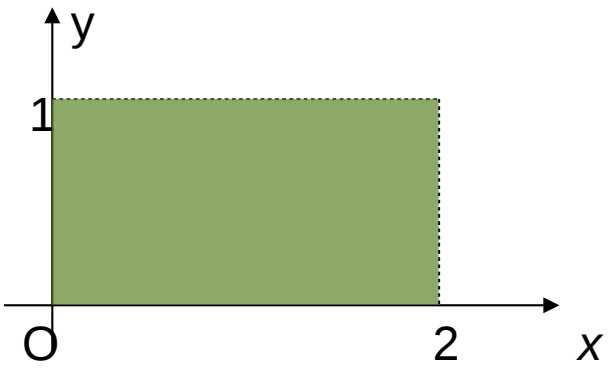
The Classes Line and LineTools

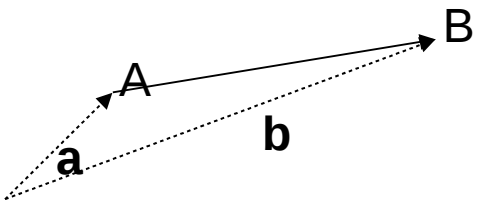
Introduction: *The World*

The World consists of the pitch, the two robots and the ball. The pitch is a rectangle with length 2 and width 1 as shown bellow:



The positive direction for the angles is **counter clockwise.**

Class Line:



A Line object is defined by two points (A and B) as shown above. The order of the points matters because it determines the direction of the line, i.e. A -> B. Thus, the objects of class Line can be treated as rays. In this case, each ray represents a movement of an object in the World, i.e. the object has moved from point A to point B, and the vector $B\vec{-}A$ defines the direction in which the object will continue to move.

The objects of class Line can also be treated as infinite lines. For instance, the function intersectionOfLines from LineTools does exactly this by ignoring the fields firstPoint and secondPoint of Line and using only the fields gradient and offset.

Class LineTools:

The class LineTools contains basic function for manipulating lines. There are eight functions in the class. Listed below are the formulas used in two of the function of the class. More about the functions can be found in the comments of LineTools.java.

public Coordinates symmetricalPoint(Coordinates point, Line line);

Let (x_1, y_1) be the point given, (x_2, y_2) - the point that is is symmetrical to (x_1, y_1) with respect to the line, and $y = ax + b$ - the equation of the line. Then the following equations hold:

$y_2 = -(1/a)x_2 + (y_1 + (1/a)x_1)$ (1)

$(y_1 + y_2)/2 = a(x_1 + x_2)/2 + b$ (2)

(1) holds because the point (x_2, y_2) is on the line with gradient $-(1/a)$

passing through the point (x_1, y_1).

(2) holds because the midpoint $((x_1+x_2)/2, (y_1+y_2)/2)$ of the segment formed by the points (x_1, y_1) and (x_2, y_2) is also on the given line.

Then we solve the two equations for x_2 and get:

$x_2=[2y_2+((1/a)-a)-2b]/[(1/a)+a]$

Respectively, for y_2 we get:

$y_2=-(1/a)x_2+[y_1+(1/a)x_1]$

public double distanceFromPointToLine(Coordinates point, Line line);

This function uses the formula:

$d=|\vec{n}\cdot\vec{p}|/|\vec{n}|$

Where d is the distance from the point to the line. $\vec{n}$ is a normal vector to the line, and $\vec{p}$ is the vector starting from the point and ending at one of the points on the line.

In the function we choose $\vec{n}=(line.gradient, -1)$ and $\vec{p}=\vec{point}-(0, line.offset)$.