TP3 Java : Interface graphique

Instructions pour l'envoi de votre code

1. Vous êtes libres de discuter entre vous mais le code est *individuel*.
2. Chaque méthode (fonction) doit être testée sur au moins deux exemples. Inclure les tests dans votre archive.
3. Faire une archive (`.zip` ou `.tar`) de tous vos fichiers sources (`.java`) et l'envoyer à `omar.fawzi@ens-lyon.fr` avant samedi 4 octobre 23h59. Merci d'inclure dans le sujet `[Proj1] [TP3]` et de nommer votre archive `NomPrenomTP3.zip`

Ressources

Vous pourrez utiliser les ressources suivantes :

— http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-java/notre-premiere-fenetre
— http://docs.oracle.com/javase/tutorial/uiswing/index.html

Quelques mots sur le projet

Le projet à faire d'ici debut novembre sera un outil de "clustering". Le programme prendra en entrée des données, qui peuvent être des points dans le plan, des images, des textes, etc... et va devoir les répartir en "clusters". Par exemple, si nos données correspondent à différentes photos de k personnes, chaque cluster sera idéalement composé de l'ensemble des photos d'une personne, on aura donc k clusters. Dans ce TP, nous allons commencer par faire une interface graphique qui affiche des points lus à partir d'un fichier texte et implémenter un des algorithmes les plus simples de clustering : l'algorithme des k-moyennes.

Interface graphique en Java

Nous allons utiliser le package `javax.swing` pour créer des fenêtres, boutons, etc...

Exercice 1 *Faire un programme qui lit dans un fichier un ensemble de points, puis créer une fenêtre* `JFrame` *et affiche ces points. Vous pourrez par exemple créer une classe* `Point`*, une classe* `DataDisplayPanel` *qui hérite de* `JPanel` *et une classe qui contient la fonction* `main` *dans laquelle vous allez créer une* `JFrame` *et y rajouter un* `DataDisplayPanel` *en utilisant la fonction* `setContentPane(pane)`*.*

Pour détecter les événements qui surviennent dans un composant, on utilise un `ActionListener`. On rajoute un objet de type `ActionListener` à un composant en utilisant la fonction `addActionListener(listener)`, où listener doit être un objet qui implémente l'interface `ActionListener`. Cet objet peut par exemple être le composant lui même à condition que l'on définisse une fonction `actionPerformed(ActionEvent e)`. Une interface est une classe dont les méthodes ne sont pas implémentées. Pour implémenter une interface on utilise le

mot clé `implements`. Pour détecter un clic de souris, c'est le même principe. Cette fois, on utilise la fonction `addMouseListener(listener)`, et listener doit implémenter l'interface `MouseListener`.

Exercice 2 *Permettre à l'utilisateur de rajouter des points qu'on appellera des centres (qui représente la moyenne d'un cluster). A chaque clic, ce point devra s'afficher avec une couleur choisie au hasard.*
Rajouter un bouton `JButton` *à votre fenêtre que l'utilisateur devra cliquer avant de pouvoir entrer les centres.*

L'algorithme des k-moyennes est simple. On commence avec k centres $m_1, \ldots, m_k$ (donnés ici par l'utilisateur). Pour chaque point $p_1, \ldots, p_n$, on assigne le centre le plus proche (distance Euclidienne) : ceci nous permet de définir k groupes. Pour améliorer le clustering, on va mettre à jour les centres : m'_j devient la moyenne de tous les points dans le groupe j, pour chaque $j \in \{1, \ldots, k\}$. On répète ces deux étapes jusqu'à convergence.

Exercice 3 *Rajouter un bouton à votre fenêtre qui va assigner chaque point au centre le plus proche. On pourra le colorier avec la couleur du centre.*
Rajouter un bouton qui met à jour les centres selon l'algorithme décrit.

Exercice 4 *Rajouter un bouton qui permet d'exécuter n fois les deux étapes décrites précédemment où n est pris en entrée en utilisant un* `JTextField`*.*