Les sources des exercices de Travaux Pratiques sont disponibles sur le site de l'UE en
`http://remi.morin.myspace.luminy.univmed.fr/I2/PP`

Récupérez l'archive compressée `TPA.tgz` disponible sur le site de l'UE et décompressez-la avec
`tar -xvzf TPA.tgz`
Vous obtiendrez les codes utilisés en cours, en travaux dirigés et dans cette planche de travaux pratiques.

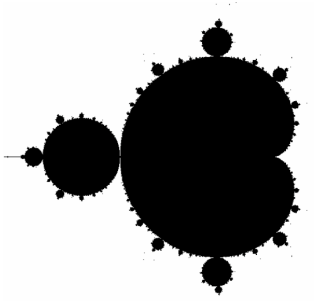
Exercice A.1 Partage statique et dynamique d'une tâche L'ensemble de Mandelbrot est une fractale définie comme l'ensemble des points c du plan complexe pour lesquels la suite définie par : $z_{n+1} = z_n^2 + c$ avec $z_0 = 0$ ne tend pas, en module, vers l'infini. Si nous reformulons cela sans utiliser les nombres complexes, en remplaçant z_n par le couple de réels (x_n, y_n) et c par le couple (a, b), alors nous obtenons :

$$x_{n+1} = x_n^2 - y_n^2 + a \text{ et } y_{n+1} = 2.x_n.y_n + b \text{ avec } x_0 = y_0 = 0$$

la condition devenant que ni x_n, ni y_n ne tendent vers l'infini (en valeur absolue).

Dans la pratique, nous allons calculer les premiers termes de cette suite pour un point (a, b) fixé. Si la valeur de $x_n^2 + y_n^2$ dépasse 4, nous considérerons (de manière arbitraire) que la suite diverge, et donc que le point (a, b) n'est pas dans l'ensemble. En revanche si la valeur $x_n^2 + y_n^2$ ne dépasse 4 pour aucun des termes calculés, nous considérerons que la suite ne diverge pas, et donc que le point (a, b) est dans l'ensemble. Ceci nous conduit à utiliser la méthode suivante, qui détermine si oui ou non le point (a, b) est dans l'ensemble de Mandelbrot lorsque l'on examine les `max` premiers éléments de la suite (x_n, y_n).

```
static boolean mandelbrot(double a, double b, int max) {
  double x = 0;
  double y = 0;
  for (int t = 0; t < max; t++) {
    if (x*x + y*y > 4.0) return false;
    double nx = x*x - y*y + a;
    double ny = 2*x*y + b;
    x = nx;
    y = ny;
  }
  return true;
}
```



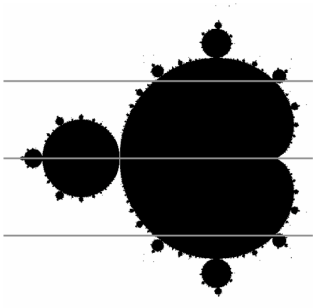
Le programme `Mandelbrot.java` disponible dans l'archive `TPA.tgz` est donné sur la figure 4. Il calcule et affiche l'ensemble de Mandelbrot situé dans la région carrée comprise entre $(-1.5, -1)$ en bas à gauche et $(0.5, 1)$ en haut à droite. Le nombre de pixels par ligne (et par colonne) est fixé à `taille=500` alors que le nombre maximal d'itérations vaut `max=200`. Le programme indique également le temps de calcul de l'image. Vous pouvez annuler l'affichage de l'image en supprimant l'instruction `image.show()`.

Question 1. Modifiez les valeurs de `taille` ou de `max` pour que le calcul de l'image sur votre machine dure une trentaine de secondes.

Question 2. Partagez ensuite le calcul de l'image sur 4 threads, chacun calculant un quart de l'image.

Question 3. Quel est le gain en temps de calcul par rapport à la version séquentielle ? Quel est le temps de calcul de chacun des quatre threads ?

Question 4. Attribuez à présent à la volée les lignes de calculs aux 4 threads au fur et à mesure qu'ils sont disponibles. Quel est le nouveau gain, en temps de calcul, par rapport à la version séquentielle ?



Exercice A.2 Evaluation de $\pi/4$ par la méthode de Monte-Carlo Nous poursuivons l'exercice I.2 de la page 1. À partir du code séquentiel du programme `Calcul_PiSurQuatre.java` donné dans l'archive et indiqué sur la figure 1, coder et tester une version parallèle utilisant 10 threads pour accélérer ce calcul.