Project 3 – Convert to and from single precision binary floating-point number and a Real number representations.

Create a class named FloatingPointNumber that contains at least two methods.

* convertRealToFloatingPointNumber(String):String
  + This method accepts a Real number as a String and converts it into a String representation of a single precision binary floating-point number.
* convertFloatingPointNumberToReal(String):String
  + This method accepts a String representation of a single precision binary floating-point number and returns a String representation of a Real number.

Create a test driver class named MyTester that allows the user to enter either a Real number to convert to a single precision floating-point number or enter a 32 “bit” String representation of a single precision floating-point number to convert to a Real number. You must use validation loops to ensure that the user enters valid input strings. A valid representation of a single precision floating-point number is any set of 32 “1s” and “0s”. A valid String for a Real number input from the user is a String that only contains leading +, -, or digit and then only digits after that. We discussed this during lecture.

You will need several algorithms. First you will need two algorithms to convert from the String Real number value to a Java double value. Then you will need two other algorithms to convert the integer part of the Real number to binary and the fractional part to binary fractional value. See the document on Single Precision Floating Point Number conversions.

* Converting the String Real value to a Java double value depends on finding the “radix point”. The radix point is the period in the Real Number String. The Real Number String may only contain 1 period.
  + the integer portion of the Real is to the left of the period, and the fractional portion to the right of the period.
  + the String “1234.567” may be converted to a Java double by multiplying each digit in the String by its positional power value. If there is no period, we can assume that the last digit in the String is at the 10 to the 0-power position.   
    (ie In 123 the 3 has the value of 3 \* 10^0 ect.)
* The binary integer part of the Real number can be found by repeatedly dividing the base 10 integer value 2 by until the quotient is zero. You use the same technique that you used for the base conversion program.
* The binary fractional values can be determined by multiplying the fractional base 10 value by 2 and concatenating the integer portion of the result to the new binary String. You may need to do 150 divides before doing your calculation of the exponent.
* You will need an algorithm to calculate the exponent value.
* You may not use any of the Math class methods nor any Java class nor method to do your conversions for you. You will need to write your own methods for the power and number conversion methods.