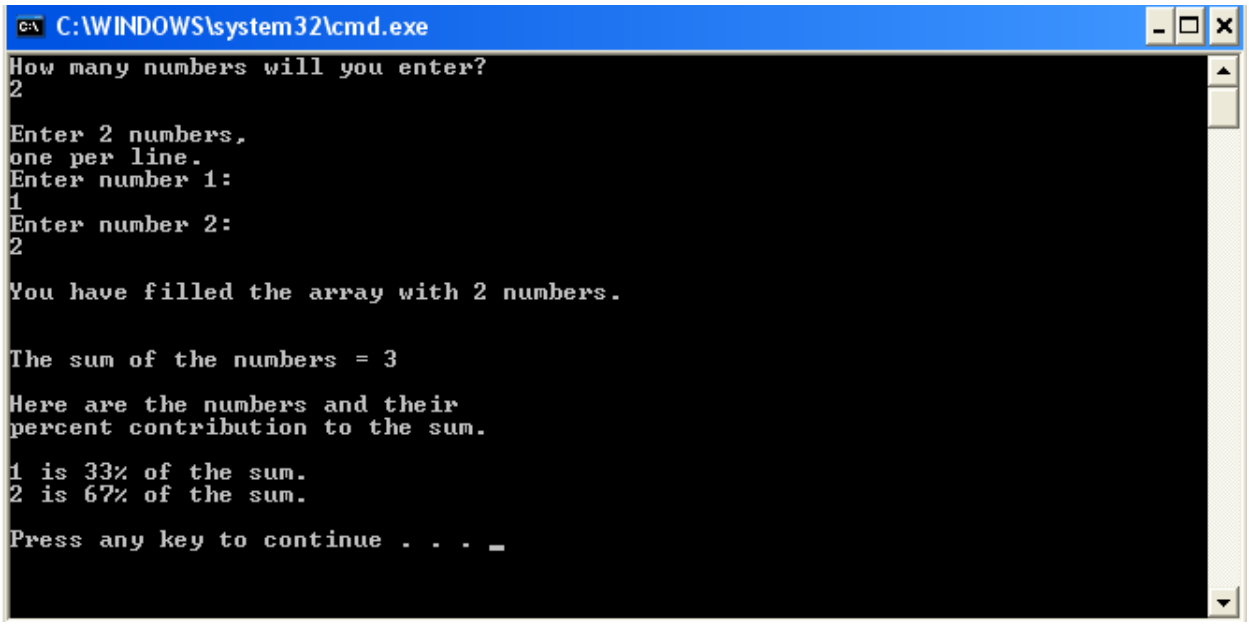
CIT 149: Java I
Chapter 7 Lab 4 Directions

In this lab we will complete programming project # 1 on page 562. This program will read a series of integers, one per line, display their sum, and the percentage each number is of the sum. When the program is run it will look similar to this:



1. Open a new document window and save the program as PercentOfSum.java.
2. Type the appropriate block comments.
3. Type the code that will import the Scanner class.
4. Type the class header, opening brace, main method header, and the opening brace for the main method.
5. Construct a new Scanner object named keyboard.

6. Type the following code that will declare our variables:

```
int thisMany, index, sum, percent;
int[] theList;
```

7. Type the following code that will ask the user how many numbers they wish to enter, and will assign the user input to the variable *thisMany*:

```
System.out.println("How many numbers will you enter?");
thisMany = keyboard.nextInt();
```

8. The array will have its length determine by the user input. Type:

```
theList = new int[thisMany];
```

So the array is limited to the number entered by the user.

9. The user will next be requested to enter the numbers, one per line. Type:

```
System.out.println("\nEnter " + thisMany + " numbers,");
System.out.println("one per line.");
```

10. Using a for loop we will ask the user to enter each number. The array element with the current index number of the variable *index* will receive the value, one by one. Type:

```
for (index = 0; index < thisMany; index++)
{
   System.out.println("Enter number " + (index + 1) + ": ");
   theList[index] = keyboard.nextInt();
}
```

- Each time the for loop is run the user will be asked to enter the number. In this case the first time the user will be asked to "Enter number 1, the value will be assigned to the array with an index

number of 0. The reason for this is because the index numbers in an array always start with zero. When asking them to enter a number we add 1 to the value of index, so we do not start off by asking the user to enter number zero.

- The loop will repeat, increasing the value of *index* by 1, until the index is less than *thisMany*. It must be less because the array starts with an index of zero.

11. We inform the user of how many numbers they have entered. Type:

```
System.out.println();
System.out.print("You have filled the array with ");
System.out.println(thisMany + " numbers.");
System.out.println();
```

12. We add the numbers by giving the variable *sum* the value returned by the addUpValues() method. The method is passed the array, and the value of the variable *thisMany*. Type:

```
sum = addUpValues(theList, thisMany);
```

13. We print out the sum. Type:

```
System.out.println();
System.out.println("The sum of the numbers = " + sum);
System.out.println();
```

14. We let the user know that we are going to print out the percentages of the sum for each number. Type:

```
System.out.println("Here are the numbers and their ");
System.out.println("percent contribution to the sum.")
System.out.println();
```

15. A for loop is used to print out the percentages. Within a System.out.println() we include the formula used to determine the percentage. Type:

```
for(index = 0; index < thisMany; index++)
System.out.println(theList[index] + " is " +
    (int)(((double)theList[index]/sum)*100 + 0.5)+ "% of the sum.");
```

The formula takes the value of the current array element, divides it by sum after casting it to a double, since a decimal will be the result of the division. The value is then multiplied by 100 and 0.5 is added to it. This is a common formula.

16. We will finish by printing out a blank line, and closing the main method. Type:

```
    System.out.println();
}
```

17. Next we create the addUpValues() method. Type:

```
public static int addUpValues(int[] listOfValues, int listLength)
{
    int total = 0;

    for(int i = 0; i < listLength; i++)
        total = total + listOfValues[i];

    return(total);
}
```

- A for loop is used to go through the array adding the value to the current value of the variable *total* each time the loop is run.
- The value of *total* is returned to where the method was invoked.

18. Close the class and compile the program.
19. Compress the following files into a single zip or rar file and submit to the appropriate drop box.

    PercentOfSum.java
    PercentOfSum.class