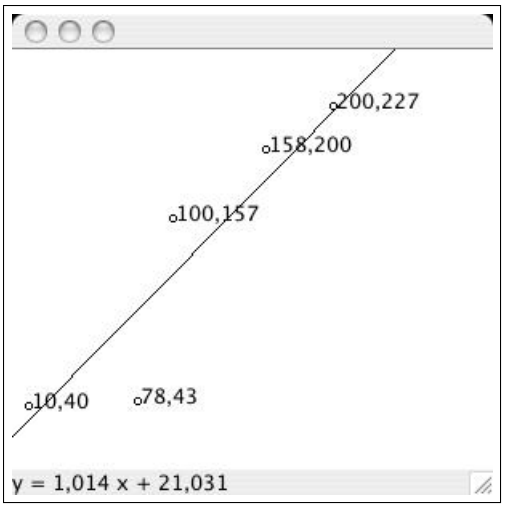



```
10 40
100 157
78 43
200 227
275 290
158 200
```

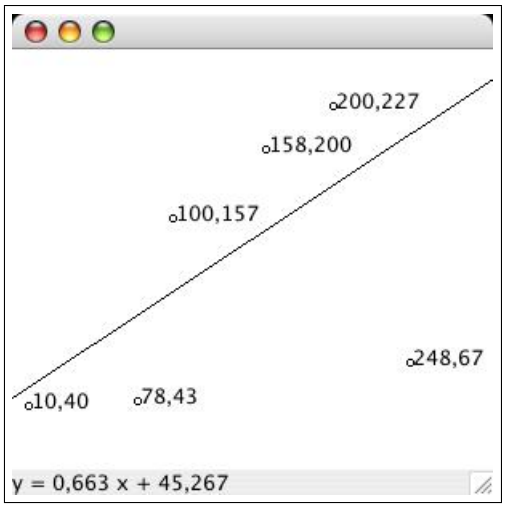


Figure 1: Application (fenêtre de 300 pixels par 300 pixels) de régression après la lecture du fichier de données contenant les 6 points donnés au milieu de la figure. L'image de droite décrit la nouvelle courbe après l'ajout du point `(248,67)` en bas à droite avec la souris.

(60) 1. Vous allez développer une application graphique Java, voir la figure ci-dessus, qui va tracer une *droite de régression* qui donne la meilleure approximation d'un ensemble de points qui sont lus sur un fichier. L'équation de la droite doit aussi être affichée en bas de la fenêtre. Les points doivent être affichés dans une zone de dessin au-dessus de l'étiquette avec un cercle de rayon 2 centré sur le point. Lorsqu'ensuite l'usager clique dans la fenêtre, ceci ajoute un nouveau point à cette position dans la fenêtre d'affichage de l'application. La droite de regression et son équation doivent être réajustées à chaque nouveau point *cliqué* par l'usager.

Une droite de régression est donnée par l'équation suivante:

$$y = \bar{y} + m(x - \bar{x}) \text{ où } \begin{cases} m = \dfrac{\sum_{i=0}^{n-1} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=0}^{n-1} x_i^2 - n\bar{x}^2} \\ n \text{ est le nombre de points} \\ \bar{x} = (\sum_{i=0}^{n-1} x_i)/n \\ \bar{y} = (\sum_{i=0}^{n-1} y_i)/n \end{cases}$$

Pour calculer la droite à afficher, il suffit de calculer la valeur y de la droite pour $x = 0$ et pour x égal à la largeur de la fenêtre et de tracer la ligne entre les ces deux points. La forme de l'équation peut être déduite facilement une fois connus m, $\bar{x}$ et $\bar{y}$.

Pour y arriver, vous allez procéder en 3 étapes indépendantes: vous pouvez effectuer une étape même si vous n'avez pas répondu aux précédentes en supposant ses méthodes comme définies.

(20) (a) Créez une classe `Points` qui permet de conserver une série de points x, y et d'en obtenir des informations en complétant le squelette de classe suivant (répondez en n'indiquant que ce que vous voulez ajouter à la place des lettres de la forme `... L ...` où `L` est `A`,`B`,`C` ou `D`):

```
public class Points {

    private int[] xs,ys;  // points des données
    private int n;
    private int sumXi,sumYi,sumXi2,sumXiYi; // cumulatifs pour accélérer les calc

    Points (){
        xs = new int[10];
        ys = new int[10];
        n =  0;
        n = sumXi = sumYi = sumXi2 = sumXiYi = 0;
    }

    // accès aux valeurs internes
    public int[]  getXs()  { return xs;}
    public int[]  getYs()  { return ys;}
    public int    getN()   { return n;}

    // calcul de la moyenne
    public double getXBar(){
       // ... A ... (5 pts)
       return sumXi/n;
    }
    public double getYBar(){
       // ... B ... (5 pts)
       return sumYi/n;
    }

    //  calcul de la pente de la courbe
    public double getM(){
        // ... C ... (5 pts)
        double xBar = getXBar();
        return (sumXiYi-n*xBar*getYBar())/(sumXi2-n*xBar*xBar);
    }

    // ajouter un point en allongeant les tableaux xs et ys si nécessaire
```

```
    public void ajoute(int x, int y){
       // ... D ... (5 pts)
        if(n==xs.length){
            // allonger le tableau si nécessaire
            int[] newXs = new int[xs.length*2];
            int[] newYs = new int[ys.length*2];
            for(int i=0;i<xs.length;i++){
                newXs[i]=xs[i];newYs[i]=ys[i];
            }
            xs = newXs;
            ys = newYs;
        }
        // ajouter le point
        xs[n]=x; ys[n]=y;n++;
        // mise à jour des valeurs cumulatives
        sumXi += x;
        sumYi += y;
        sumXi2 += x*x;
        sumXiYi += x*y;
    }
}
```

(20) (b) Complétez la classe suivante qui définit la surface de traçage.

```
public class Dessin extends JPanel implements MouseListener{
    private Points pts;
    private JLabel l;

    Dessin(Points pts,JLabel l){
       // ... A ... (5 pts)
        setBackground(Color.white);
        l.setBackground(Color.white);
        this.pts = pts;
        this.l = l;
        addMouseListener(this);
    }


    //  affichage des points et de l'équation
    public void paintComponent(Graphics g){
       // ... B ... (10 pts)
        super.paintComponent(g);
        int w = getSize().width;
        int h = getSize().height;
        int n = pts.getN();
```

```
            if(n==0)return;
            for(int i=0;i<n;i++){
                g.drawOval(pts.getXs()[i]-2,h-pts.getYs()[i]-2,4,4);
                g.drawString(pts.getXs()[i]+","+pts.getYs()[i],pts.getXs()[i]+2,h-pts
            }
            int x0 = 0; double y0 = pts.getYBar()+pts.getM()*(x0-pts.getXBar());
            int x1 = w; double y1 = pts.getYBar()+pts.getM()*(x1-pts.getXBar());
            g.drawLine(x0,(int)Math.round(h-y0),x1,(int)Math.round(h-y1));
            // mise à jour et affichage du label
            double m = pts.getM();
            double b = pts.getYBar()-m*pts.getXBar();
            l.setText("y = "+fmt.format(m)+" x + "+ fmt.format(b));
            l.repaint();
        }

        // traitement des clics de souris
        public void mousePressed(MouseEvent event){
           // ... C ... (5 pts)
            pts.ajoute(event.getX(),getSize().height-event.getY());
            repaint();
        }
        // méthodes ignorées
        public void mouseClicked(MouseEvent event){}
        public void mouseReleased(MouseEvent event){}
        public void mouseEntered(MouseEvent event){}
        public void mouseExited(MouseEvent event){}

    }
```

(20) (c) Complétez la classe suivante qui définit les éléments d'interface, qui lit le fichier dont le nom est donné en paramètre de l'application et qui lance cette dernière.

```
public class Regression {
    public static void main (String args[]) {
        JFrame f = new JFrame();

        // ... A ... initialisation de l'application (10 pts)
        Points pts = new Points();
        JLabel l = new JLabel("y = a  x + b");
        Dessin d = new Dessin(pts,l);
        f.getContentPane().add(BorderLayout.SOUTH,l);
        f.getContentPane().add(d);

        // ... B ... traitement du fichier (10 pts)
        if(args.length>0){
```

```
                try {
                    String ligne = null;
                    BufferedReader in = new BufferedReader(new FileReader(args[0]));
                    while((ligne=in.readLine())!=null && ligne.trim().length()>0){
                        StringTokenizer st = new StringTokenizer(ligne);
                        pts.ajoute(new Integer(st.nextToken()).intValue(),
                                   new Integer(st.nextToken()).intValue());
                    }
                    System.out.println(pts.getN()+" points lus sur "+args[0]);
                    f.repaint();
                } catch (FileNotFoundException e){
                    System.out.println("pas trouvé fichier "+args[0]);
                } catch (IOException e){
                    System.out.println("erreur d'entree-sortie");
                }
            }

            f.setSize(300,300);
            f.setLocation(100,100);
            f.setVisible(true);
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
    }
```

(40) 2. Définition de classes avec héritage. Il ne faut redéfinir de méthodes qui pourraient être héritées.

(10) (a) Définissez une classe abstraite `Triangle` ayant comme constructeur les entiers `x1,y1,x2,y2,x3,y3` correspondant aux coordonnées des sommets du triangle. Définissez également les deux méthodes suivantes:

- la méthode `perimetre` qui retourne le périmètre (i.e. la somme des longueurs des côtés du triangle)
- la méthode `surface` donnée par la formule de Héron suivante:

$$\sqrt{s(s-a)(s-b)(s-c)}$$

où s est le demi-périmètre et a, b, c sont les longueurs des côtés du triangle.

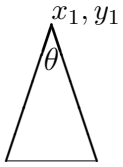
```
public abstract class Triangle {

    protected int    x1,y1,x2,y2,x3,y3;
    protected double a,    b,    c; // longueurs de côtés opposés aux sommets

    Triangle(int x1,int y1,int x2,int y2,int x3,int y3){
        this.x1=x1;this.y1=y1;
        this.x2=x2;this.y2=y2;
        this.x3=x3;this.y3=y3;
        a=Point2D.distance(x2,y2,x3,y3);
        b=Point2D.distance(x1,y1,x3,y3);
        c=Point2D.distance(x1,y1,x2,y2);
    }
    public double perimetre(){
        return a+b+c;
    }
    public double surface(){
        double s=perimetre()/2;
        return Math.sqrt(s*(s-a)*(s-b)*(s-c));
    }
}
```

(10) (b) Définissez `Scalene` une sous-classe de `Triangle` dont le constructeur a `x1,y1,x2,y2,x3,y3` comme paramètres et qui a comme méthodes:

- `perimetre` qui retourne son périmètre
- `surface` qui retourne sa surface

```
public class Scalene extends Triangle{
    Scalene(int x1,int y1,int x2,int y2,int x3,int y3){
        super(x1,y1,x2,y2,x3,y3);
    }
}
```

(10) (c) Définissez `Isocele` une sous-classe de `Triangle` qui représente un triangle dont deux des côtés sont égaux autour d'un angle à un sommet. Définissez, si c'est nécessaire



- le constructeur de la classe `Isocele` ayant comme paramètres `x1,y1` le sommet où terminent les deux côtés égaux, `theta` l'angle à ce sommet ainsi que `longueur` la longueur d'un des deux cotés égaux (indiqué en gras dans le schéma ci-contre)
- la méthode `perimetre` qui retourne le périmètre (i.e. la somme des longueurs des côtés du triangle)
- la méthode `surface` qui calcule la surface en calculant le demi-produit de la hauteur par la base.

```
public class Isocele extends Triangle{
    protected double base,hauteur;

    Isocele(int x1,int y1,int longueur, double angle){
        super(x1,y1,
              x1-(int)Math.round(longueur*Math.sin(angle/2)),
              y1+(int)Math.round(longueur*Math.cos(angle/2)),
              x1+(int)Math.round(longueur*Math.sin(angle/2)),
              y1+(int)Math.round(longueur*Math.cos(angle/2)));
        base = x3-x2;
        hauteur = y2-y1;
    }

    public double surface(){
        return (base*hauteur)/2;
    }
}
```

(10) (d) Définissez `Equilateral` une sous-classe de `Isocele` qui représente un triangle dont les trois côtés sont égaux autour d'un angle à un sommet. Définissez, si c'est nécessaire

- le constructeur de la classe `Equilateral` ayant comme paramètres `x1,y1` un sommet ainsi que `longueur` la longueur d'un des cotés
- la méthode `perimetre` qui retourne le périmètre (i.e. la somme des longueurs des côtés du triangle)
- la méthode `surface` qui calcule la surface en calculant le demi-produit de la hauteur par la base.

```
public class Equilateral extends Isocele{
    Equilateral(int x1, int y1, int cote){
        super(x1,y1,cote,Math.PI/3);
    }
}
```