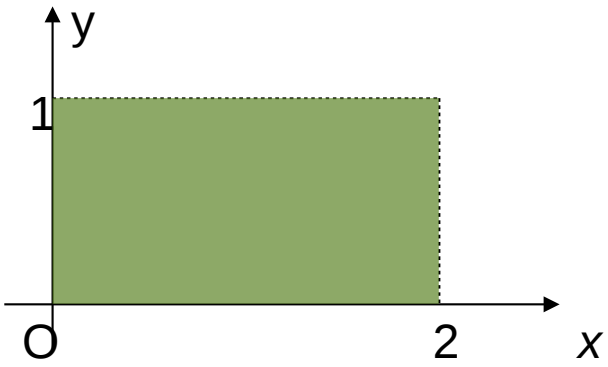
The Classes Line and LineTools

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.**

Note: Sometimes when we use the word "line" below it may not mean an actual line but an object of class Line.

Class Line:

```
public class Line {
    private double gradient;
    private double offset;
    private boolean direction;
    private double rangeMin;
    private double rangeMax;

    private Coordinates firstPoint;
    private Coordinates secondPoint;

    public Line(Coordinates A, Coordinates B){}
    public Line(Coordinates A, Coordinates B, double rangeMin, double rangeMax){}
    ...
    standard getters
    ...
    public boolean isOnLineAndInRange(Coordinates point){}
}
```

A Line object can be defined by two points (A and B) as shown above or two points and a range [min, max]. The order of the points A and B matters because it determines the direction of the line, i.e. A -> B.

The gradient and the offset of the line are such that y=gradient*x+offset is the equation of the line. The gradient cannot be -Infinity. If the line is vertical and has a downwards direction then the direction will be equal to false.

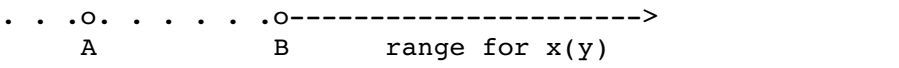
direction is true if the direction of the line is in the direction in which x increases. If x is constant, i.e. the line is vertical, then direction is true if the direction of the line is in the direction in which y increases.

The range [rangeMin, rangeMax] is with respect to x if the line is not vertical. If the line is vertical, then the range is with respect to y.

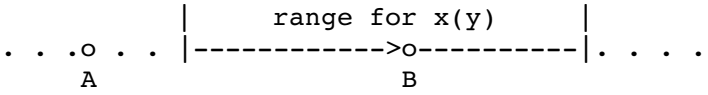
firstPoint and secondPoint are the points A and B.

When we define a Line object by two points, the object represents the movement of a ball, i.e. the ball has moved from point A to point B, and the vector

$\vec{BA}$ defines the direction in which it will continue to move. Then the the range for x is in direction AB starting at point B.



If the Line object is defined by two points and a range then the object represents a segment. This is used when we want a Line object to represent a wall in the World.



isOnLineAndInRange(Coordinates point) checks if the given point is on the line and in the range of the line.

Class LineTools:
The class LineTools contains basic function for manipulating lines.

```
public class LineTools {
    public static Coordinates intersectionOfLines(Line l, Line m){}
    public static Coordinates symmetricalPoint(Coordinates point, Line line){}
    public static double distanceFromPointToLine(Coordinates point, Line line){}
    public static double angleBetweenLineAndDirection(Line l, Direction direction){}
    public static int sideOfLine(Coordinates point, Line line){}
    public static Coordinates[] formRectagleAroundPoint(Coordinates point, Direction direction,
double length, double width){}
    public static Line[] formLinesAroundPoint(Coordinates point, Direction direction, double
length, double width){}
    public static Line lineIntersectingLines(Line line, Line[] lines){}
    public static Line ballOnTheTable(Line ball, Coordinates robot1, Direction dirR1, double
lengthR1, double widthR1, Coordinates robot2, Direction dirR2, double lengthR2, double widthR2){}
}
```

Listed below is some non-intuitive information about the functions or things that cannot be explained only with comments in the code.

public static Coordinates intersectionOfLines(Line l, Line m);
The function intersectionOfLines no longer ignores the range of each Line object. If there is no intersection point or the intersection point is infeasible for one of the lines the function returns **null.**

public Coordinates symmetricalPoint(Coordinates point, Line line);

Let (x_1, y_1) be the point given, (x_2, y_2) - the point that 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 $\vec{p} = \vec{point} - (0, line.offset)$.