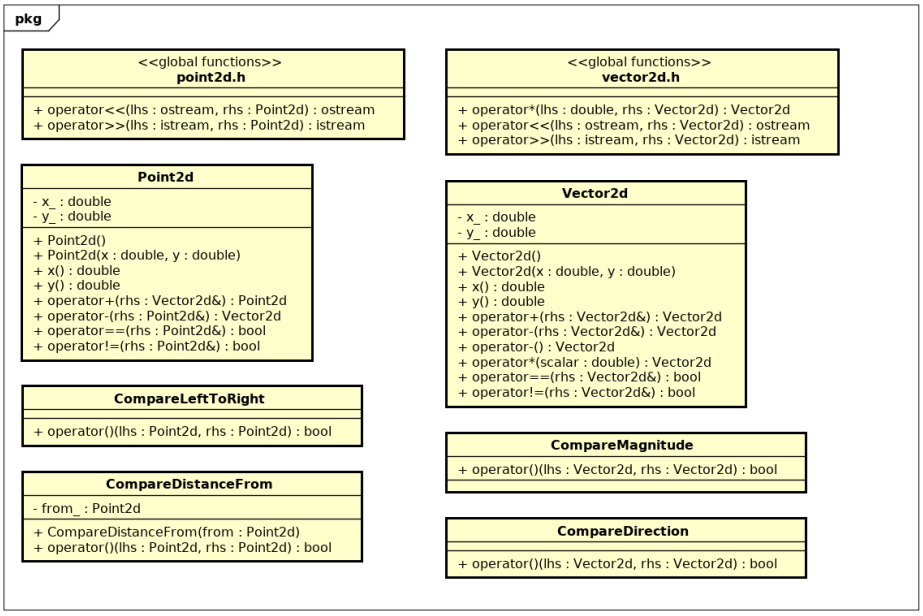
You shall upload a zipped (AND ONLY ZIPPED) archive to Blackboard containing the directory hw5 and:

1. A `hw5/vector2d.h` file with your includes and class declarations,
2. A `hw5/vector2d.cc` file with your class definitions,
3. A `hw5/point2d.h` file with your includes and class declarations, and
4. A `hw5/point2d.cc` file with your class definitions, at least.

We are creating a more C++-like library. We are implementing a pair of classes—Point2d and Vector2d. These classes implement the behavior of two-dimensional points and vectors in a Cartesian coordinate system.

They are described in the UML below as well the attached header files.



In an effort to force you to read others' code, I have provided you with a fairly detailed set of unit test functions demonstrating the functionality. IJ and I will be referring any questions answered by reading the provided Unit Tests to the provided Unit Tests. Questions about the tests themselves or beyond the tests will, of course, be answered.

You may compile and link to them to ensure that your classes meet all requirements. The behavior demonstrated in the tests should well-define the expectations of the classes' operations.

The libraries are further described as follows:

1. vector2d.h
   (a) Vector2d—a two-tuple representing direction and magnitude in the Cartesian coordinate system.
      i. operator+ and operator- return the result of the operation where the calling object is the left-hand side of the operation
      ii. operator- returns a vector with the opposite direction of calling object
      iii. operator== and operator!= return true or false when approximately equal or not
   (b) global functions
      i. `operator*` return the result of scaling the rhs by the lhs
      ii. `operator<<` extracts a vector of the form (1.0, 2.0) into the ostream, where 1.0 and 2.0 are the member variables.
      iii. `operator>>` inserts into the rhs, from the istream in the form "1.0 2.0"
   (c) CompareMagnitude—overloads the `operator()` operator such that it returns true if lhs has less magnitude than rhs, false otherwise
   (d) CompareDirection—overloads the `operator()` operator such that it returns true if lhs is closer to 0 than rhs when converted to angles. You may use `atan2` from cmath, but keep in mind that it provides a range from $[-\pi, \pi]$ and the range we want to simulate is $[0.0, 2\pi)$
2. point2d.h
   (a) Point2d—a two-tuple representing location in the Cartesian coordinate system.
      i. operator- returns the magnitude and direction from the calling object to the Point2d parameter as a Vector2d
      ii. operator+ returns a Point2d offset from the calling object by the Vector2d parameter
      iii. operator== and operator!= return true if approximately equal or not
   (b) global functions
      i. operator<< extracts the values from the calling object as (x, y) and adds to the ostream parameter, then returns the ostream parameter
      ii. operator>> reads two floating point values from the istream and adds them to the two attributes of the calling object, then returns the istream parameter.
   (c) CompareLeftToRight—overloads the `operator()` operator and returns true when lhs has a lesser x coordinate or if equal in x, a lesser y coordinate.
   (d) CompareDistanceFrom—overloads the `operator()` operator and returns true when lhs is closer to the point stored in the constructor than rhs.

As in the previous assignments, you will receive up to 100% credit for turning it on the day it is due and will lose 25% per day late.

There are 7 points possible in the tests and another 3 possible in the styling.

The End.