Conception d'un atlas routier

Vous allez dans ce projet écrire une applet permettant de calculer et de visualiser sur une carte un itinéraire entre deux villes de France, calculer des itinéraire optimum en temps ou en consommation d'essence, etc...

Les fichiers nécéssaire à ce projet sont les suivants: **TextFile.class, ville.dat, route.dat (a télécharger sur http://tondeurh.ifrance.com/tp_javalig)**

Etape 1: Chargement des données

Les données nécéssaires à votre application sont stockées dans des fichiers. Dans un premier temps, nous allons nous intérésser au fichier "ville.dat", qui contient les informations concernant les villes elle-mêmes. Le fichier contient une ligne pour chaque ville enregistrée.

Chaque ligne est construite ainsi: il y a quatre champs, le premier est le nom de la ville, le second son code, le troisième et le quatrième ses coordonnées. Les coordonnées sont fournies en km, et se situent donc entre 0 et 1000. Les champs sont séparés par des ";".

Voici un extrait de ce fichier:

```
Valenciennes;2;650;82
Nice;50;958;797
...
```

Vous pouvez éditer directement ce fichier dans Notepad ou WordPad pour voir son contenu.

Pour modéliser l'ensemble de nos villes, vous allez écrire deux classes:

- La classe Ville, qui contient les données relatives à une ville (nom, code, coordonnées)
- La classe Monde, qui contient l'ensemble de toutes les villes. La classe Monde doit posséder un tableau de villes et une méthode:

  public void charger(String fichier), dont le rôle est de remplir le tableau avec les données contenues dans le fichier passé en paramètre (qui sera "ville.dat" dans notre cas).

Pour écrire la méthode "charger", vous avez besoin des outils suivants:

- La classe TextFile, qui vous permet de charger un fichier dans un tableau de Strings. Le fichier se charge en créant un objet TextFile ainsi:

  ```
  TextFile tf = new TextFile("ville.dat");
  ```

  tf contient alors l'ensemble des lignes du fichier "ville.dat". Cet ensemble peut être interrogé par deux méthodes de la classe TextFile: void getSize() qui renvoie le nombre de ligne, et String getLine(int i) qui renvoie la String correspondant à la ième ligne (à partir de 0).
- La classe StringTokenizer, qui va vous permettre d'extraire chacun des champs de la ligne. Vous consulterez la documentation en ligne de Java pour apprendre à vous en servir.
- La fonction Integer.parseInt(String s) qui vous permettra de convertir les Strings obtenues en entiers, si besoin est.

Une fois la classe Monde terminée, vous la testerez en créant un programme de test qui créé un objet Monde, charge le fichier "ville.dat", et affiche pour chacune des villes son nom, son code, ses coordonnées.

Etape 2: Affichage de la carte

Premières expérimentations

Dans un premier temps, vous allez créer une classe TestDessin pour expérimenter le dessin dans les fenètres en Java.

Les dessins se font au sein d'une méthode spéciale appelé automatiquement par Java au moment de l'affichage. Vous devez écrire cette méthode ainsi:

```
public void paint(Graphics g) {
  super.paint(g);
  ...
}
```

A rendre pour le 4 Janvier 2008 par mail à M. Tondeur H => herve.tondeur@wanadoo.fr

Vous devez remplacer les trois points par vos instructions d'affichage, qui seront des méthodes de l'objet graphique g. Par exemple, pour dessiner un rectangle de 10 sur 10 aux coordonnées (100,50), on utilisera la méthode:

g.fillRect(100,50,10,10)

Vous devez également ajouter à votre programme la méthode suivante, pour lui premettre de mettre à jour correctement votre dessin en cas de déplacement ou de redimensionnement de la fenêtre:

```
public void update(Graphics g) {
  super.update(g);
  paint(g);
}
```

Expérimentez différentes méthodes d'affichage (rectangle, ligne...) ainsi que l'affichage de chaines de caractères. Consultez la documentation de la class Graphics pour avoir plus d'informations sur les instructions d'affichage.

La classe Carte

Vous allez écrire une classe Carte héritant de JFrame qui sera chargée de l'affichage de la carte. Comme dans la classe de test précédente, l'affichage se fera au sein de la méthode paint.

Bien sûr, vous allez avoir besoin des données sur les villes pour afficher la carte. Celles-ci vous seront fournies sous forme d'un objet de type Monde qui sera passé au constructeur de Carte et stockés sous forme d'attribut de la classe Carte.

Dans un premier temps, l'affichage de la carte ne consiste qu'en l'affichage de chacune des villes. Un point (ou un rectangle ou un cercle) aux coordonnées de chaque ville, et son nom affiché à coté. Pour rendre la carte plus lisible, vous pouvez utiliser une police de caractère plus réduite, ainsi que des couleurs.

Ecrivez la classe Carte et testez-la. Si votre programme est correct, l'ensemble des villes doit se disposer pour former la silhouette de la France.

Etape n°3: Un premier Atlas

Vous allez maintenant transformer votre carte en une application capable d'indiquer une ville selectionnée par l'utilisateur.

L'application que vous allez créer sera une fenêtre divisée en deux parties. Dans la partie gauche, il y aura la carte, et dans la partie droite, il y aura l'interface utilisateur, comportant un titre, une zone de saisie et un bouton. Quand l'utilisateur rentrera le nom d'une ville et cliquera sur le bouton, alors celle-ci devra s'afficher en rouge.

L'interface graphique

Modifiez la classe Carte pour que celle-ci puisse s'insérer dans votre application. Pour cela, elle ne doit plus hériter de JFrame, mais de JPanel.

Votre carte n'est maintenant plus directement affichable, car ce n'est plus une fenêtre, mais un composant graphique. Vous devez donc supprimer l'instruction show(). Pour spécifier la taille du composant, remplacez setBounds(X,Y) par setPreferredSize(new Dimension(X,Y)). Votre composant graphique Carte est maintenant prêt.

Créez maintenant une classe Atlas héritant de JFrame, qui sera votre application principale. Dans le constructeur de votre application, créez un Container et insérez-y un label pour le titre, une zone de saisie et un bouton. Ce Container sera votre interface. Utilisez le layout suivant pour votre container:

```
BoxLayout interfaceLayout = new BoxLayout(c,BoxLayout.Y_AXIS)
```

Pour que la zone de saisie ne soit pas trop grande, vous pouvez lui donner une taille maximum avec la méthode setMaximumSize(Dimension d). Ajoutez dans le Content Pane de votre application la carte et l'interface.

Testez votre application. Vous devez voir apparaitre la carte sur la gauche et l'interface sur la droite.

Un premier traitement

Ecrivez dans la classe Monde une méthode:

Ville getVilleParNom(String nomVille)

qui renvoie la ville dont le nom est passé en paramètre.

Ajoutez dans la classe Carte un attribut villeSelectionnee qui contient le code de la ville selectionnée, et une méthode

setVilleSelectionnee(Ville v)

qui permet de changer cet attribut.

Modifiez la méthode paint de la classe Carte pour que la ville selectionnée s'affiche en couleur.

Enfin, modifiez la classe Atlas pour que lorsque l'utilisateur clique sur le bouton "Rechercher", la ville qu'il a entré dans la zone de saisie s'affiche en couleur.

Etape n°4: Les routes

Chargement

Le fichier "route.dat" contient les informations concernant les routes entre les villes. Chaque ligne contient trois nombres. Le premier est le code de la ville de départ, le second le code de la ville d'arrivée et le troisième la distance.

Pour chaque ville, vous allez ajouter les informations concernant ses voisines. Pour cela, vous allez ajouter dans la classe Ville trois attributs:

- un attribut int nbVoisines, contenant le nombre de voisines
- un tableau Ville [] voisines, contenant les villes voisines. Il n'y a jamais plus de 10 voisines, donc vous pouvez déclarer à l'avance un tableau de cette taille.
- un tableau int [] distanceVoisines, contenant les distances correspondantes.

Vous allez également ajouter une méthode

public void ajouterVoisine(Ville v,int distance)

permettant d'ajouter une nouvelle voisine.

Dans la classe Monde, vous allez écrire une méthode

public Ville getVilleParCode(int code)

qui renvoie la ville dont le code est passé en paramètre.

Enfin, vous allez ajouter le chargement des routes à la suite du chargement des villes dans la classe Monde. Attention, pour chaque route entre deux villes X et Y, Y doit être ajouté aux voisines de X, et X aux voisines de Y.

Affichage

Modifiez la méthode paint de la classe Carte pour que les routes soit affichées.

Etape n°5: Les itinéraires

Le principe de l'algorithme

Le calcul du plus cours chemin entre deux villes n'est pas un problème très simple à resoudre. Vous allez ici utiliser un algorithme simplifié, qui ne donne pas toujours le meilleur résultat, mais généralement une bonne approximation.

Pour aller d'une ville A vers une ville B, vous allez adopter la demarche suivante:

- On se place en A.
- On se deplace vers la ville voisine la plus proche de B.
- On recommence l'étape précédente jusqu'à se situer en B.

Cet algorithme fonctionne dans la grande majorité des cas, mais pas toujours, comme vous le verrez par la suite.

Mise en oeuvre

Distances

La première chose à faire est d'écrire une méthode de calcul de distance entre deux villes. **Attention!** Le fichier "route.dat" vous fournit les longueurs des routes entre villes voisines. Dans l'algorithme précédent, vous avez besoin de connaître la distance réélle entre deux villes quelconques. Vous allez pour cela faire un simple calcul de distance grâce aux coordonnées géométriques des villes.

Ajoutez à la classe Ville une méthode public int distance(Ville v) qui renvoie la distance à la ville passée en paramètre.

Aide: pour calculer la racine carrée d'un entier x, vous devez utiliser la méthode suivante: (int)(Math.sqrt(x))

Itineraire

Vous allez maintenant écrire une classe Itineraire, qui caractérise un itinéraire entre deux villes.

La classe contient un tableau de villes, qui sont les villes présentes sur l'itinéraire, ainsi que le nombre de ville. Le tableau sera remplit pendant le calcul de l'itinéraire.

La classe, en plus des accesseurs getVille et getNbVille, possède une méthode:

public boolean contains(Ville v)

qui renvoie true si v est sur l'itinéraire, et false sinon.

Enfin, et surtout, elle possède un constructeur

public Itineraire(Ville depart,Ville arrivee)

qui construit l'itinéraire entre les villes depart et arrivee grâce à l'algorithme décrit précédemment.

Affichage

De même que pour villeSelectionnee, vous allez ajouter un attribut itineraireSelectionne et une méthode

public void setItineraireSelectionne(Itineraire i)

à votre classe Carte.

Vous allez modifier la méthode paint de manière à ce que les villes et les routes sur l'itinéraire apparaissent en couleur.

Enfin, vous allez ajouter à votre atlas deux champ de saisie, l'un pour le départ, l'autre pour l'arrivée, et un bouton pour le calcul de l'itinéraire .

Ecrivez une classe Itinéraire qui décrit un itinéraire entre deux villes, sous forme d'une succession de villes.

Le constructeur Itinéraire

Ajoutez un attribut itineraireSelectionne dans la classe Carte et modifiez la méthode paint de manière à ce que l'itinéraire selectionné s'affiche en rouge.

Rectification

Testez votre programme, par exemple de Bordeaux à Lille. Tout doit fonctionner correctement. Testez-le ensuite du Havre à Caen. Que se passe-t-il? Pourquoi? Comment rectifier cela?

Etape n°6: Quelques améliorations...

Voici quelques améliorations que vous pouvez faire à votre programme:

- Affichage de la distance totale de l'itinéraire
- Possibilité de choisir les villes en cliquant dessus

Bibliographie et sources de recherche ...

http://java.sun.com/javase/6/docs/api documentations sur les api Java...
http://tondeurh.ifrance.com/ Documentations sur Java et swing