**Programming Languages (Java), 2ed lab**

**Task 1**

Create a Point class containing two fields, x and y of type double. Create a method move(dx,dy) which can translate a point with coordinates dx and dy. Create a method mirror(cx,cy) which can reflect a point through the coordinates cx and cy.

class Point {

double x, y; //ins. variable

void move( double dx, double dy ){

x += dx; //x=x+dx;

y += dy;

}

void mirror( double cx, double cy ){

x = 2\*cx-x;

y = 2\*cy-y;

}

}

Create a PointMain program to test the use of the Point class.

class PointMain {

public static void main( String[] args ){

Point p = new Point();

p.x = 2;

p.y = 3;

p.move(1,1);

System.out.println(p.x + " " + p.y);

p.mirror(0,0);

System.out.println(p.x + " " + p.y);

}

}

**Task 2**

Rewrite the mirror method in class Point in such a way that it has a single parameter of type Point, representing the point through which the reflection should be performed.

Create a distance(p) method, which computes the distance of a point from the point p (provided as parameter). Use the Math.sqrt(...) function and the Pythagorean theorem.

Update the PointMain class to test the new operations!

class Point {

double x, y;

void move( double dx, double dy ){

x += dx;

y += dy;

}

void mirror( double cx, double cy ){

x = 2\*cx-x;

y = 2\*cy-y;

}

void mirror( Point c ){

x = 2\*c.x-x;

y = 2\*c.y-y;

}

double distance( Point that ){

double dx = x - that.x,

dy = y - that.y;

return Math.sqrt( dx\*dx + dy\*dy );

}

}

**class PointMain {**

**public static void main( String[] args ){**

**Point p = new Point();**

**p.x = 2;**

**p.y = 3;**

**p.move(1,1);**

**System.out.println(p.x + " " + p.y);**

**p.mirror(0,0);**

**System.out.println(p.x + " " + p.y);**

**Point origin = new Point();**

**origin.x = origin.y = 0;**

**p.mirror(origin);**

**System.out.println(p.x + " " + p.y);**

**System.out.println( p.distance(origin) );**

**}**

**}**

### Task 3

Implement the Complex class representing complex numbers with a real and imaginary component. The class should contain fields re and im of type double. Create a method abs() which computes the absolute value of the complex number.

Implement the add(c) and sub(c) and mul(c) methods, which, respectively, add another complex number c to a complex number, subtract c from the complex number and multiply the complex number with c. These methods should be used as, and should behave like, illustrated below.

alpha.re = 3

alpha.im = 2

beta.re = 1

beta.im = 2

alpha.add(beta)

// [alpha.re](http://alpha.re/) == 4 && [alpha.im](http://alpha.im/) == 4 && [beta.re](http://beta.re/) == 1 && [beta.im](http://beta.im/) == 2

https://www2.clarku.edu/faculty/djoyce/complex/abs.html#:~:text=For%20a%20complex%20number%20z,on%20the%20real%20number%20line.

class Complex {

double re, im;

double abs() {

return Math.sqrt(re\*re + im\*im);

}

void add(Complex that) {

re += that.re;

im += that.im;

}

void sub(Complex that) {

re -= that.re;

im -= that.im;

}

void mul(Complex that) {

double new\_re = this.re\*that.re - this.im\*that.im;

double new\_im = this.re\*that.im + this.im\*that.re;

re = new\_re;

im = new\_im;

}

}

class ComplexMain {

public static void main(String[] args) {

Complex num1 = new Complex(); //re,im are both instance variables so we should create object to call them (num1)

num1.re = 3;

num1.im = -2;

Complex num2 = new Complex();

num2.re = 1;

num2.im = 4;

System.out.println("abs(3 - 2i) = " + num1.abs());

System.out.print("("+ num1.re + " + " + num1.im + "i)");

System.out.print(" + ");

System.out.print("("+ num2.re + " + " + num2.im + "i)");

System.out.print(" = ");

num1.add(num2);

System.out.println( num1.re + " + " + num1.im + "i" );

System.out.print("("+ num1.re + " + " + num1.im + "i)");

System.out.print(" - ");

System.out.print("("+ num2.re + " + " + num2.im + "i)");

System.out.print(" = ");

num1.sub(num2);

System.out.println( num1.re + " + " + num1.im + "i" );

System.out.print("("+ num1.re + " + " + num1.im + "i)");

System.out.print(" \* ");

System.out.print("("+ num2.re + " + " + num2.im + "i)");

System.out.print(" = ");

num1.mul(num2);

System.out.println( num1.re + " + " + num1.im + "i" );

}

}

### Task 4

Create a class Circle representing circles with an origin (fields x and y of type double) and a radius (field of type double). Create a method enlarge(f), which multiplies the radius with f, enlarging the circle with a factor f (or shrinking if f is smaller than 1). Create a method getArea() which computes the area of the circle. Use the constant Math.PI!

class Circle {

double x, y;

double radius;

void enlarge(double factor) {

radius \*= factor;

}

double getArea() {

return Math.PI\*radius\*radius;

}

}

**class CircleMain {**

**public static void main( String[] args ){**

**Circle c = new Circle();**

**c.x = 2;**

**c.y = 3;**

**c.radius= 5;**

**System.out.println("the area before is =" + c.getArea());**

**c.enlarge(2);**

**c.getArea();**

**System.out.println("the area after is=" + c.getArea());**

**}**

**}**

### Task 5

Create class Line, which can represent lines on a plane. A line is characterized by its points (x, y), which satisfy the equality ax + by = c. The fields of the Line class will be a, b and c, all of type double.

Create a method contains(p), which decides whether the point p lies on the line.

Create the methods isParallelWith() and isOrthogonalTo(), which decide, respectively, whether a line is parallel to, and vertical to the line l provided as a parameter.

https://www.youtube.com/watch?v=-DnmoFnQUtk

**class Line {**

**double a, b, c;**

**boolean contains( Point p ){**

**return a\*p.x + b\*p.y == c;**

**}**

**/\* Ax+B=y ax + by = c (-a/b)x + c/b = y**

**\* parallel if the slope (A) is the same**

**\* if b == 0 then it is dependent \*/**

**boolean isParallelWith( Line that ){**

**return (this.a \* that.b == that.a \* this.b);**

**}**

**boolean isOrthogonalTo( Line that ){**

**return (this.a \* that.a + this.b \* that.b == 0);**

**}**

**}**

**class LineMain {**

**public static void main( String[] args ){**

**Line e = new Line(), f = new Line();**

**e.a = Double.parseDouble(args[0]);**

**e.b = Double.parseDouble(args[1]);**

**e.c = Double.parseDouble(args[2]);**

**f.a = Double.parseDouble(args[3]);**

**f.b = Double.parseDouble(args[4]);**

**f.c = Double.parseDouble(args[5]);**

**System.out.println( e.isParallelWith(f) );**

**System.out.println( e.isOrthogonalTo(f) );**

**}**

**}**

### Task 6

Write the Distance program, which interprets its command line arguments as a sequence of points, and sums up the distances found between any two consecutive points (i.e. the length of a polygonal chain, or polyline). The first two command line arguments represent the x and y coordinates of the first point, then the third and fourth command line arguments represent the second point, and so on. We can assume that the number of command line arguments is even, and each command line argument is an integer number.

The program should work by using the Point class and its distance method. For example, if the six coordinates of three points are provided as command line arguments, the program should compute the distance between the first two points, as well as the distance of the last two points, and print the sum of these two distances.

Examples:

> java Distance

0.0

> java Distance 1 2

0.0

> java Distance 0 0 3 4

5.0

> java Distance 1 2 4 6

5.0

> java Distance 1 2 4 6 7 6

8.0

<https://predictivehacks.com/tip-how-to-define-your-distance-function-for-hierarchical-clustering/>

First you should update the Point class to include the getDistance method.

class Point {

double x, y;

void move( double dx, double dy ){

x += dx;

y += dy;

}

void mirror( double cx, double cy ){

x = 2\*cx-x;

y = 2\*cy-y;

}

void mirror( Point c ){

x = 2\*c.x-x;

y = 2\*c.y-y;

}

double distance( Point that ){

double dx = x - that.x,

dy = y - that.y;

return Math.sqrt( dx\*dx + dy\*dy );

}

static double getDistance (Point a, Point b) {

double dx = a.x - b.x, dy = a.y - b.y;

return Math.sqrt(dx\*dx + dy\*dy);

}

}

**class Distance {**

**public static void main(String[] args) {**

**double distance = 0;**

**for(int i = 0; i < args.length-2; i=i+2) {**

**Point p1 = new Point();**

**p1.x = Integer.parseInt(args[i]);**

**p1.y = Integer.parseInt(args[i+1]);**

**Point p2 = new Point();**

**p2.x = Integer.parseInt(args[i+2]);**

**p2.y = Integer.parseInt(args[i+3]);**

**distance += Point.getDistance(p1,p2);**

**}**

**System.out.println(“distance is=” + distance);**

**}**

**}**