Methodology

Section 1: getInput()

This section handles retrieving and parsing data from the user. Input is taken using a Scanner object and stored into 2 strings. From there the decimal is located in each string and the strings are separated into integer and decimal parts. Finally, the 2 integer parts are made to be the same length, same with the 2 decimal parts, this makes addition and subtraction much easier later on.

Section 2: findLarger()

To find the larger of the strings I first recombine them by their parts, then traverse both strings, when the larger hasn’t been found then each character is compared. After the comparison a larger number is either found or not. In the case that it is found the larger is stored, after the loop is finished and a larger is not found both numbers are the exact same, so one is picked to be the larger. After deciding a larger, the numbers are classified as bigger or smaller, this allows the difference to be calculated without finding the larger again.

Section 3: calculateSum()

The sum of 2 numbers is found by first combining each number from it’s parts. Then, starting at the back of both strings, each character is added together along with the carry and then adjusted if it is greater than 10, causing a carry to be set; this occurs in a loop going through each character. After the adjustment the number is concatenated onto the front a string that holds the sum. After the loop the total sum is split into it’s integer and decimal parts, this is done by knowing that the integer part of the sum will have the same length as the integer part of both numbers. Finally, if a carry is found a ‘1’ is added to the front of the integer.

Section 4: calculateDifference()

To find the difference first the bigger number is created from it’s parts, same with the smaller number. Next, a loop traverses the strings from back to front and subtracts the characters, along with the carry adjusting the number if the difference is negative and setting the carry. Next, the difference is added to the front of a string that holds the total difference. After this loop the total difference is split into it’s integer and decimal parts, this is done by knowing that the length of the integer part will be constant.

Section 5: printResults()

This section simply prints out the results, trivial considering that the results are calculated and have been stored prior to using this method.

Bonus Section: padString(s, length, isFront)

This section will loop until the length of the string, s, is equal to the target length, length. This is done with a simple loop, in the loop a ‘1’ is added to the front or back of the string based on the isFront parameter. Finally, the padded string is returned.

Proofs

Section 1: getInput()

{P}

Scanner keyb = new Scanner(System.in);

System.out.print("String1 = ");

num1 = keyb.nextLine();

System.out.print("String2 = ");

num2 = keyb.nextLine();

keyb.close();

{R1}

short idx1 = (short) num1.indexOf(".");

integer1 = num1.substring(0, idx1);

decimal1 = num1.substring(idx1 + 1, num1.length());

short idx2 = (short) num2.indexOf(".");

integer2 = num2.substring(0, idx2);

decimal2 = num2.substring(idx2 + 1, num2.length());

{R2}

integer1 = padString(integer1, (short) integer2.length(), true);

integer2 = padString(integer2, (short) integer1.length(), true);

decimal1 = padString(decimal1, (short) decimal2.length(), false);

decimal2 = padString(decimal2, (short) decimal1.length(), false);

{Q}

{P} … {Q}: Precondition

1) num1, num2, integer1, decimal1, integer2, decimal2 have been declared but not initialized

Postcondition

1. num1, num2 have the users values
2. integer1, integer2 have the content of respective numbers before the decimal
3. decimal1, decimal2 have the content of respective numbers after the decimal
4. integer1, integer2 have the same length
5. decimal1, decimal2 have the same length

{P} … {R1}: Precondition