```h
#pragma once
#include<string>
#include<vector>
#include<iostream>
class P_ctrl
{
public:
    P_ctrl(const float& P); // constructor for P controller
    float calc_ctrl_P(float error); // function that calculates control input based on provided error from the system we aim to control.
    float m_P;
};


class PI_ctrl : public P_ctrl
{
private:

protected:

public:
    float m_I;
    float m_dT;
    float m_integrator;
    float calc_ctrl_I( float error);
    PI_ctrl(const float& P,
            const float& I,
            const float& dT);
    float calc_ctrl_PI( float error);
};

class PID_ctrl : public PI_ctrl
{
private:

protected:

public:
    std::string newName;
    std::string  getName();
    std::string setName(std::string);
    float m_D;
    float m_dT;
    float m_p_error;
    float calc_ctrl_D( float error);
    //std::string m_ID; // if you'll need to compare two std::string objects you have to use string::compare (find online), because '==' operator is not defined for std::string
    PID_ctrl(float P, float I, float D, float dT, std::string ID);
    float calc_ctrl_PID( float error);
    void setD(const float& P, const float& I, const float& D); //IMPLEMENT! use references to set and get P,I and D constants.. P and I are set as protected which makes them accessible to PID class
    void getD(float& P, float& I, float& D); //IMPLEMENT!
};
```