Answer

```
/**
 * Note there are 5 files:
 * 1. point.h
 * 2. point.cpp
 * 3. line.h
 * 4. line.cpp
 * 5. main.cpp
 *
 * use the command to compile the code:
 * g++ main.cpp line.cpp point.cpp
*/
//point.h


#ifndef POINT_H
#define POINT_H


//class point
class  Point {
   double x, y; //coordinates variables
public:
   void setX(double); //sets x coordinate
   void setY(double); //sets y coordinate
```

```
    double getX() const; //return x coordinate
    double getY() const; //return y coordinate

};

#endif
```

//point.cpp

```
#include "point.h"

//sets x coordinate
void Point::setX(double x) {
    this->x = x;
}

//sets y coordinate
void Point::setY(double y) {
    this->y = y;
}

//return x coordinate
double Point::getX() const {
    return x;
}

//return y coordinate
```

```
double Point::getY() const {
    return y;
}
```

//line.h

```
#ifndef LINE_H
#define LINE_H
#include "point.h"
#include <iostream>
using namespace std;

class line {
    Point point1, point2; //staring point and ending point
public:
    line(); //default constructor
    line(double, double, double, double); //parametrized constructor
    line(line& l); //copy constructor

    void SetPoint1(double, double); //sets starting point
    void SetPoint2(double, double); //sets ending point
    void SetLine(double, double, double, double); //sets both the points
    double Distance(); //retruns length of the line
    double Slope(); //returns slope of the line
    friend ostream& operator<<(ostream &out, const line& l); //output line to the console
    friend istream& operator>>(istream &in, line &l); //input line from the console
```

```
    bool operator==(const line& l); //returns true if two line are equal
    bool operator!=(const line& l); //returns true if two lines are not equal
};
#endif
```

//line.cpp

```
#include "line.h"
#include <cmath>

//default constructor
line::line() {}

//parametrized constructor
line::line(double x1, double y1, double x2, double y2) {
    point1.setX(x1);
    point1.setY(y1);
    point2.setX(x2);
    point2.setY(y2);
}

//copy constructor
line::line(line& l) {
    point1.setX(l.point1.getX());
    point1.setY(l.point1.getY());
    point2.setX(l.point2.getX());
    point2.setY(l.point2.getY());
```

```
}

//sets starting point
void line::SetPoint1(double x1, double y1) {
    point1.setX(x1);
    point1.setY(y1);
}

//sets ending point
void line::SetPoint2(double x2, double y2) {
    point2.setX(x2);
    point2.setY(y2);
}

//sets both the points
void line::SetLine(double x1, double y1, double x2, double y2) {
    point1.setX(x1);
    point1.setY(y1);
    point2.setX(x2);
    point2.setY(y2);
}

//retruns length of the line
double line::Distance() {
    //using distance formula to calculate the length
    double distance = sqrt(pow(point2.getX() - point1.getX(), 2) + pow(point2.getY() - point1.getY(), 2));
```

```
    return distance;
}

//returns slope of the line
double line::Slope() {

    //using the formual: y2-y1 / x2-x1 to calculate the slope
    double slope =  (point2.getY() - point1.getY()) / (point2.getX() - point1.getX());
    return slope;
}

//returns true if two line are equal
bool line::operator==(const line& l) {

    //checking all the corresponding coordinates in both the lines
    //all the corresponding coordinates need to be equal
    if ((point1.getX() == l.point1.getX()) &&
        (point1.getY() == l.point1.getY()) &&
        (point2.getX() == l.point2.getX()) &&
        (point2.getY() == l.point2.getY()))
       return true;

    return false;
}

//returns true if two lines are not equal
```

```
bool line::operator!=(const line& l) {
    //checking all the corresponding coordinates in both the lines
    //even if a single coordinate is found unequal it returns true
    if ((point1.getX() != l.point1.getX()) ||
        (point1.getY() != l.point1.getY()) ||
        (point2.getX() != l.point2.getX()) ||
        (point2.getY() != l.point2.getY()))
       return true;

    return false;
}

//output line to the console
ostream& operator<<(ostream &out, const line& l) {
    out << "(" << "(" << l.point1.getX() << "," << l.point1.getY()  << ")" << "
(" << l.point2.getX() << "," << l.point2.getY() <<  ")" << ")";
    return out;
}

//input line from the console
istream& operator>>(istream &in, line &l) {
    double x1, y1, x2, y2;
    in >> x1 >> y1 >> x2 >> y2;
    l.point1.setX(x1);
    l.point1.setY(y1);
    l.point2.setX(x2);
    l.point2.setY(y2);
```

```
    return in;
}
```

//main.cpp

```
#include "line.h"

int main() {
    line l1(1, 2, 3, 4); //using constructor
    line l2(l1); //using copy constructor

    //settting points using methods
    line l3;
    l3.SetPoint1(5, 6);
    l3.SetPoint2(9, 10);

    //printing lines
    cout << "line l1: " << l1 << endl;
    cout << "line l2: " << l2 << endl;
    cout << "line l3: " << l3 << endl;

    //reading line from the user
    line l4;
    cout << "Enter line L4: ";
    cin >> l4;
    cout << "line L4: " << l4 << endl;

    //calling Distance() and Slope() method
```

```
    cout << "length of L1: " << l1.Distance() << endl;
    cout << "slope of L1: " <<  l1.Slope() << endl;

    //Demonstrating comparison operators
    if (l1 == l2) {
        cout << "yes L1 is equal to L2\n";
    }

    if (l2 != l3) {
        cout << "L2 is not equal to L3\n";
    }
}
```

View image!

Likes: 0 **Dislikes: 0**
