with option to output in text file.

* IEEE-754 Binary-16 floating point converter (including all special cases)

- Input: (1) binary mantissa and base-2 (i.e., 101.01×2^5) (2) Decimal and base-10 (i.e. 65.0×10^3) Also should support special cases (i.e., NaN).
- Output: (1) binary output with space between section (2) its hexadecimal equivalent (3) with option to output in text file.

For the given specs we used Java as our programming language, under the public class HalfPrecisionConverter, we have our classes convertInput, base10toDecimal, decimaltoBinary, binaryToHex, normalize, and getExpRep.

For convertInput, we prepare our variables, the XIndex,decimal and signBit in order to concatenate after reading the input, the class inside gets the signbit of the input, checks if the input is base-10 that converts into a binary value , normalize to 1.f, gets the exponent and fraction.

For base10Decimal class, its function is to convert base-10 input to decimal value as double, with the formula `double decimal = base * Math.pow(10, exponent).`

For the decimalToBinary, its function is to convert decimal value to binary value as string. If the fractional part is equal to 0 it returns to the binary integer part, if the fractional part is more than 0, the length should not be more than 32 bits, else, it appends to 1 or 0 then it returns to a concated binary string.

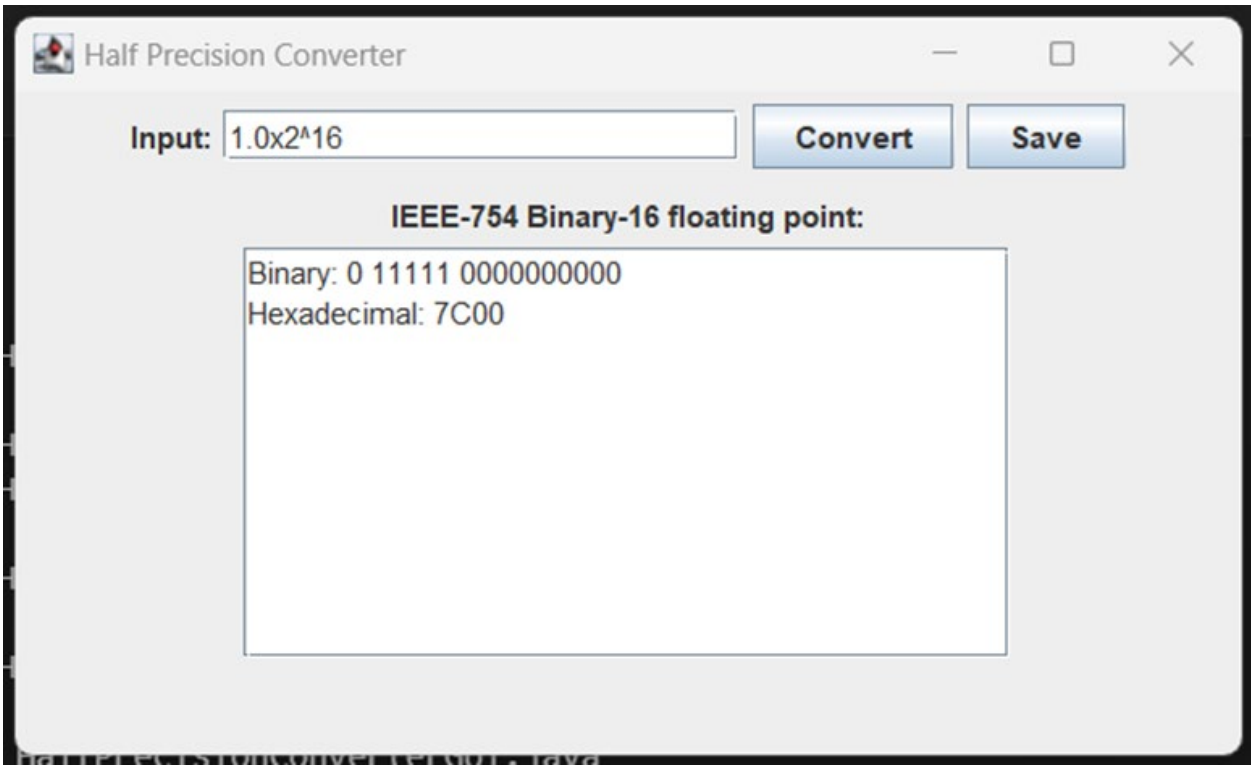
For the class binaryToHex it converts the binary value from the class decimalToBinary to hexadecimal. It checks and cuts the spaces of the given binary, then an operation modulo 4 is used until it reaches 0. Then it initializes the stringbuilder to store the hexadecimal.

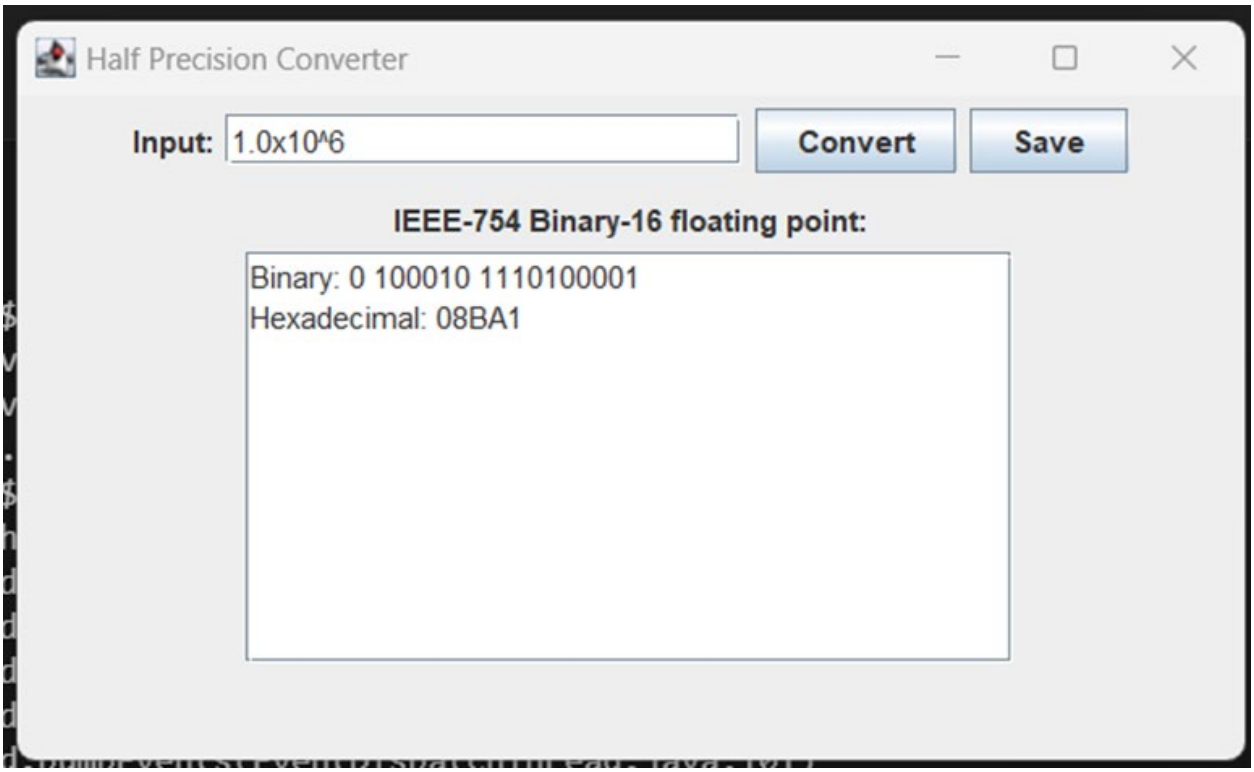
For the normalize class, it normalizes binary to 1.f

The normalize method adjusts a binary string to represent a normalized floating-point number with the format 1.f * 2^e. It extracts the exponent part from the input string and adjusts it based on the position of the radix point. Then, it constructs the normalized representation by appending the binary fraction part to '1.' and the adjusted exponent, returning it as a string.

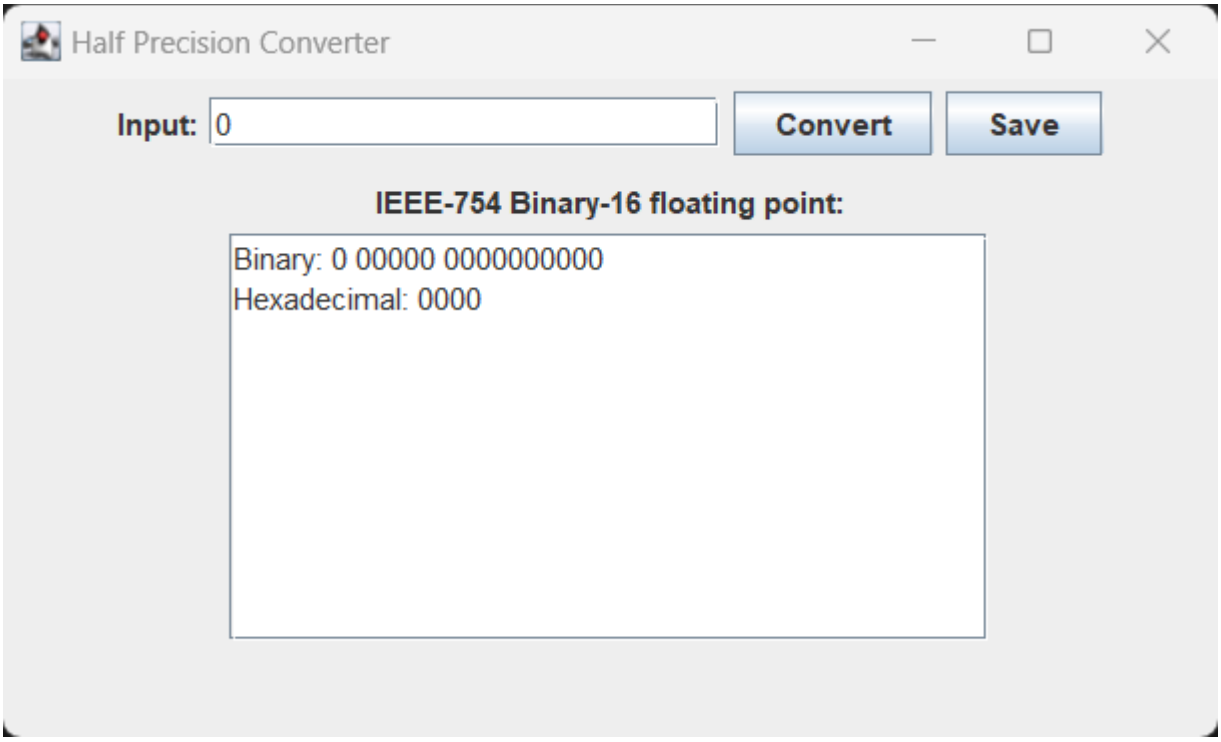
For the getExpRep, it gets the binary exponent representation as string.

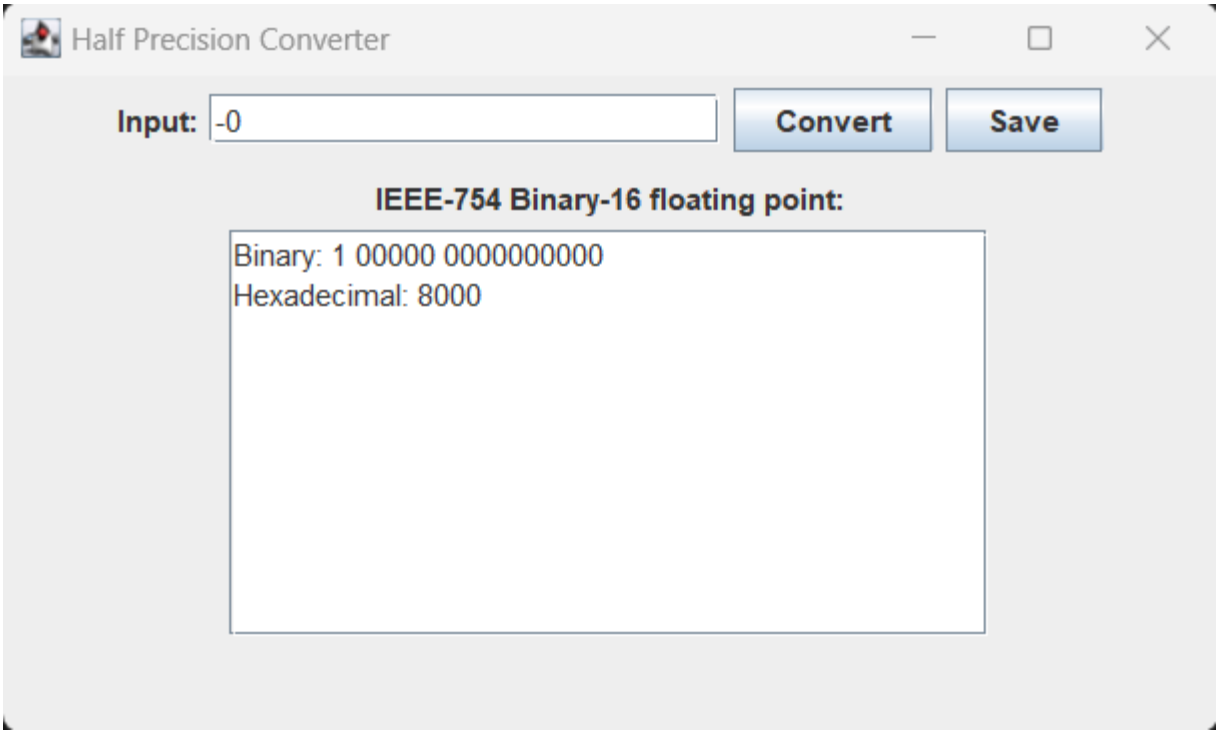
Code Snippets:


Half Precision Converter
Input: 1.0x2^16
Convert
Save
IEEE-754 Binary-16 floating point:
Binary: 0 11111 0000000000
Hexadecimal: 7C00

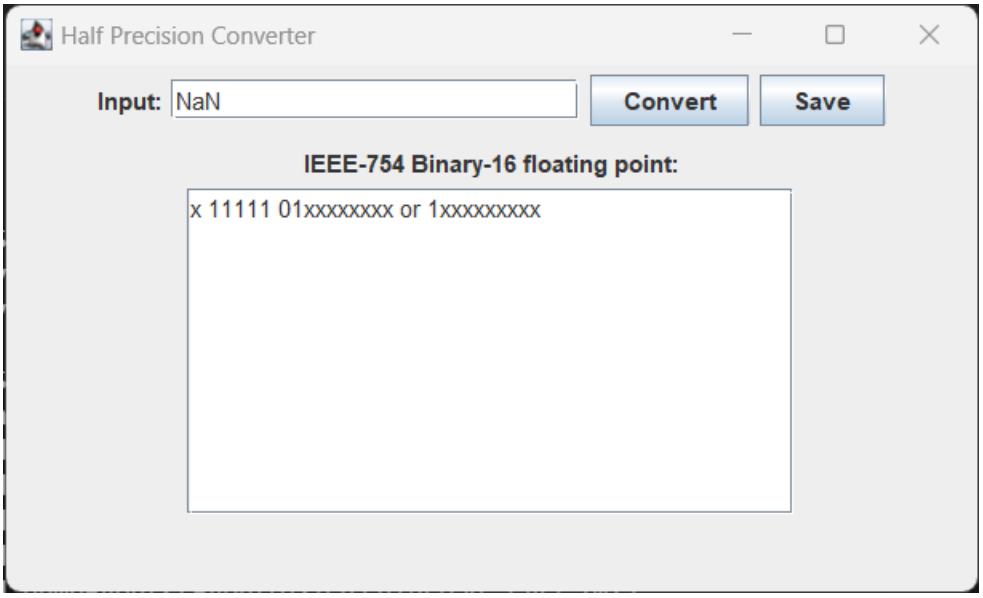

Half Precision Converter
Input: 1.0x10^6
Convert
Save
IEEE-754 Binary-16 floating point:
Binary: 0 100010 1110100001
Hexadecimal: 08BA1

Zero

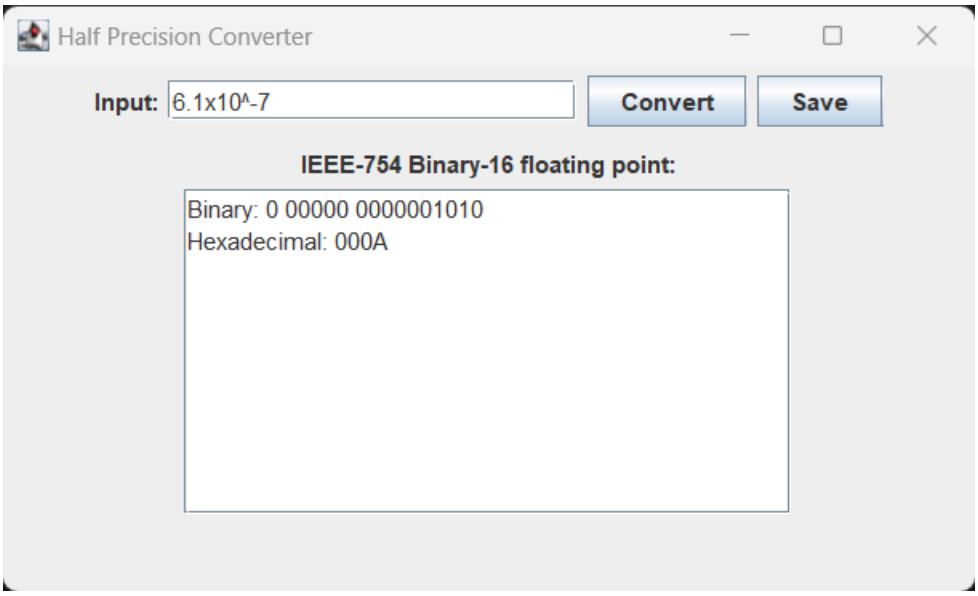

Half Precision Converter
Input: 0
Convert
Save
IEEE-754 Binary-16 floating point:
Binary: 0 00000 0000000000
Hexadecimal: 0000

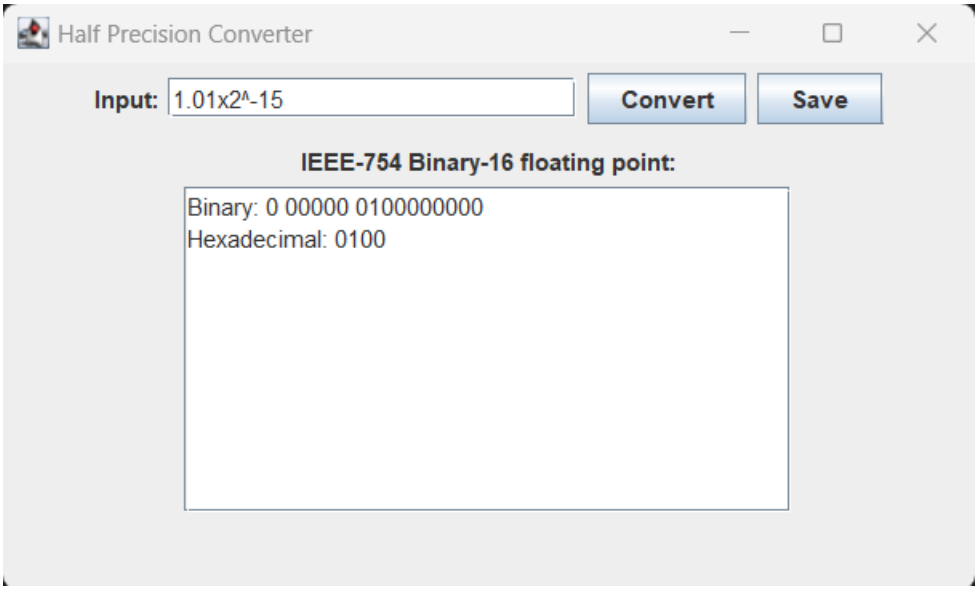

Half Precision Converter
Input: -0
Convert
Save
IEEE-754 Binary-16 floating point:
Binary: 1 00000 0000000000
Hexadecimal: 8000

NaN



Denormalized




Half Precision Converter
Input:
1.01x2^-15
Convert
Save
IEEE-754 Binary-16 floating point:
Binary: 0 00000 0100000000
Hexadecimal: 0100