EXAMPLE

```
import chn.util.*;
import apcslib.Format;
import java.util.*;

//Driver program for the Sorts class.
public class SortingMethods
{
  private ConsoleIO console;
  private int[] myArray;
  private Sorts mySorts;

  //Constructor for the SortStep object
  public SortingMethods()
  {
    console = new ConsoleIO();
    mySorts = new Sorts();
  }

  /**
   *  Asks the user to select a sorting algorithm, fills the array
   *  with an amount of random integer data chosen by the user, calls
   *  the sorting algorithm, and gives an option of printing out the
   *  data after it has been sorted.
   */
  public void sortMenu()
  {
    String choice;
    String print;

    do
    {
      System.out.println();
      System.out.println("Sorting algorithm menu");
      System.out.println();
      System.out.println("(1) Bubble sort");
      System.out.println("(2) Selection sort");
      System.out.println("(3) Insertion sort");
      System.out.println("(Q) Quit");
      System.out.println();
      System.out.print("Choice ---> ");
      choice = console.readLine() + " ";
      if ('1' <= choice.charAt(0) && choice.charAt(0) <= '3')
      {
        System.out.println();
        System.out.print("How many numbers do you wish to generate? ");
        int numInts = console.readInt();
        System.out.print("Largest integer to generate? ");
        int largestInt = console.readInt();
        fillArray(numInts, largestInt);

        switch (choice.charAt(0))
        {
            case '1':
              mySorts.bubbleSort(myArray);
              break;
            case '2':
              mySorts.selectionSort(myArray);
              break;
            case '3':
              mySorts.insertionSort(myArray);
              break;
        }
        System.out.println();
        System.out.print("Print list to screen (y/n)? ");
        print = console.readLine();
        if (print.charAt(0) == 'y' || print.charAt(0) == 'Y') this.screenOutput();
        System.out.println();
        System.out.print("Hit return to continue");
```

```
        console.readLine();
      }
    } while (choice.charAt(0) != 'Q' && choice.charAt(0) != 'q');
  }

  /**
   *  Initializes myArray with random integers in the range: 1 - largestInt
   * @param  largestInt  largest possible random integer to create
   */
  private void fillArray(int numInts, int largestInt)
  {
    myArray = new int[numInts];
    Random randGen = new Random();
    for (int loop = 0; loop < myArray.length; loop++)
    {
      myArray[loop] = randGen.nextInt(largestInt) + 1;
    }
  }

    //prints out the contents of the array in tabular form, 12 columns
    private void screenOutput()
    {
        for (int loop = 0; loop < myArray.length; loop++)
        {
            if ( loop!=0 && loop%12==0 ) System.out.println();
            System.out.print(Format.right(myArray[loop], 6));
        }
        System.out.println();
    }

    //Provides a main method for access to the sorting menu
    public static void main(String[] args)
    {
        SortingMethods doSorts = new SortingMethods();
        doSorts.sortMenu();
    }
}
```

```
/**
*    1. Bubble sort
*    2. Selection sort
*    3. Insertion sort
*/
public class Sorts
{
    //Interchanges two elements in an integer array
    public void swap(int[] list, int a, int b)
    {
        int c = list[a];
        list[a] = list[a = b];
        list[b] = c;
    }

    //Sorts an array in ascending order using a bubble sort algorithm
    public void bubbleSort(int[] list)
    {
        for (int outer = 0; outer < list.length - 1; outer++)
        {
            for (int inner = 0; inner < list.length - outer - 1; inner++)
            {
                if (list[inner] > list[inner + 1]) swap(list, inner, inner + 1);
            }
        }
    }
```

```
    //Sorts an array in ascending order using a selection sort algorithm
    public void selectionSort(int[] list)
    {
        int min;
        int temp;
        for (int outer = 0; outer < list.length - 1; outer++)
        {
            min = outer;
            for (int inner = outer + 1; inner < list.length; inner++)
            {
                if (list[inner] < list[min])
                {
                    min = inner;
                }
            }
            swap(list, outer, min);
        }
    }

    //Sorts an array in ascending order using an insertion sort algorithm
    public void insertionSort(int[] list)
    {
        for (int outer = 1; outer < list.length; outer++)
        {
            int position = outer;
            int key = list[position];

            // Shift larger values to the right
            while (position > 0 && list[position - 1] > key)
            {
                list[position] = list[position - 1];
                position--;
            }
            list[position] = key;
        }
    }
}
```

OUTPUT:

Sorting algorithm menu

(1) Bubble sort
(2) Selection sort
(3) Insertion sort
(Q) Quit

Choice ---> 2

How many numbers do you wish to generate? 50
Largest integer to generate? 10000

Print list to screen (y/n)? y

292 483 667 941 1171 1289 1329 1430 1447 2201 2214 2234
2385 2564 2969 3020 3334 3797 3967 3974 4012 4038 4364 4422
4455 4806 5039 5136 5336 5430 5827 6127 6790 7185 7206 7366
7527 7572 7785 8031 8333 8511 8622 8878 9212 9288 9439 9647
9897 9909

Hit return to continue

Sorting algorithm menu

(1) Bubble sort
(2) Selection sort
(3) Insertion sort
(Q) Quit

Choice ---> 3

How many numbers do you wish to generate? 20
Largest integer to generate? 200

Print list to screen (y/n)? y

13 14 17 36 38 65 79 92 109 111 113 131
132 138 156 159 161 161 181 191

Hit return to continue

Sorting algorithm menu

(1) Bubble sort
(2) Selection sort
(3) Insertion sort
(Q) Quit

Choice ---> q