ARRAYSORTER.CPP

```cpp
#include "ArraySorter.h"

using namespace std;

ArraySorter::ArraySorter( const int aArrayOfNumbers[], unsigned int aArraySize )
{
    // copy array into sorter
    fArrayOfNumbers = new int[aArraySize];

    for ( unsigned int i = 0; i < aArraySize; i++ )
    {
        fArrayOfNumbers[i] = aArrayOfNumbers[i];
    }

    fArraySize = aArraySize;
}

ArraySorter::~ArraySorter()
{
    // delete memory associated with array
    delete [] fArrayOfNumbers;
}

void ArraySorter::stepCompleted(std::ostream &aOStream)
{
    aOStream << "State: "<< *this << endl;
}


void ArraySorter::swapElements( unsigned int aSourcIndex, unsigned int aTargetIndex)
{
    int temp = at(aSourcIndex);
    fArrayOfNumbers[aSourcIndex] = at(aTargetIndex);
    fArrayOfNumbers[aTargetIndex] = temp;
}
```

```
const unsigned int ArraySorter::at(unsigned int aIndex) const
{
    if (aIndex > fArraySize)
    {
        throw range_error("The index's length is not within the array size. Please modify the index");
    }
    return  fArrayOfNumbers[aIndex];
}


const unsigned int ArraySorter::getRange() const
{
    return fArraySize;
}



void ArraySorter::sort(ostream& aOStream)
{
    stepCompleted(aOStream);
}



ostream& operator<<(std::ostream& aOStream, const ArraySorter& aObject)
{
    aOStream << "[";
    for (unsigned int i = 0; i < aObject.getRange(); i++)
    {
        aOStream << aObject.at(i);
        if (i < aObject.getRange() - 1)
            aOStream << ", ";
    }
    aOStream << "]";
    return aOStream;
}
```

SELECTIONSORT.CPP

```
#include "SelectionSort.h"

using namespace std;


SelectionSort::SelectionSort(int aArrayOfNumbers[], unsigned int aArraySize):
ArraySorter::ArraySorter(aArrayOfNumbers, aArraySize)
{

}

void SelectionSort::sort(std::ostream& aOStream)
{
    for (unsigned int i = 0; i < getRange() - 1; i++)
    {
        unsigned int b = i;
        for (unsigned int a = i + 1; a < getRange(); a++)
        {
            if (at(a) < at(b))
            {
                b = a;
            }
        }
         swapElements(i, b);
         stepCompleted(aOStream);
    }
}
```

INSERTIONSORT.CPP

```
#include "InsertionSort.h"

using namespace std;

InsertionSort::InsertionSort(int aArrayOfNumbers[], unsigned int aArraySize) :
ArraySorter::ArraySorter(aArrayOfNumbers, aArraySize)
{

}

void InsertionSort::sort(std::ostream& aOStream)
{
    for (unsigned int i = 0; i < getRange()-1; i++)
    {
        for (unsigned int b = i+1; b > 0; b--)
        {
            if (at(b) < at(b-1))
            {
                swapElements(b, b-1);
            }
        }
        stepCompleted(aOStream);
        }
    }
```

```
Test selection sort:
[34, 2, 890, 40, 16, 218, 20, 49, 10, 29]
State: [2, 34, 890, 40, 16, 218, 20, 49, 10, 29]
State: [2, 10, 890, 40, 16, 218, 20, 49, 34, 29]
State: [2, 10, 16, 40, 890, 218, 20, 49, 34, 29]
State: [2, 10, 16, 20, 890, 218, 40, 49, 34, 29]
State: [2, 10, 16, 20, 29, 218, 40, 49, 34, 890]
State: [2, 10, 16, 20, 29, 34, 40, 49, 218, 890]
State: [2, 10, 16, 20, 29, 34, 40, 49, 218, 890]
State: [2, 10, 16, 20, 29, 34, 40, 49, 218, 890]
State: [2, 10, 16, 20, 29, 34, 40, 49, 218, 890]
Test insertion sort:
[34, 2, 890, 40, 16, 218, 20, 49, 10, 29]
State: [2, 34, 890, 40, 16, 218, 20, 49, 10, 29]
State: [2, 34, 890, 40, 16, 218, 20, 49, 10, 29]
State: [2, 34, 40, 890, 16, 218, 20, 49, 10, 29]
State: [2, 16, 34, 40, 890, 218, 20, 49, 10, 29]
State: [2, 16, 34, 40, 218, 890, 20, 49, 10, 29]
State: [2, 16, 20, 34, 40, 218, 890, 49, 10, 29]
State: [2, 16, 20, 34, 40, 49, 218, 890, 10, 29]
State: [2, 10, 16, 20, 34, 40, 49, 218, 890, 29]
State: [2, 10, 16, 20, 29, 34, 40, 49, 218, 890]
Program ended with exit code: 0
```