```python
from pybricks.parameters import Stop
from pybricks.tools import StopWatch, wait
from threading import Thread


# Class to control forklifts
class Forklift:
    def __init__(self, config, motor, rackLength, stallDir=-1, offset=0):
        self.config = config
        self.motor = motor
        self.STALLDIR = stallDir
        self.RACKLENGTH = rackLength
        self.RATIO = 360/self.RACKLENGTH
        self.CORR = 0
        self.OFFSET = offset

    def _limit(self, input):
        return max(min(input, 180), -180)

    def _moveTo(self, angle, speed, Wait=10000):
        if self.motor.angle() > angle:
            angle = self._limit(angle + self.CORR)
        else:
            angle = self._limit(angle)
        timer = StopWatch()
        Thread(target=self.motor.run_target, args=(speed, angle),
               kwargs={"then": Stop.HOLD}).start()
        wait(50)
        while Wait > 0 and self.done() == False:
            if timer.time() > Wait:
                self.stop()
                break
            wait(50)

    def _move(self, deltaAngle, speed, wait=10000):
        # print("move: ", self.motor.angle(), deltaAngle)
        angle = self.motor.angle()
        self._moveTo(angle+deltaAngle, speed, Wait=wait)

    def initPos(self, pos=0):
        if self.config.state.getState() == 3:
            return

        self.motor.reset_angle(0)
        self.motor.run_until_stalled(200*self.STALLDIR, duty_limit=50)
        self.motor.reset_angle(
            self.STALLDIR * (self.RACKLENGTH/2*self.RATIO)+self.OFFSET)
        self.moveTo(pos, 200)
        self.stop()

    def getPos(self):
        return round(self.motor.angle() / self.RATIO)

    def stop(self):
        self.motor.stop()

    def moveTo(self, dist, speed=400, wait=10000):
        if self.config.state.getState() == 3:
            return
        self._moveTo(dist*self.RATIO, speed, wait)

    def move(self, deltaDist, speed=400, wait=10000):
        if self.config.state.getState() == 3:
            return
        self._move(deltaDist*self.RATIO, speed, wait)

    def correction(self, corr):
```

```
68            self.CORR = corr * self.RATIO
69
70        def done(self):
71            return self.motor.control.done()
72
73        def stalled(self):
74            return self.motor.control.stalled()
75
76        def getRange(self):
77            return [-self.RACKLENGTH/2, self.RACKLENGTH/2]
78
```