```python
from DRV8833 import *
from Encoder import *

class MotorController:

    def __init__(self, motor, encoder):
        '''Controller for a single motor
        motor: motor driver (DRV8833)
        encoder: motor encoder (Encoder)
        '''
        self.mot = motor
        self.end = encoder
        self.integ = 0

    def p_control(self, desired_cps, P=1):
        '''Set motor control to rotate at desired_cps'''
        actual_cps = self.end.get_cps()
        error = desired_cps - actual_cps
        self.mot.set_speed(P*error)
        # return speed (e.g. for plotting)
        return actual_cps

    # add new method:
    def pi_control(self, desired_cps, Ts, P=1, I=1):
        actual_cps = self.end.get_cps()
        error = desired_cps - actual_cps
        self.integ += error * Ts
        # clamp integrator, e.g. if desired_cps exceeds maximum motor speed
        self.integ = max(-150, min(self.integ, 150))
        self.mot.set_speed(P*error + I*self.integ)
        return actual_cps
```

```python
from DRV8833 import *
from Encoder import *

class MotorController:

    def __init__(self, motor, encoder):
        '''Controller for a single motor
        motor: motor driver (DRV8833)
        encoder: motor encoder (Encoder)
        '''
        self.mot = motor
        self.end = encoder

    def p_control(self, desired_cps, P=1):
        '''Set motor control to rotate at desired_cps'''
        actual_cps = self.end.get_cps()
        error = desired_cps - actual_cps
        self.mot.set_speed(P*error)
        # return speed (e.g. for plotting)
        return actual_cps
```

```
from MotorController import *
from DRV8833 import *
from Encoder import *

desired_cps = 100    # controller setpoint
P = 1                # controller proportional gain
Ts = 20              # controller operating period in [ms]

controller = MotorController(DRV8833(19, 16), Encoder(34, 39, 0))

def callback(timer):
    global controller, desired_cps, P
    # proportional control and print actual_cps (for plotting)
    print(controller.p_control(desired_cps, P))

timer = Timer(0)
timer.init(period=Ts, mode=Timer.UP, callback=callback(timer))
```

```
from MotorController import *
from DRV8833 import *
from Encoder import *

desired_cps = 100    # controller setpoint
P = 1                # controller proportional gain
Ts = 20              # controller operating period in [ms]

controller = MotorController(DRV8833(19, 16), Encoder(34, 39, 0))

def callback(timer):
    global controller, desired_cps, P
    # proportional control and print actual_cps (for plotting)
    print(controller.pi_control(desired_cps, P))

timer = Timer(0)
timer.init(period=Ts, mode=Timer.UP, callback=callback(timer))
```