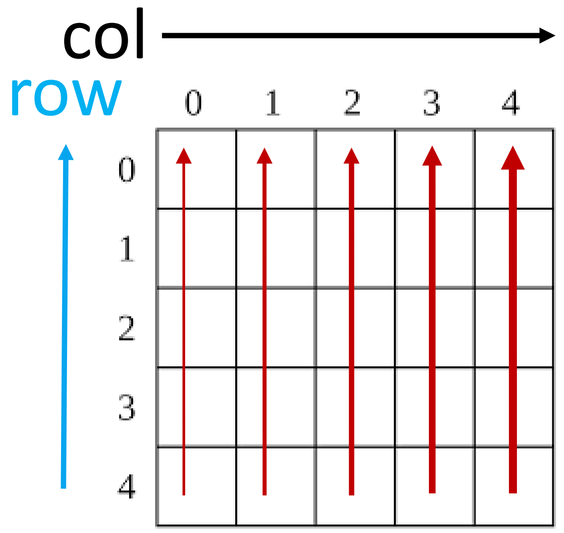


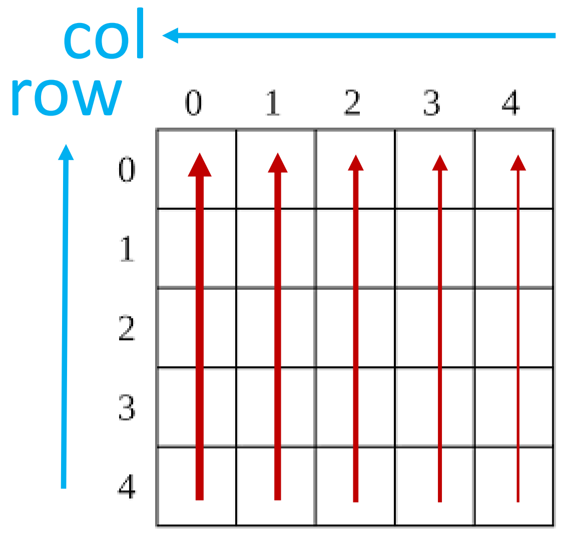


```
for(col=0; col<m[0].length; col++)
  for(row=0; row<m.length; row++)
    System.out.println(m[row][col]);
```

```
for(col=m[0].length-1; col>=0; col--)
  for(row=0; row<m.length; row++)
    System.out.println(m[row][col]);
```

2-D Array Indexing for Traversal
int row, col; int[][] m=new int[5][5];

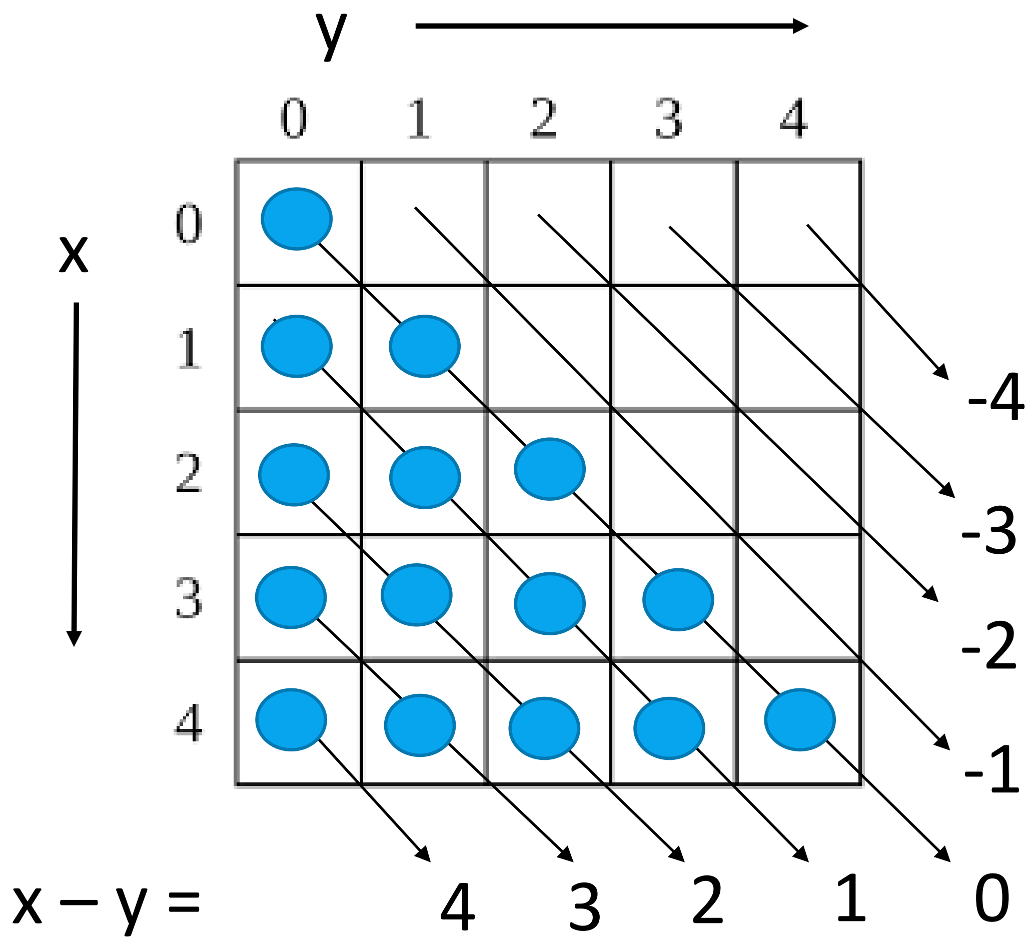




```
for(col=0; col<m[0].length; col++)
  for(row=m.length-1; row>=0; row--)
    System.out.println(m[row][col]);
```

```
for(col=m[0].length-1; col>=0; col--)
  for(row=m.length-1; row>=0; row--)
    System.out.println(m[row][col]);
```

Partial Array Traversal



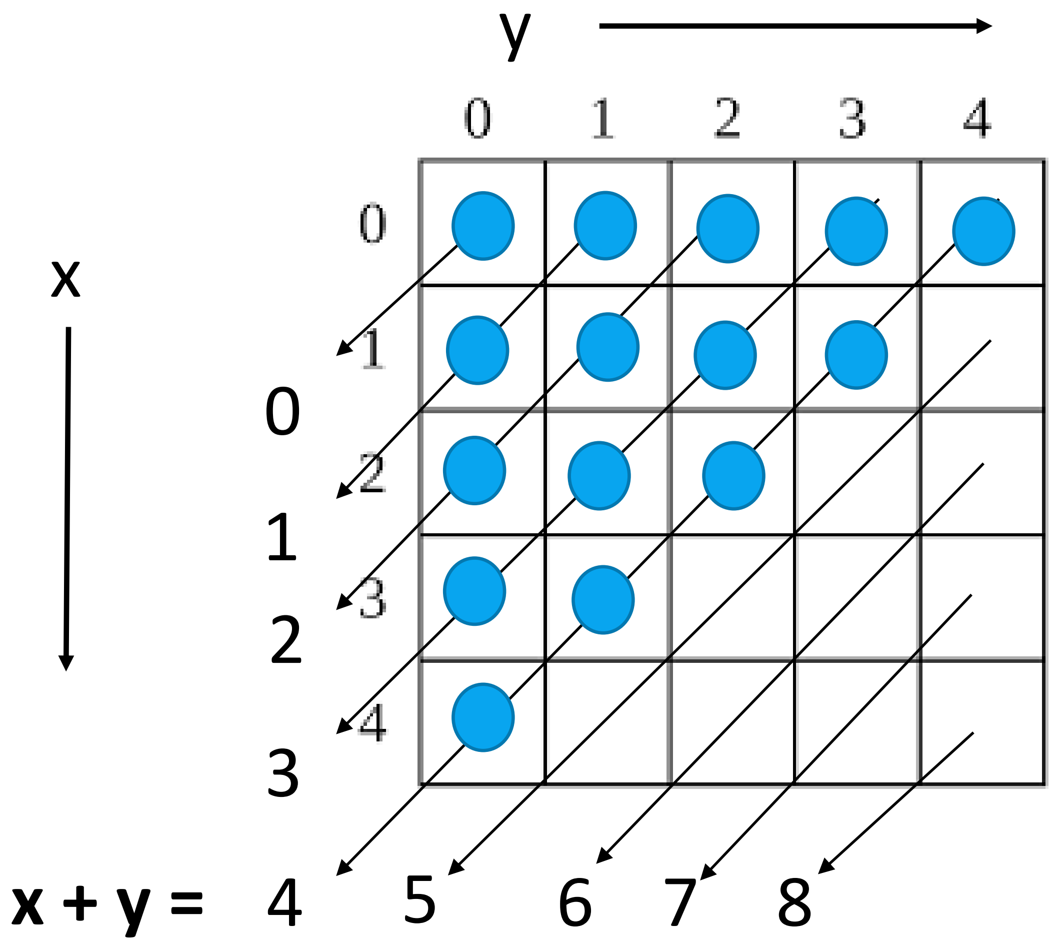
```
for (int I = 0; i<m.length; i++)
   for (int j =0; j< i +1;  j ++)
      { /* do something */}
```

Index:

Stop Condition: j stop at j = i.

i – j = 0;

Partial Array Traversal



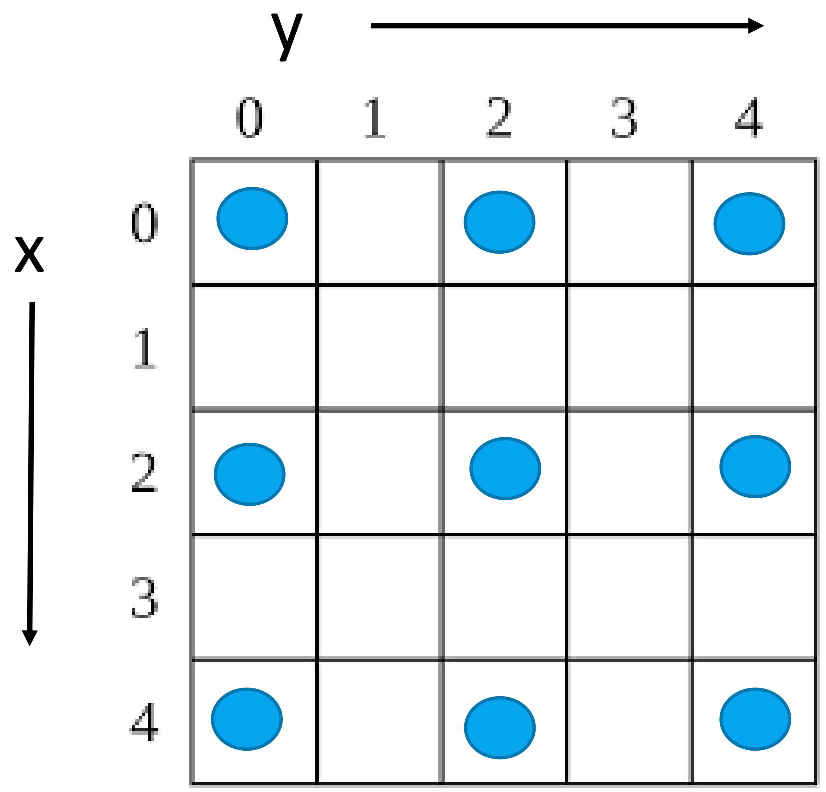
```
for (int i = 0; i<m.length; i++)
   for (int j =m.length-1-i; j>=0;  j --)
      { /* do something */}
```

Index:

Start Condition: i + j = m.length-1;

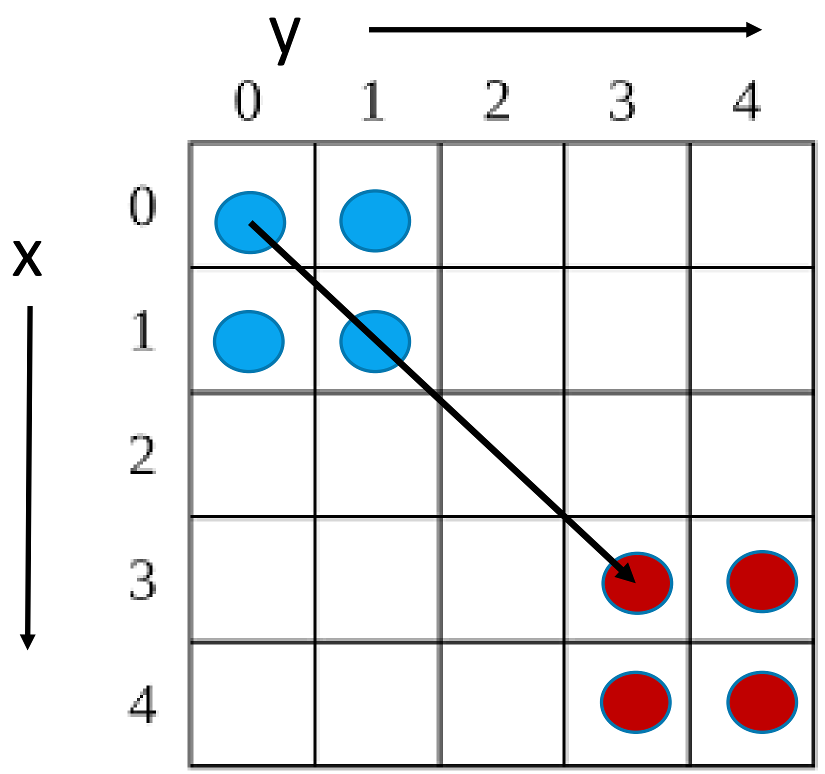
i + j = 4;

Partial Array Traversal

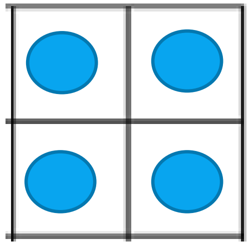


```
for (int i = 0; i<m.length; i+=2)
  for (int j =0; j<m[0].length;  j+=2)
    { /* do something */}
```

Vector Operation (Area Copy)



```
for (int i = 0; i<2; i++)
   for (int j=0; j<2; j++)
      m[3+i][3+j] = m[0+i][0+j];
         /* 0 is not needed */
```

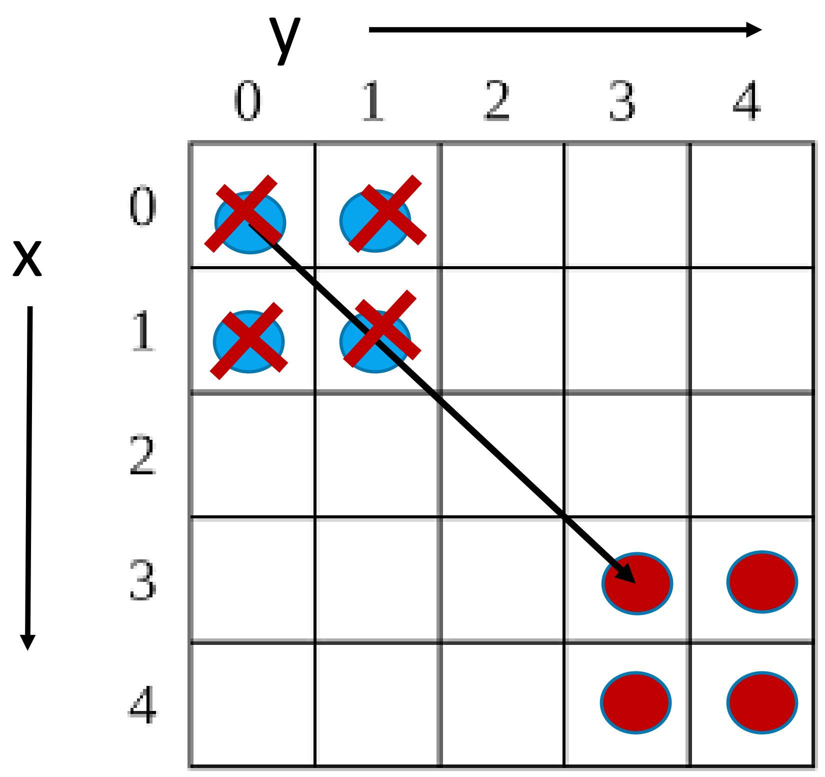


Area to be Copied: **2 x 2** block

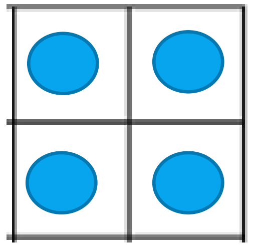
From (0, 0) to (3, 3)

If there is overlap, we need to put the area into a buffer first.

Vector Operation (Area Move)



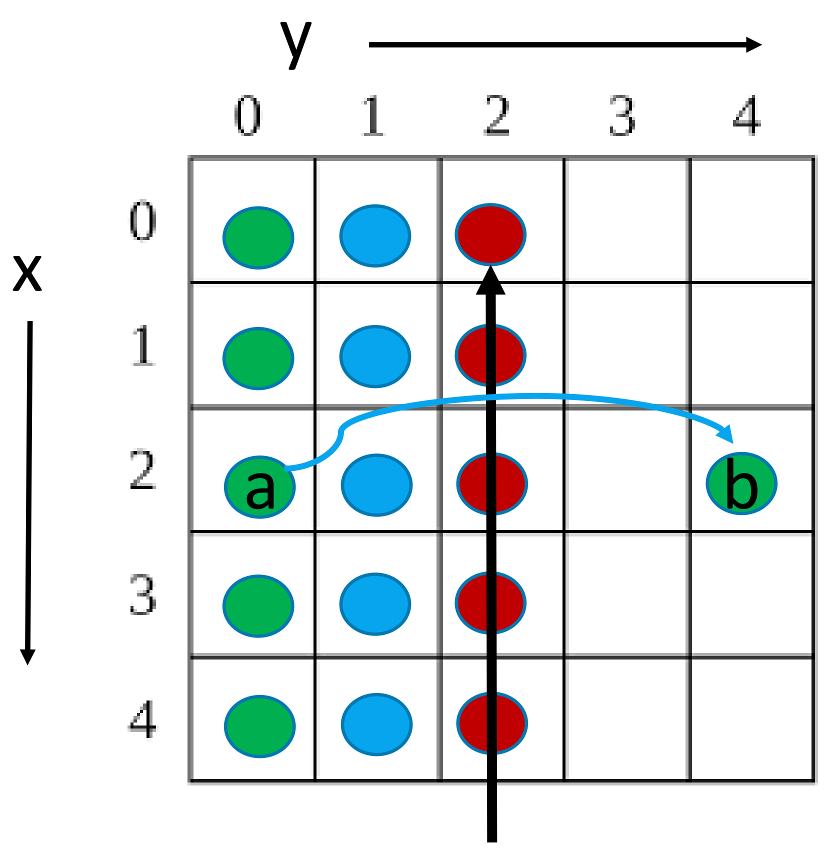
```
for (int i = 0; i<2; i++)
  for (int j=0; j<2; j++)
    m[3+i][3+j] = m[0+i][0+j];
for (int i = 0; i<2; i++)
  for (int j=0; j<2; j++)
    m[0+i][0+j]=0;
```



Area to be Copied: **2 x 2** block

From (0, 0) to (3, 3)

Flip



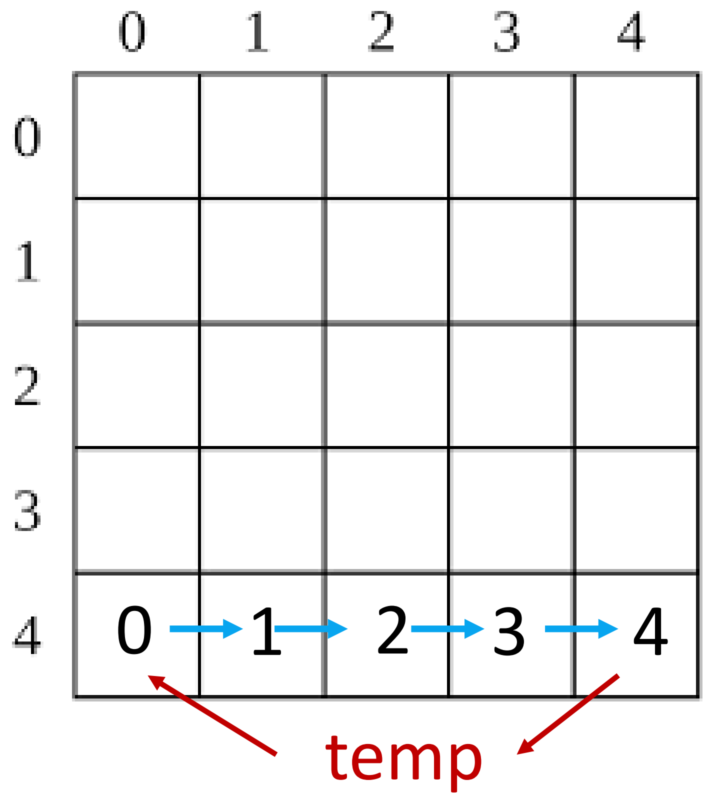
Symmetric Line for Flipping:
j = 2;   // j = m.length/2

(a + b)/2 = 2;
b == 4-a;   // b = m.length –a-1;

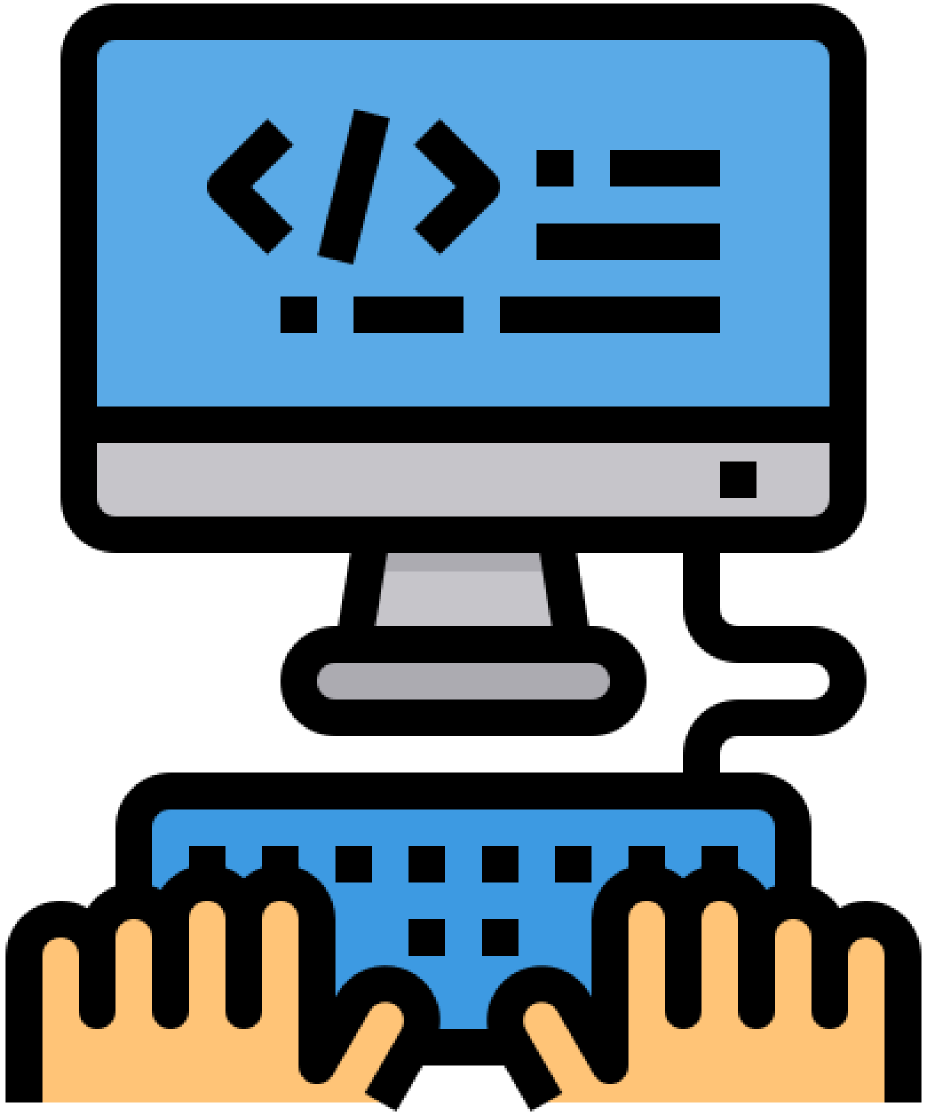
```
for (int i=0; i<m.length; i++){
  for (int j=0; j<m.length/2; j++){
     m[i][m.length-j-1] = m[i][j];
  }
}
```

Area Shift

```
for (int i=0; i<m.length; i++){
    int temp = m[i][m.length-1];
    for (int j=m[0].length-2; j<=0; j--){
        m[i][j+1] = m[i][j];
    }
    m[i][0] =temp;
}
```

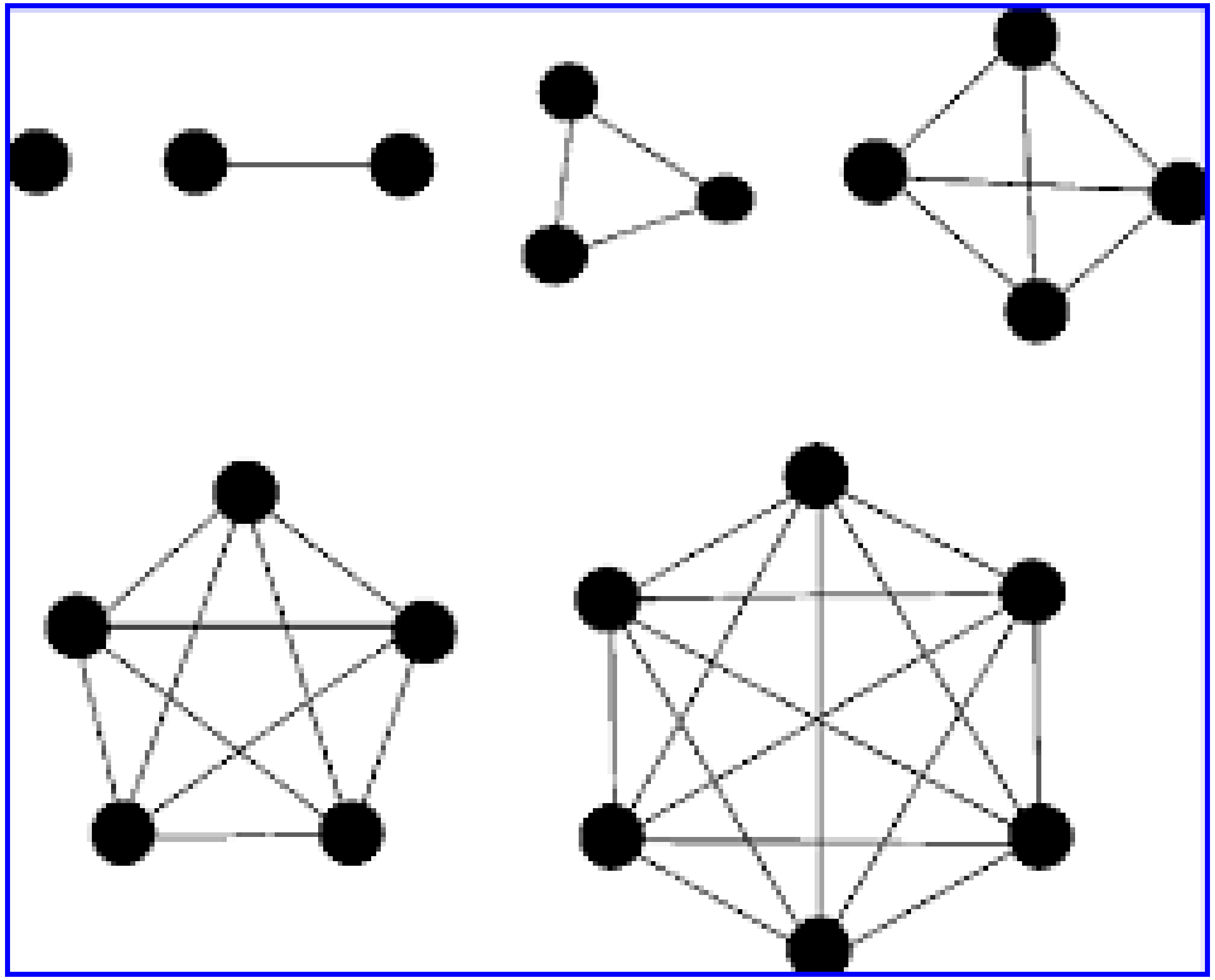


One Row after Shift: 4 0 1 2 3



2D Array Processing

Section 5



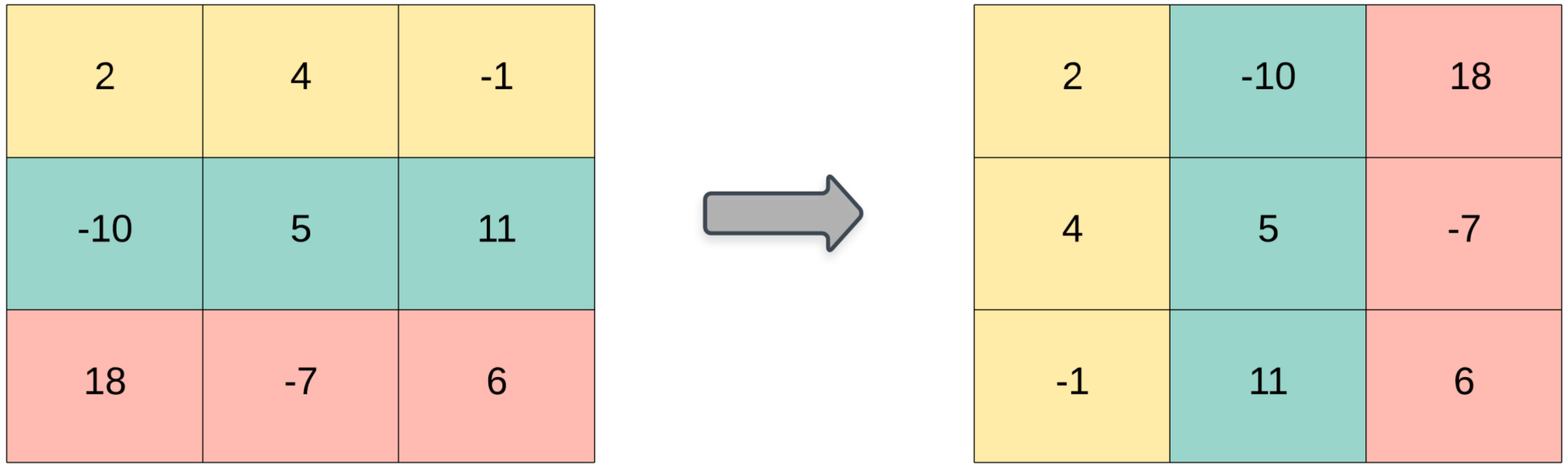
K Graph Problems

Combination Problem

```
static String s = "ABCDE";

public static void combination(String s){
   for (int i=0; i<s.length()-1; i++){
        for (int j=i+1; j<s.length(); j++){
            String x = ""+s.charAt(i)+s.charAt(j);
            System.out.println(x);
        }
   }
}
```

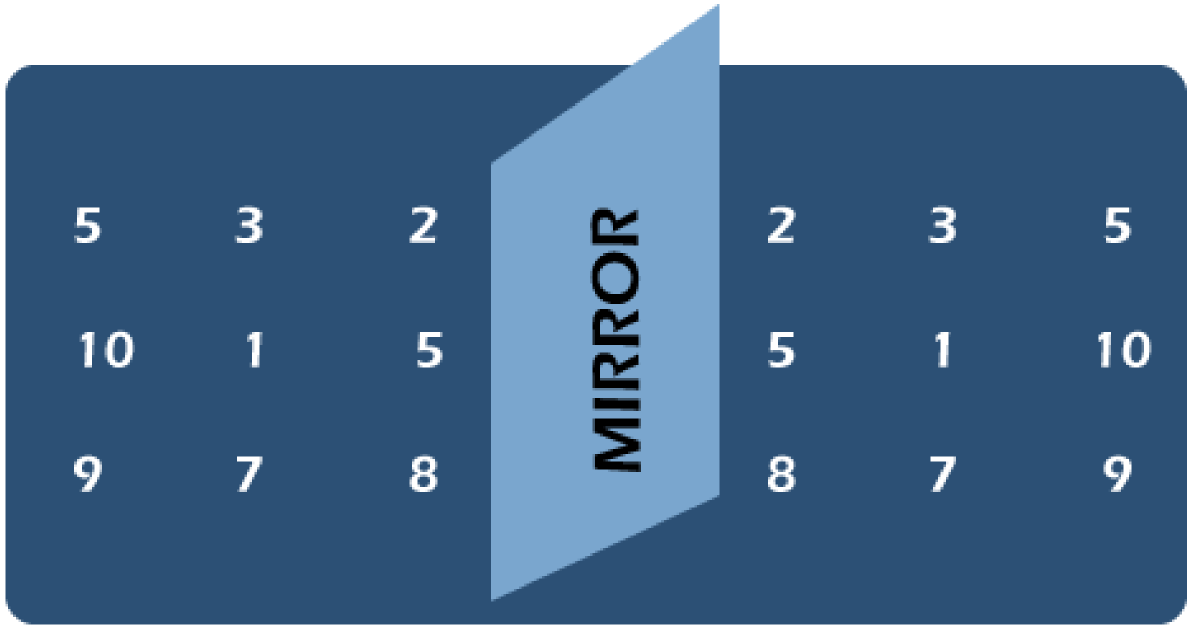
Transpose



2	4	-1
-10	5	11
18	-7	6

2	-10	18
4	5	-7
-1	11	6

Mirror



Mirror Image of a 2D Array