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, temp;

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

{

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

{

**if** (arr[i] > arr[j])

{

temp = arr[i];

arr[i] = arr[j];

arr[j]= temp;

}

}

}

}

}

I made a couple of changes. I got rid of the variable pos, since it was not needed. Also in the original code, there was a switch being made at every iteration of the for loop. Instead, I made it so that a switch is only made whenever there is a number smaller than the number in the front.

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();

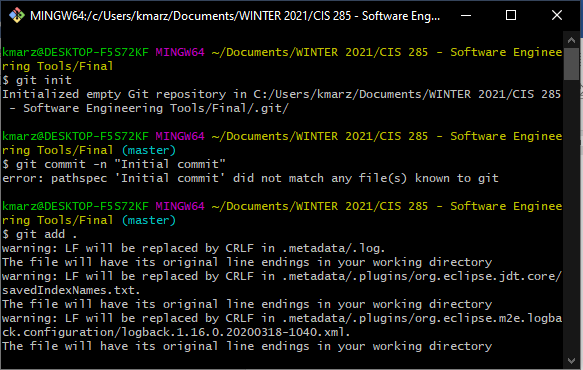
temp1.sort(arr);

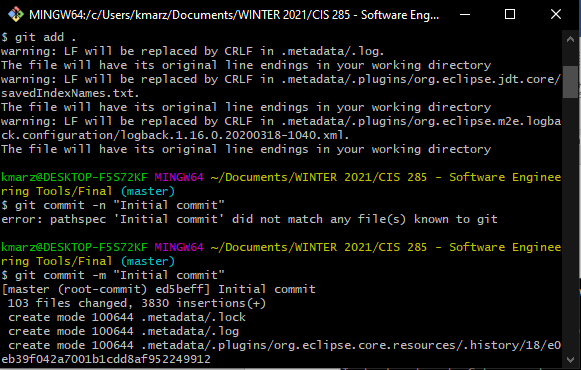
*assertArrayEquals*(arr, sorted);

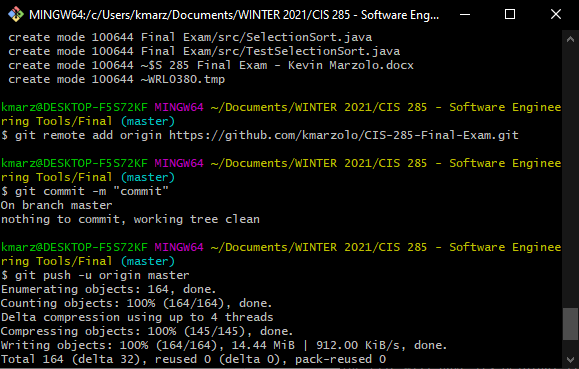
}

}

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