CIS 285: Software Engineering Tools

Final Exam Winter 21

Student Name: Kevin Marzolo

1. Given the Java class ‘SelectionSort’ as below.
   1. Develop and execute three unit tests 10pts
      1. testPositive – testing a list of all positive integers
      2. testNegatives – testing a list of all negative integers
      3. testMixed – testing a list of positive, negative and zero
   2. Can above tests pass? If not, what change should be made in Class ‘SelectionSort’ to make them pass. 10pts
   3. Write git command to 10pts
      1. Add a local repository
      2. Add initial production code and test code to staging phase
      3. Commit files
      4. After making changes in codes, show differences
      5. Add and commit updated production code

Before changes:

**public** **class** SelectionSort {

/\* Selection Sort function \*/

**public** **static** **void** sort( **int** arr[] ){

**int** N = arr.length;

**int** i, j, pos, temp;

**for** (i = 0; i < N; i++)

{

pos = j;

**for** (j = i+1; j < N-1; j++)

{

**if** (arr[j] < arr[pos])

{

pos = i;

}

}

/\* Swap arr[i] and arr[pos] \*/

temp = arr[i];

arr[i] = arr[pos];

arr[pos]= temp;

}

}

}

After changes:

**public** **class** SelectionSort {

/\* Selection Sort function \*/

**public** **static** **void** sort( **int** arr[] ){

**int** N = arr.length;

**int** i, j, pos, temp;

**for** (i = 0; i < N; i++)

{

pos = i;

**for** (j = i+1; j < N; j++)

{

**if** (arr[j] < arr[pos])

{

pos = j;

}

}

/\* Swap arr[i] and arr[pos] \*/

temp = arr[i];

arr[i] = arr[pos];

arr[pos]= temp;

}

}

}

The original code would not pass because if a switch needs to be made, the program is assigning i to pos, then switching them i and pos, essentially swapping the same element. Also, the second for loop doesn’t search the entire array; stopping just before the last element. I made 2 changes, I initialized pos equal to i and when a switch needed to be made, I assigned pos to j. I also made sure the second for loop searched that last item.

Test code:

**import** **static** org.junit.jupiter.api.Assertions.\*;

**import** org.junit.jupiter.api.Test;

**class** TestSelectionSort {

**private** SelectionSort temp1;

@Test

**public** **void** test() {

testPositive();

testNegative();

testMixed();

}

**public** **void** testPositive()

{

**int**[] arr = {8, 9, 7, 10, 2};

**int**[] sorted = {2, 7, 8, 9, 10};

SelectionSort temp1 = **new** SelectionSort();

temp1.sort(arr);

*assertArrayEquals*(arr, sorted);

}

**public** **void** testNegative()

{

**int**[] arr = {-8, -9, -7, -10, -2};

**int**[] sorted = {-10, -9, -8, -7, -2};

SelectionSort temp1 = **new** SelectionSort();

temp1.sort(arr);

*assertArrayEquals*(arr, sorted);

}

**public** **void** testMixed()

{

**int**[] arr = {-8, 9, -7, -10, 2};

**int**[] sorted = {-10, -8, -7, 2, 9};

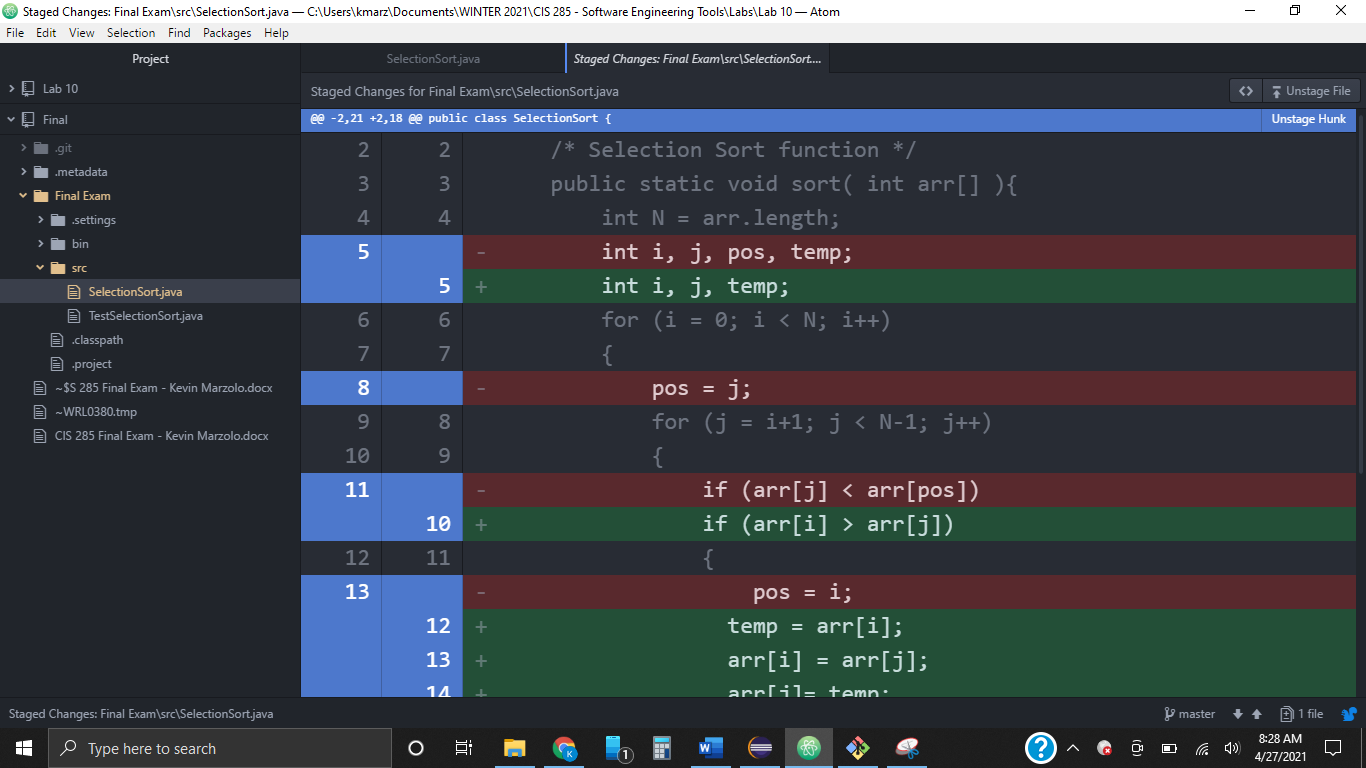
SelectionSort temp1 = **new** SelectionSort();

temp1.sort(arr);

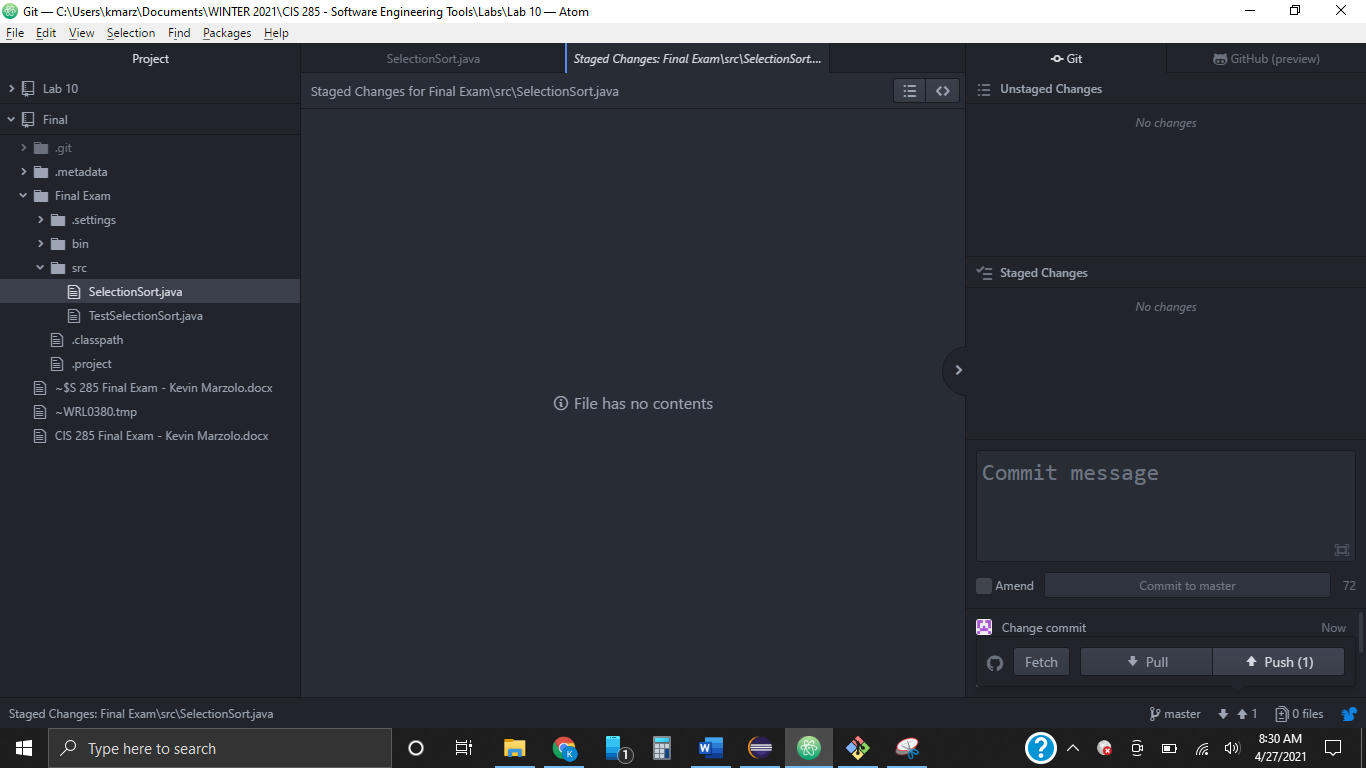
*assertArrayEquals*(arr, sorted);

}

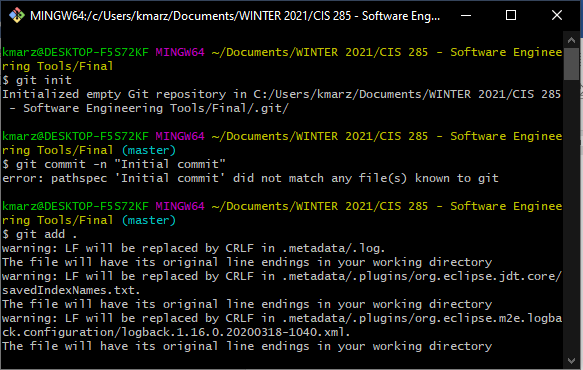
}

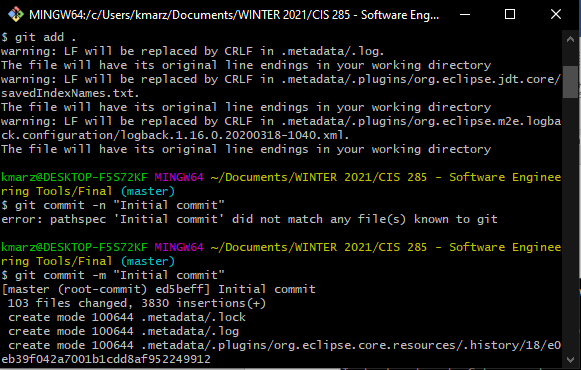


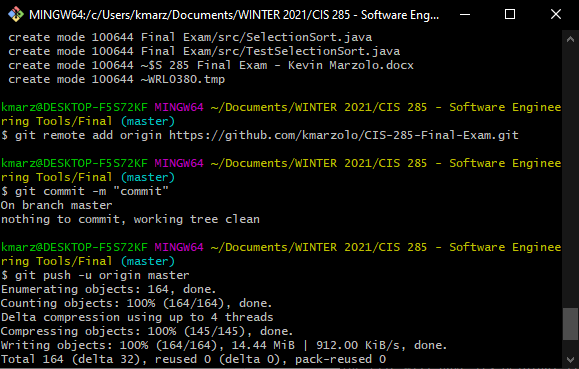
Changes were made in Atom. Above you can see the changes made. Below is the push request in the bottom right corner.



Git commands used







1. A marketing plan (MP) software takes salesperson’s target group in terms of income, house value, and zip code, the software search internal patron database and returns all name and address that satisfy requirement. The user can export and print the result list and statistical report of the list.
2. Write 10 functional requirements 10pts
3. Write 5 non-functional requirements 10pts
4. Draw a UML use case diagram to complement the functional requirement (2 or 3 use cases are enough). Each use case must contain entry condition, exit condition, flow of event, constraints. 10pts
5. Draw an analysis level class diagram but classes involved in e) and f) must contain both attribute and operation . 10pts
6. Draw one UML sequence diagram 10pts
7. Choose one class in d) and define states and then draw a state diagram for that class, please note, you must first provide definition of the each state in terms of attribute before draw state diagram. 10pts
8. Draw an activity diagram 10pts

Functional Requirements

1. Generate a list of targets
2. Generate statistical report
3. Export list of targets
4. Export statistical report
5. Print list of targets
6. Print statistical report
7. Sort by income, house value, zip code, etc.
8. Search for specific target(s)
9. Delete targets
10. Add targets

Non-Functional Requirements

1. Generating list of targets should take no longer than 3 seconds
2. Searching for targets should take less than 1 second
3. Server should be available 24 hours a day
4. Exporting reports should take less than 1 second
5. Keep a database of all targets in a group

Use cases used in diagrams:

Name: Generate a list of targets

Entry condition:

The salesperson wants to generate a report

There are already patrons in the patron database

Exit condition:

A report is displayed to the salesperson

Flow of events:

The salesperson clicks the “generate report” button

The program goes into the patron database

The program writes every patron information into a pdf file

The program displays the pdf file

Constraints:

None

Name: Export list of targets

Entry condition:

A report has been generated and displayed

The salesperson wants to save the report

Exit condition:

The report is saved onto the computer hard drive

Flow of events:

The salesperson clicks the “save” button

The program finds space in the computer’s hard drive

The program downloads the report into the computer’s hard drive

Constraints:

The program cannot access any other data in the hard drive

Name: Search for specific targets

Entry condition:

There are already patrons in the patron database

The salesperson wants to find patron(s) based on income, zip code, etc.

Exit condition:

The program displays all patrons that match the description

Flow of events:

The salesperson enters in the search criteria into the search bar

The salesperson clicks the “search” button

The program searches the patron database for patrons that match the criteria

The program displays all of the matching patrons

Constraints:

None