Report

Jingjing Nie (UID: 304567417)

**a.**

The notable obstacles I encountered in this project were to figure out how to use class in correct way with correct syntax and format. Also, how to build nested classes (to use class within another class) is pretty challenging for me since the type and constructor (it gave the warning of no default constructor) are quite confusing.

**b.**

Test cases that can be used:

Date (10, 25) should have the sign Scorpio, and it is not on the cusp, so for the second sign, it should have Aries by default.

Date (1, 1) should have the sign Capricorn, and it is not on the cusp, so for the second sign, it should have Aries by default.

Date (11, 21) should have the sign Scorpio, and it is on the cusp, so it should have the second sign of Sagittarius.

Date (8, 23) should have the sign Virgo, and it is on the cusp, so it should have the second sign of Leo.

Also, similar test cases can be written in assert forms:

Date jan1( 1, 1 );

ZodiacReader reader( jan1 );

assert( reader.stringifySign( reader.checkSign() ) == "Capricorn" );

assert( reader.checkSign( ) == ZodiacReader::CAPRICORN );

assert( !reader.onCusp( ) );

Date feb1( 2, 1 );

ZodiacReader reader1( feb1 );

assert( reader1.stringifySign( reader1.checkSign() ) == "Aquarius" );

assert( reader1.checkSign( ) == ZodiacReader::AQUARIUS );

assert( !reader1.onCusp( ) );

Date jan20( 1, 20 );

ZodiacReader reader2( jan20 );

assert( reader2.stringifySign( reader2.checkSign() ) == "Aquarius" );

assert( reader2.checkSign( ) == ZodiacReader::AQUARIUS );

assert( reader2.onCusp( ) );

assert( reader2.stringifySign( reader2.cuspSign() ) == "Capricorn" );

assert( reader2.cuspSign( ) == ZodiacReader::CAPRICORN );

Date march21( 3, 21 );

ZodiacReader reader3( march21 );

assert( reader3.stringifySign( reader3.checkSign() ) == "Aries" );

assert( reader3.checkSign( ) == ZodiacReader::ARIES );

assert( reader3.onCusp( ) );

assert( reader3.stringifySign( reader3.cuspSign() ) == "Pisces" );

assert( reader3.cuspSign( ) == ZodiacReader::PISCES );

Date april( 4, 21 );

ZodiacReader reader4( april );

assert( reader4.stringifySign( reader4.checkSign() ) == "Taurus" );

assert( reader4.checkSign( ) == ZodiacReader::TAURUS );

assert( reader4.onCusp( ) );

assert( reader4.stringifySign( reader4.cuspSign() ) == "Aries" );

assert( reader4.cuspSign( ) == ZodiacReader::ARIES );

Date may20 ( 5, 20 );

ZodiacReader reader5( may20 );

assert( reader5.stringifySign( reader5.checkSign() ) == "Taurus" );

assert( reader5.checkSign( ) == ZodiacReader::TAURUS );

assert( reader5.onCusp( ) );

assert( reader5.stringifySign( reader5.cuspSign() ) == "Gemini" );

assert( reader5.cuspSign( ) == ZodiacReader::GEMINI );

Date june( 6, 22 );

ZodiacReader reader6( june );

assert( reader6.stringifySign( reader6.checkSign() ) == "Cancer" );

assert( reader6.checkSign( ) == ZodiacReader::CANCER );

assert( reader6.onCusp( ) );

assert( reader6.stringifySign( reader6.cuspSign() ) == "Gemini" );

assert( reader6.cuspSign( ) == ZodiacReader::GEMINI );

Date july( 7, 28 );

ZodiacReader reader7( july );

assert( reader7.stringifySign( reader7.checkSign() ) == "Leo" );

assert( reader7.checkSign( ) == ZodiacReader::LEO );

assert( !reader7.onCusp( ) );

Date august( 8, 15 );

ZodiacReader reader8( august );

assert( reader8.stringifySign( reader8.checkSign() ) == "Leo" );

assert( reader8.checkSign( ) == ZodiacReader::LEO );

assert( !reader8.onCusp( ) );

Date sept10( 9, 10 );

ZodiacReader reader9( sept10 );

assert( reader9.stringifySign( reader9.checkSign() ) == "Virgo" );

assert( reader9.checkSign( ) == ZodiacReader::VIRGO );

assert( !reader9.onCusp( ) );

Date sept( 9, 21 );

ZodiacReader reader10( sept );

assert( reader10.stringifySign( reader10.checkSign() ) == "Virgo" );

assert( reader10.checkSign( ) == ZodiacReader::VIRGO );

assert( reader10.onCusp( ) );

assert( reader10.stringifySign( reader10.cuspSign() ) == "Libra" );

assert( reader10.cuspSign( ) == ZodiacReader::LIBRA );

Date october1( 10, 19 );

ZodiacReader reader11( october1 );

assert( reader11.stringifySign( reader11.checkSign() ) == "Libra" );

assert( reader11.checkSign( ) == ZodiacReader::LIBRA );

assert( !reader11.onCusp( ) );

assert( reader11.stringifySign( reader11.cuspSign() ) == "Aries" );

assert( reader11.cuspSign( ) == ZodiacReader::ARIES );

Date october25( 10, 25 );

ZodiacReader reader12( october25 );

assert( reader12.stringifySign( reader12.checkSign() ) == "Scorpio" );

assert( reader12.checkSign( ) == ZodiacReader::SCORPIO );

assert( !reader12.onCusp( ) );

assert( reader12.stringifySign( reader12.cuspSign() ) == "Aries" );

assert( reader12.cuspSign( ) == ZodiacReader::ARIES );

Date nove( 11, 22 );

ZodiacReader reader13( nove );

assert( reader13.stringifySign( reader13.checkSign() ) == "Sagittarius" );

assert( reader13.checkSign( ) == ZodiacReader::SAGITTARIUS );

assert( reader13.onCusp( ) );

assert( reader13.stringifySign( reader13.cuspSign() ) == "Scorpio" );

assert( reader13.cuspSign( ) == ZodiacReader::SCORPIO );

Date dec25( 12, 5 );

ZodiacReader reader14( dec25 );

assert( reader14.stringifySign( reader14.checkSign() ) == "Sagittarius" );

assert( reader14.checkSign( ) == ZodiacReader::SAGITTARIUS );

assert( !reader14.onCusp( ) );

assert( reader14.stringifySign( reader14.cuspSign() ) == "Aries" );

assert( reader14.cuspSign( ) == ZodiacReader::ARIES );

Date dec20( 12, 20 );

ZodiacReader reader15( dec20 );

assert( reader15.stringifySign( reader15.checkSign() ) == "Sagittarius" );

assert( reader15.checkSign( ) == ZodiacReader::SAGITTARIUS );

assert( reader15.onCusp( ) );

assert( reader15.stringifySign( reader15.cuspSign() ) == "Capricorn" );

assert( reader15.cuspSign( ) == ZodiacReader::CAPRICORN );