EUROPEAN UNIVERSITY OF LEFKE

Faculty of Engineering

# Department of Computer Engineering



# COMP218

OBJECT-ORIENTED PROGRAMMING

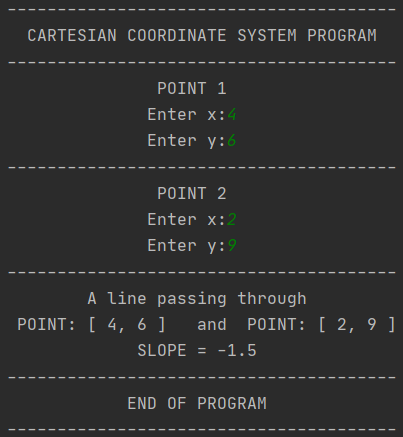
## Lab Work No. 7

Prepared by Seward Richard Mupereri (20140175)

Submitted to Dr. Ferhun Yorgancıoğlu

### Task (1)

#include <iostream>  
#include <iomanip>  
  
using namespace std;  
  
class Point {  
public:  
 Point();  
 Point( int, int );  
 Point( const Point& );  
 ~Point();  
  
 void set\_x( int );  
 void set\_y( int );  
  
 int get\_x();  
 int get\_y();  
  
 void print();  
  
private:  
 int x, y;  
  
};  
  
Point::Point() : x( 0 ), y( 0 ) {}  
  
Point::Point( int x, int y )  
{  
 set\_x( x );  
 set\_y( y );  
}  
  
Point::Point( const Point& copy ) : x( copy.x ), y( copy.y ) {}  
  
Point::~Point(){}  
  
void Point::set\_x( int x )  
{  
 this->x = x;  
}  
  
void Point::set\_y( int y )  
{  
 this->y = y;  
}  
  
int Point::get\_x()  
{  
 return x;  
}  
  
int Point::get\_y()  
{  
 return y;  
}  
  
void Point::print()  
{  
 cout << " POINT: [ " << x << ", " << y << " ]";  
}  
  
class Line {  
public:  
 Line();  
 Line( Point, Point );  
 Line( const Line& );  
 ~Line();  
  
 void set\_point1( Point );  
 void set\_point2( Point );  
  
 double get\_slope();  
  
 void print();  
  
private:  
 Point p1, p2;  
  
};  
  
Line::Line(){}  
  
Line::Line ( Point p1, Point p2 )  
{  
 set\_point1( p1 );  
 set\_point2( p2 );  
}  
  
Line::Line( const Line& copy ) : p1(copy.p1), p2(copy.p2) {}  
  
Line::~Line() {}  
  
void Line::set\_point1( Point p )  
{  
 p1 = p;  
}  
  
void Line::set\_point2( Point p )  
{  
 p2 = p;  
}  
  
double Line::get\_slope()  
{  
 return ( (double) ( p1.get\_y() - p2.get\_y() ) / ( p1.get\_x() - p2.get\_x() ) );  
}  
  
void Line::print()  
{  
 cout << " A line passing through " << endl;  
 p1.print();  
 cout << " and ";  
 p2.print();  
 cout << endl;  
 cout << " SLOPE = " << get\_slope();  
}  
  
int main()  
{  
 int x, y;  
  
 cout << "---------------------------------------" << endl;  
 cout << " CARTESIAN COORDINATE SYSTEM PROGRAM " << endl;  
 cout << "---------------------------------------" << endl;  
  
 cout << setw(22) << "POINT 1" << endl;  
 cout << setw(22) << "Enter x:";  
 cin >> x;  
  
 cout << setw(22) << "Enter y:";  
 cin >> y;  
  
 Point p1( x, y );  
  
 cout << "---------------------------------------" << endl;  
  
 cout << setw(22) << "POINT 2" << endl;  
 cout << setw(22) << "Enter x:";  
 cin >> x;  
  
 cout << setw(22) << "Enter y:";  
 cin >> y;  
  
 Point p2( x, y );  
  
 Line a( p1, p2 );  
  
 cout << "---------------------------------------" << endl;  
  
 a.print();  
  
 cout << endl;  
 cout << "---------------------------------------" << endl;  
 cout << " END OF PROGRAM " << endl;  
 cout << "---------------------------------------" << endl;  
  
 return 0;  
}



### Task (2)

**Header file – line.h**

#ifndef LABWORK7\_LINE\_H  
#define LABWORK7\_LINE\_H  
  
class Point {  
public:  
 Point();  
 Point( int, int );  
 Point( const Point& );  
 ~Point();  
  
 void set\_x( int );  
 void set\_y( int );  
  
 int get\_x();  
 int get\_y();  
  
 void print();  
  
private:  
 int x, y;  
  
};  
  
class Line {  
public:  
 Line();  
 Line( Point, Point );  
 Line( const Line& );  
 ~Line();  
  
 void set\_point1( Point );  
 void set\_point2( Point );  
  
 double get\_slope();  
  
 void print();  
  
private:  
 Point p1, p2;  
  
};  
#endif //LABWORK7\_LINE\_H

**Implementation file – line.cpp**

#include <iostream>  
#include "line.h"  
using namespace std;  
  
Point::Point() : x( 0 ), y( 0 ) {}  
  
Point::Point( int x, int y )  
{  
 set\_x( x );  
 set\_y( y );  
}  
  
Point::Point( const Point& copy ) : x( copy.x ), y( copy.y ) {}  
  
Point::~Point(){}  
  
void Point::set\_x( int x )  
{  
 this->x = x;  
}  
  
void Point::set\_y( int y )  
{  
 this->y = y;  
}  
  
int Point::get\_x()  
{  
 return x;  
}  
  
int Point::get\_y()  
{  
 return y;  
}  
  
void Point::print()  
{  
 cout << " POINT: [ " << x << ", " << y << " ]";  
}  
  
  
  
Line::Line(){}  
  
Line::Line ( Point p1, Point p2 )  
{  
 set\_point1( p1 );  
 set\_point2( p2 );  
}  
  
Line::Line( const Line& copy ) : p1(copy.p1), p2(copy.p2) {}  
  
Line::~Line() {}  
  
void Line::set\_point1( Point p )  
{  
 p1 = p;  
}  
  
void Line::set\_point2( Point p )  
{  
 p2 = p;  
}  
  
double Line::get\_slope()  
{  
 return ( (double) ( p1.get\_y() - p2.get\_y() ) / ( p1.get\_x() - p2.get\_x() ) );  
}  
  
void Line::print()  
{  
 cout << " A line passing through " << endl;  
 p1.print();  
 cout << " and ";  
 p2.print();  
 cout << endl;  
 cout << " SLOPE = " << get\_slope();  
}

**Driver file – main.cpp**

#include <iostream>  
#include <iomanip>  
#include "line.h"  
  
using namespace std;  
  
int main()  
{  
 int x, y;  
  
 cout << "---------------------------------------" << endl;  
 cout << " CARTESIAN COORDINATE SYSTEM PROGRAM " << endl;  
 cout << "---------------------------------------" << endl;  
  
 cout << setw(22) << "POINT 1" << endl;  
 cout << setw(22) << "Enter x:";  
 cin >> x;  
  
 cout << setw(22) << "Enter y:";  
 cin >> y;  
  
 Point p1( x, y );  
  
 cout << "---------------------------------------" << endl;  
  
 cout << setw(22) << "POINT 2" << endl;  
 cout << setw(22) << "Enter x:";  
 cin >> x;  
  
 cout << setw(22) << "Enter y:";  
 cin >> y;  
  
 Point p2( x, y );  
  
 Line a( p1, p2 );  
  
 cout << "---------------------------------------" << endl;  
  
 a.print();  
  
 cout << endl;  
 cout << "---------------------------------------" << endl;  
 cout << " END OF PROGRAM " << endl;  
 cout << "---------------------------------------" << endl;  
  
 return 0;  
}

