```
//
//  EULERODE.hpp
//  ODEsolver
//
//  Created by Ben Stager on 5/3/21.
//

#ifndef EULERODE_hpp
#define EULERODE_hpp

#include <iostream>
const double defaultTimeStep = .001; // Default Time Step (sec)

class EulerODE{
public:
    // default constructor
    EulerODE();

    // constructor to set the size, initial Y(0)=0
    EulerODE(int size);

    // copy constructor
    EulerODE(const EulerODE &p);

    //  destructor
    ~EulerODE();

    // Integrate for a time step
    void incrementTime();

    // compute Y'(t)
    virtual void computeY_dot();

    // get the number of components
    int getSize() const;

    // get a component
    double getComponent(int i) const;

    // get a derivative of component
    double getDotComponent(int i) const;

    // get time step
    double getTimeStep() const;

    // set time step
    void setTimeStep(double dt);

    // get elapsed time
    double getElapsedTime() const;

    // set component
    void setComponent(int i,double v) ;

    // set a derivative of component
    void setDotComponent(int i,double vp) ;

    // export components
    friend std::ostream& operator << (std::ostream& str, const EulerODE &p);
```

```
private:
    double *vectorY; // components
    double *vectorYdot; // components derivatives
    int  vectorSize; // number of components
    double timeStep; // time step
    double elapsedTime; // Elapsed time
};
#endif /* EULERODE_hpp */
```