BALANCING A BALL

OBJECTIVES

You should implement a simulator for balancing a ball on a beam. The ball position can be influenced by controlling the angle α of the beam relative to the ground. The beam can rotate around its center point.

The only force acting on the ball is gravity (9.81m/s²), all other forces (friction, air drag, …) can be ignored. You may also ignore the rotational energy of the ball and assume that it only translates along the beam. The force acting on the ball and the resulting acceleration in our case are therefore:

$$F = m \cdot g \cdot \sin(\alpha)$$
$$a = \frac{F}{m} = g \cdot \sin(\alpha)$$

The following limitations apply:

- The angle α lies between alpha_max and –alpha_max
- The angle α may change by no more than max_d_alpha_max
- The length of the beam is l_beam

Those values should be read using a distinct class Configuration_reader from a configuration file named beam.cfg containing the above 3 values separated by semicola, e.g.:

```
22.5;11.25;2
```

would be the content of a configuration where alpha_max is 22.5°, max_d_alpha_max is 11.25°/s and l_beam is 2m.

You should control the beam in such a way that the ball is as close to the beam's center as possible. It should not drop off the beam. The ball's initial position is at the center of the beam.

To make things a bit more challenging, it should be possible to push the ball to either side. A push results in an immediate change in velocity by 0.5m/s.

The beam's angle should be controlled by a PID controller. You should implement an application that tries to find the optimal parameters for the PID controller using gradient descent optimization. "Optimal" in this case means that the integrated absolute deviation of the ball's position from the beam's center over 10s is minimized for the following scenario:

- The ball lies in the center of the beam
- The ball is immediately pushed to the right, its velocity changes by 0.5m/s
- After 5s, the ball is pushed to the left, its velocity changes by -0.5m/s

The resulting optimal controller parameters should be written into an output file using a separate class Result_writer. The result file should contain the values for P, I and D separated by semicola, e.g.:

```
1.02;0.03;0.13251
```

The part of the application that tries to find the optimal parameters for controlling the beam should be exchangeable, in case someone wants to implement a different type of optimizer.

You should write unit tests for at least 3 of your classes, you should achieve 100% test coverage for at least one class.