Example: Per-Student Averages

```java
class ArrayEx5 {
    public static void main(String[] args) {
        double[][] scores = {{95, 82, 13, 96}, {88, 93, 71, 75}, {79, 76, 78, 96}, {89, 92, 90}, {99, 90, 89, 98}};
        double average = 0;
        for (int row = 0; row < scores.length; row++) {
            for (int col = 0; col < scores[row].length; col++) {
                average = average + scores[row][col];
            }
            average = average / scores.length;
            System.out.println("average of row " + row + " is " + average);
        }
    }
}
```

Recap: Two-Dimensional Arrays

- In Java, 2D array implemented internally as array of arrays
- but externally syntax of 2D array may seem easier to use

Selection sort

Let's say want to sort array values in increasing order
- one way to approach problem is to use algorithm called selection sort
- Start by setting pointer to first element in array
- This is where smallest value in array will be placed

Example: Per-Quiz Averages

```java
class ArrayEx4 {
    public static void main(String[] args) {
        double[] scores = {88, 71, 84, 78, 73, 71, 84, 78, 73, 71};
        double average;
        for (int row = 0; row < scores.length; row++) {
            average = 0;
            for (int col = 0; col < scores[row].length; col++) {
                average = average + scores[row][col];
            }
            average = average / scores[row].length;
            System.out.println("average of column " + col + " is " + average);
        }
    }
}
```

Recap: CokeEmpire

- What does this return?
```java
CokeEmpire.getNumberOfCans();
CokeEmpire.getNumberOfCans();
```

Recap: CokeEmpire

- Apply method overload
- CokeEmpire.getNumberOfCans();
- CokeEmpire.getNumberOfCans(int location);
- CokeEmpire.getNumberOfCans(location: "Main Library" val = 201.25)
- CokeEmpire.getNumberOfCans(location: "Chan Centre" val = 201.25)

Recap: CokeEmpire

- Apply method overload
- CokeEmpire.getNumberOfCans();
- CokeEmpire.getNumberOfCans(int location);
Selection sort

Let's say want to sort array values in increasing order.

one way to approach problem is to use algorithm called selection sort.

Start by setting pointer to first element in array.

this is where smallest value in array will be placed.

Then look at every value in this unsorted array.

find minimum value.

The smallest value so far is 16.

Its index is 0.

Let's say want to sort array values in increasing order.

one way to approach problem is to use algorithm called selection sort.

Start by setting pointer to first element in array.

this is where smallest value in array will be placed.

Then look at every value in this unsorted array.

find minimum value.

The smallest value so far is 3.

Its index is 1.

Let's say want to sort array values in increasing order.

one way to approach problem is to use algorithm called selection sort.

Start by setting pointer to first element in array.

this is where smallest value in array will be placed.

Then look at every value in this unsorted array.

find minimum value.

The smallest value so far is 8.

Its index is 3.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.

At this point we know.

smallest number in array is in first element (index 0).

first element is sorted.

rest of array remains unsorted.

Now select second element of array to be location which will hold next smallest value.

In other words, do everything again to unsorted part of array.

in this case, all but first element.

Now first two elements of array are sorted.

Select third element of array to be location of next smallest value.

Search unsorted portion of array for that value, just like before.
Selection sort

- Now first two elements of array are sorted
- Select third element of array to be location of next smallest value
- Search unsorted portion of array for that value, just like before
- Again, swap values
- then do whole thing again

The smallest value so far is 16
Its index is 3

Are we done?
- could select last element of array (index 4)
- but all of array except for last element is already sorted
- so last element is largest value in array
- and that's the right place
- Yes, array is sorted, and we're done
- no need to select last element

Selection sort

- Now first two elements of array are sorted
- Select third element of array to be location of next smallest value
- Search unsorted portion of array for that value, just like before
- Again, swap values
- then do whole thing again
- Swap again
- not actually necessary in this case
- but we follow algorithm

Selection sort

- The smallest value so far is 16
Its index is 3

Are we done?
- could select last element of array (index 4)
- but all of array except for last element is already sorted
- so last element is largest value in array
- and that's the right place
- Yes, array is sorted, and we're done
- no need to select last element

Selection sort

- Now first two elements of array are sorted
- Select third element of array to be location of next smallest value
- Search unsorted portion of array for that value, just like before
- Again, swap values
- then do whole thing again
- Swap again
- not actually necessary in this case
- but we follow algorithm

Selection sort

- Now first two elements of array are sorted
- Select third element of array to be location of next smallest value
- Search unsorted portion of array for that value, just like before
- Again, swap values
- then do whole thing again
- Swap again
- not actually necessary in this case
- but we follow algorithm

Selection sort

- The smallest value so far is 16
Its index is 3

Are we done?
- could select last element of array (index 4)
- but all of array except for last element is already sorted
- so last element is largest value in array
- and that's the right place
- Yes, array is sorted, and we're done
- no need to select last element
Selection sort
Selection sort
Selection sort
Selection sort
Selection sort
Selection sort
Selection sort
Selection sort

// selection sort
public class SortTest1{
  public static void main(String[] args) {
    int[] numbers = {16,3,19,8,12};
    int min, temp;
    for (int i = 0; i < numbers.length; i++) {
      temp = numbers[i];
      for (int j = i+1; j < numbers.length; j++) {
        if (numbers[j] < temp) {
          //swap two values in the array
          numbers[min] = temp;
          numbers[i] = numbers[min];
        }
      }
    }
    System.out.println("Printing sorted result");
    System.out.println(numbers);
  }
}

16 319 812
i j
 0 1
System.out.println("Printing sorted result");
System.out.println(numbers[0]);
Selection sort
public class SortTest1{
  public static void main(String[] args) {
    // the array to be sorted
    int[] numbers = {16, 3, 19, 8, 12};
    int min = 0, temp = 0;
    System.out.println("Printing sorted result");
    for (i = 0; i < numbers.length; i++) {
      System.out.println(numbers[i]);
      for (j = i + 1; j < numbers.length; j++) {
        if (numbers[j] < numbers[min]) {
          min = j;
          temp = numbers[i];
          numbers[i] = numbers[j];
          numbers[j] = temp;
        }
      }
    }
    //print the sorted array
    System.out.println("Sorted array: ");
    for (i = 0; i < numbers.length; i++) {
      System.out.println(numbers[i]);
    }
  }
}

Selection sort

public class SortTest1
{
  public static void main(String[] args)
  {
    int[] numbers = {16, 3, 19, 8, 12};
    int min = 0;
    int temp;
    System.out.println("Printing sorted result");
    for (int i = 0; i < numbers.length; i++)
    {
      System.out.println(numbers[i]);
    }
    
    for (int i = 0; i < numbers.length - 1; i++)
    {
      int j = i + 1;
      while (j < numbers.length)
      {
        if (numbers[j] < numbers[min])
        {
          min = j;
        }
        j++;
      }
      temp = numbers[i];
      numbers[i] = numbers[min];
      numbers[min] = temp;
    }
    
    System.out.println("Printing sorted result");
    for (int i = 0; i < numbers.length; i++)
    {
      System.out.println(numbers[i]);
    }
  }
}

Selection sort

public class SortTest1
{
  public static void main(String[] args)
  {
    int[] numbers = {16, 3, 19, 8, 12};
    int min = 0;
    int temp;
    System.out.println("Printing sorted result");
    for (int i = 0; i < numbers.length; i++)
    {
      System.out.println(numbers[i]);
    }
    
    for (int i = 0; i < numbers.length - 1; i++)
    {
      int j = i + 1;
      while (j < numbers.length)
      {
        if (numbers[j] < numbers[min])
        {
          min = j;
        }
        j++;
      }
      temp = numbers[i];
      numbers[i] = numbers[min];
      numbers[min] = temp;
    }
    
    System.out.println("Printing sorted result");
    for (int i = 0; i < numbers.length; i++)
    {
      System.out.println(numbers[i]);
    }
  }
}

Selection sort

public class SortTest1
{
  public static void main(String[] args)
  {
    int[] numbers = {16, 3, 19, 8, 12};
    int min = 0;
    int temp;
    System.out.println("Printing sorted result");
    for (int i = 0; i < numbers.length; i++)
    {
      System.out.println(numbers[i]);
    }
    
    for (int i = 0; i < numbers.length - 1; i++)
    {
      int j = i + 1;
      while (j < numbers.length)
      {
        if (numbers[j] < numbers[min])
        {
          min = j;
        }
        j++;
      }
      temp = numbers[i];
      numbers[i] = numbers[min];
      numbers[min] = temp;
    }
    
    System.out.println("Printing sorted result");
    for (int i = 0; i < numbers.length; i++)
    {
      System.out.println(numbers[i]);
    }
  }
}
Selection sort
// selection sort
public class SortTest1{
    public static void main(String[] args) {
        int[] numbers = {16, 3, 19, 8, 12};
        // find the smallest value in the remainder of the array
        int min = i;
        for (int j = i + 1; j < numbers.length; j++) {
            if (numbers[j] < numbers[min]) {
                min = j;
            }
        }
        // select location of next sorted value
        int temp = numbers[i];
        numbers[i] = numbers[min];
        numbers[min] = temp;
    }
}

Printing sorted result
Tracing with the Debugger