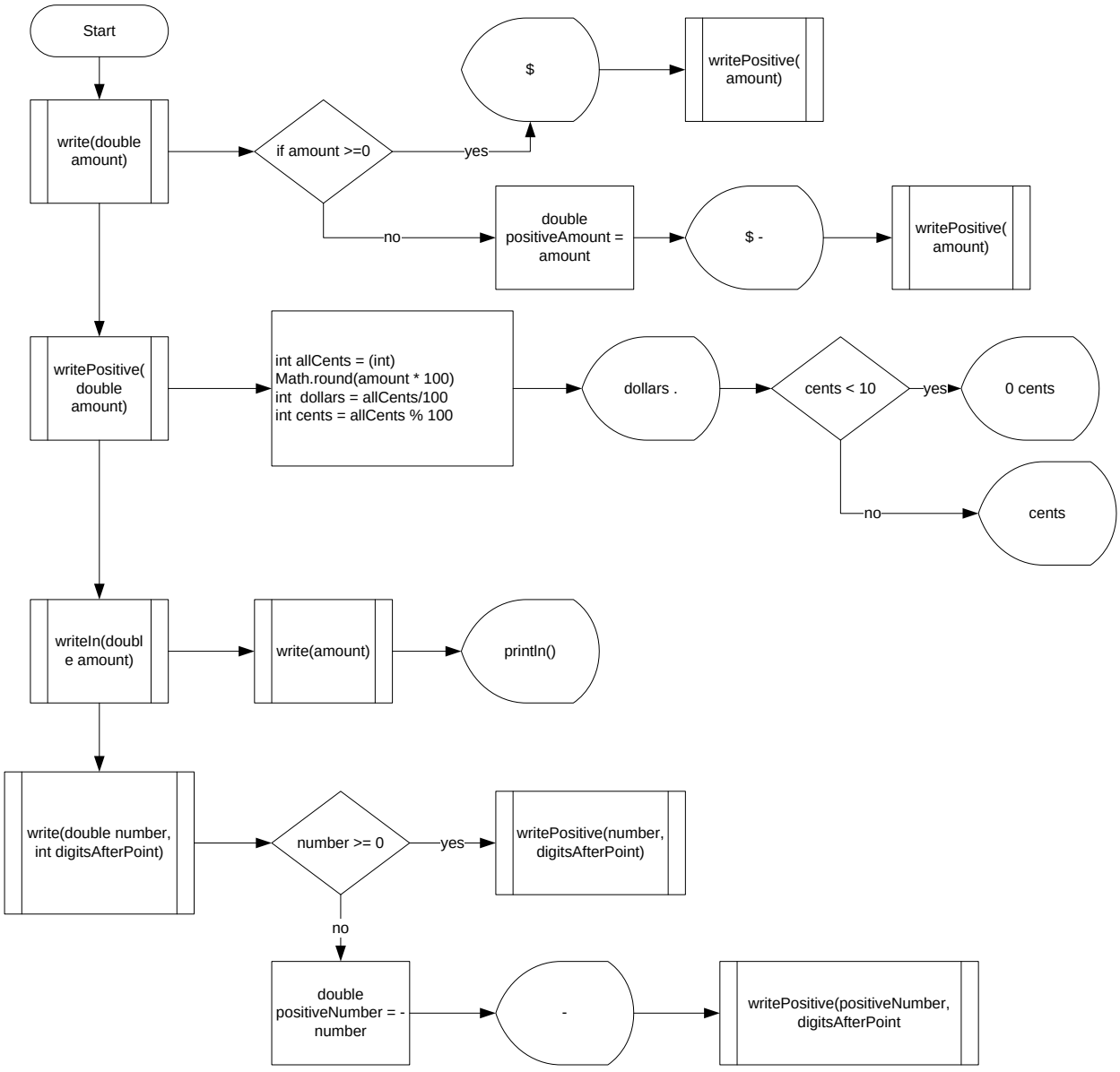
CIT 149: Java I
Chapter 6 Lab 1

In this program we will work on #1 on page 466. I will provide only a portion of the flowchart for the helper class since this program is quite long. With the longer assignments, I will provide only partial flowcharts to show you the process.



DoubleOut class

This class will provide methods to output floating point numbers in three formats. At the beginning of the class there will be block comments to tell the purpose of the program.

Let's get started!

1. Open a new document window in TextPad and save the program as DoubleOut.java.
2. First we will type our block comments. Type:

```
/**
File Name: DoubleOut.java

Outputs a floating point number (of type double)
in three different formats:
  (1) dollars and cents,
  (2) decimal value with the number of digits to the right of
      the decimal point specified, and
  (3) "e" (scientific) notation: one digit to the left of
      the decimal point and the specified number of digits to
      the right, followed by the letter "e" and the exponent of 10.
*/
```

3. Type your class header and opening brace.

4. First is our write method. Type:

```
/**
   Outputs amount in dollars and cents notation. Rounds after two
   decimal points. Does not advance to the next line after output.
*/
public static void write(double amount)
{
   if (amount >= 0)
   {
      System.out.print('$');
      writePositive(amount);
   }
   else
   {
      double positiveAmount = -amount;

       System.out.print('$');

       System.out.print('-');

       writePositive(positiveAmount);

     }

  }
```

- if *amount* is >= 0 display the value with a $ sign in front
- else create a new double and set its value to the value of *amount*
    - display a $ sign and a hyphen and invoke the writePositive() method passing to it the value of the new double

5. Our writePositive() method will output the amount as dollars and cents. Type:

```
    // Precondition: amount >= 0;
  // Outputs amount in dollars and cents notation. Rounds
  // after two decimal points. Omits the dollar sign.

  private static void writePositive(double amount)
  {
     int allCents = (int)(Math.round(amount * 100));
     int dollars = allCents / 100;
     int cents = allCents % 100;

     System.out.print(dollars);
     System.out.print('.');
     if (cents < 10)
     {
        System.out.print('0');
        System.out.print(cents);
     }
     else
        System.out.print(cents);
  }
```

- Notice that it checks to see if there are any cents. Modular division is used to find the cents

6. Our writeln() method is fairly simple. This is a lowercase L not an uppercase I. Type:

```
/**
  Outputs amount in dollars and cents notation. Rounds after
  two decimal points. Advances to the next line after output.
 */
 public static void writeln(double amount)
 {
    write(amount);
    System.out.println();
 }
```

7. Next we overload the write() method. When overloading methods, the method has the same name but different parameters. Type:

```
/**
   Writes out number with digitsAfterPoint digits after
   the decimal point. Round any extra digits.
   Does not advance to the next line after output.
*/
private static void write(double number, int digitsAfterPoint)
{
   if (number >= 0)
      writePositive(number, digitsAfterPoint);
   else
   {
      double positiveNumber = -number;
      System.out.print('-');
      writePositive(positiveNumber, digitsAfterPoint);
   }
}
```

8. We also overload the writePositive() method. Type:

```
// Precondition: number >= 0
 // Writes out number with digitsAfterPoint digits after the
 // decimal point. Rounds any extra digits.

 private static void writePositive(double number, int digitsAfterPoint)
 {
    int mover = (int)(Math.pow(10, digitsAfterPoint));
                    // 1 followed by digitsAfterPoint zeros

    int allWhole;   // Number with the decimal point
                    // moved digitsAfterPoint places

    allWhole = (int)(Math.round(number * mover));
    int beforePoint = allWhole / mover;
    int afterPoint = allWhole % mover;

    System.out.print(beforePoint);
    System.out.print('.');
    writeFraction(afterPoint, digitsAfterPoint);
 }
```

- Notice that at the end we invoke the writeFraction() method.

9. The writeFraction() method outputs a value at a certain power using the Math.pow() method. Type:

```
// Outputs the integer afterPoint with enough 0s
 // in front to make it digitsAfterPoint digits long.

 private static void writeFraction(int afterPoint, int digitsAfterPoint)
 {
    int n = 1;
    while (n < digitsAfterPoint)
    {
       if (afterPoint < Math.pow(10, n))
          System.out.print('0');

       ++n;
    }

    System.out.print(afterPoint);
 }
```

- if the variable *afterPoint* is less than the number 10, power of the variable *n* then we display 0, else we add 1 to *n*.
- This is done within a while loop which is run if the variable *digitsAfterPoint* is greater than *n*.

10. We overload the writeln() method. Type:

```
/**
  Writes out number with digitsAfterPoint digits after
  the decimal point. Rounds any extra digits.
  Advances to the next line after output.
 */
 public static void writeln(double number, int digitsAfterPoint)
 {
    write(number, digitsAfterPoint);
    System.out.println();
 }
```

11. The scienceWrite() method handles displaying the number in scientific notation. Type:

```
// E (SCIENTIFIC) NOTATION
 // Writes out in "e" (scientific) notation


 /**
  Writes out number in scientific ("e") notation, i.e. with
  one (non-zero) digit to the left of the decimal point,
  digitsAfterPoint digits after the decimal point, and the
  exponent of 10 required to correctly place the decimal point.
  Rounds any extra digits.
  Does not advance to the next line after output.
 */
 private static void scienceWrite(double number, int digitsAfterPoint)
 {
    if (number >= 0)
       scienceWritePositive(number, digitsAfterPoint);
    else
    {
       double positiveNumber = -number;
       System.out.print('-');
       scienceWritePositive(positiveNumber, digitsAfterPoint);
    }
 }
```

- The scienceWritePositive() method will handle the process.

12. The scienceWritePositive() method is written:

```
// Precondition: number >= 0
 // Writes out number with one non-zero digit to left of decimal
 // point and digitsAfterPoint digits after the decimal point
 // and the exponent of 10. Rounds any extra digits.

 private static void scienceWritePositive(
                    double number, int digitsAfterPoint)
 {
    int wholePart = (int) number; // Get just the whole number part
    int e = 0;                    // Initialize exponent

    if(wholePart > 0)
    {
       // Positive exponent = number of divisions by 10

       while(wholePart >= 10)
       {
          ++e;
          wholePart = wholePart / 10;
       }
    }
    else
```

```
{
   // Negative exponent = number of multiplications by 10

   double nextValue = number;
   while((int)nextValue < 1)
   {
      --e;
      nextValue = nextValue * 10;
   }
}

int mover = (int)(Math.pow(10, digitsAfterPoint));
                  // 1 followed by digitsAfterPoint zeros

int secondMover;
                  // Additional decimal point adjuster
                  // to obtain digitsAfterPoint digits

int allWhole;       // Number with the decimal point
                  // moved digitsAfterPoint places

if(e < 0)
{
   secondMover = (int)(Math.pow(10, -e));
                  // pow does not accept negative exponents
```

```
        allWhole = (int)(Math.round(number * mover * secondMover));
    }
    else // e greater than or equal to zero
    {
        secondMover = (int)(Math.pow(10, e));
        allWhole = (int)(Math.round(number * mover / secondMover));
    }

    int beforePoint = allWhole / mover;
    int afterPoint = allWhole % mover;

    System.out.print(beforePoint);
    System.out.print('.');
    writeFraction(afterPoint, digitsAfterPoint);
    System.out.print("e" + e);
}
```

- this can appear complicated but if you go through the code line by line it is not as difficult as it seems.

13. Our final method is the scienceWriteln(). Type:

```
/**
  Writes out number in scientific ("e") notation, i.e.
  with one digit to the left of the decimal point,
  digitsAfterPoint digits after the decimal point, and the
  exponent of 10 required to correctly place the decimal point.
  Rounds any extra digits.
  Advances to the next line after output.
 */
 public static void scienceWriteln(double number, int digitsAfterPoint)
 {
    scienceWrite(number, digitsAfterPoint);
    System.out.println();
 }
```

14. Close the class. Compile the program and fix any errors as necessary.

DoubleOutDriver class

This class will test the DoubleOut class.

1. Open a new document window and save the program as DoubleOutDriver.java.

2. First we will add block comments that display the purpose of the class:

```
/**

  File name: DoubleOutDriver.java

  Until the user says not to repeat:
    Read in a floating point number (type double).
    Call all three writeln methods to display the number
          (using 5 for the number of digits to the right of the
          decimal point when you need to specify such a number).
*/
```

3. Type the import statement that will import all classes within the java.util package.
4. Type the class header and opening brace, and main method header and opening brace.
5. Within the main method construct a new Scanner object named keyboard and simply declare a double named value and a char named ans
6. Write the code that will display the words "Testing all 3 DoubleOut writeln methods"

7. Within a do/while loop we will request information from the user, display the amount in 3 formats and then ask if the user want to test the DoubleOut class again. Type:

```
do
{
   System.out.println("Enter a value of type double:");
   value = keyboard.nextDouble();
   System.out.println();

   // Use writeln method to output in dollars format
   // followed by a new line.
   DoubleOut.writeln(value);
   System.out.println();

   // Use the writeln method to output with a specified
   // number of decimal places (5 for this test)
   // followed by a new line.
   DoubleOut.writeln(value, 5);
   System.out.println();

   // Use the scienceWriteln method to output in scientific
   // notation followed by a new line.
   DoubleOut.scienceWriteln(value, 5);
   System.out.println();

   System.out.println("Test again?(y/n)");
```

```
        ans = keyboard.next().charAt(0);

        System.out.println();


    }while ((ans == 'y') || (ans == 'Y'));
```

8. Type the code that will display "End of test"
9. Close the main method and the class.
10. Compile the program, fix any errors.
11. Run and test the program.
12. Compress ALL files into a single zip or rar file and submit to the drop box.