Point Class

This class is the simplest one out of the bunch. Its usage is to store 2 doubles as the X and Y coordinates from the points found using the LiDAR. Since the coordinates can't change after assignment, the variables that store those values are held private and can only be accessed using the methods *getX()* and *getY()*. To declare a Point object we have two constructor options, being the default one *Point()* that assigns X and Y coordinates to -10^{20}, and the other being *Point(x, y)*. The default value was chosen in order to allow operations to be made without overflow or errors because it is pretty far from the double range, but it is so large that it is impossible to get during normal execution.

Finally, this class implements equal, different and print operators, the method *isAssigned()* that returns true if X and Y coordinates are different than the default one, and *distanceToOrigin()* that calculates the point's distance to the origin. This is useful because the robot is always the origin, so this method calculates the point's distance to the robot's center.

Public Methods

- Point( ) [1/2]

  Point::Point ( )

  Default constructor that assigns default value to point

- Point( ) [2/2]

  ```
  Point::Point ( const double x,
                 const double y
               )
  ```

  Constructor to assign value

  Parameters

  x is the point's x coordinate
  y is the point's y coordinate

- getX( )

  double Point::getX ( ) const

  Get point's x-coordinate

- getY( )

  double Point::getY ( ) const

  Get point's y-coordinate

- isAssigned( )

  bool Point::isAssigned ( ) const

  Checks if point's coordinate are set to something other than the default value

- distanceToOrigin( )

  double Point::distanceToOrigin ( ) const

  Calculates euclidean distance from point to (0,0)

- operator == ( )

  bool Point::operator == (const Point & p) const

  Checks if two points both coordinates are equal

  Parameters

  p is the second point to be compared

- operator != ( )

  bool Point::operator == (const Point & p) const

  Checks if any of the two points coordinates are different

  Parameters

  p is the second point to be compared

- friend operator << ( )

  std::ostream & operator << (std::ostream & out, const Point & p)

  Print point object in terminal in the form Point: [x: 'p.x', y: 'p.y']

  Parameters

  out is where to print, normally terminal

  p is the object to be printed