2009

ECOLE NATIONALE D'INGENIEURS DE BREST

Lepage Sébastien & Hoornaert Nicolas

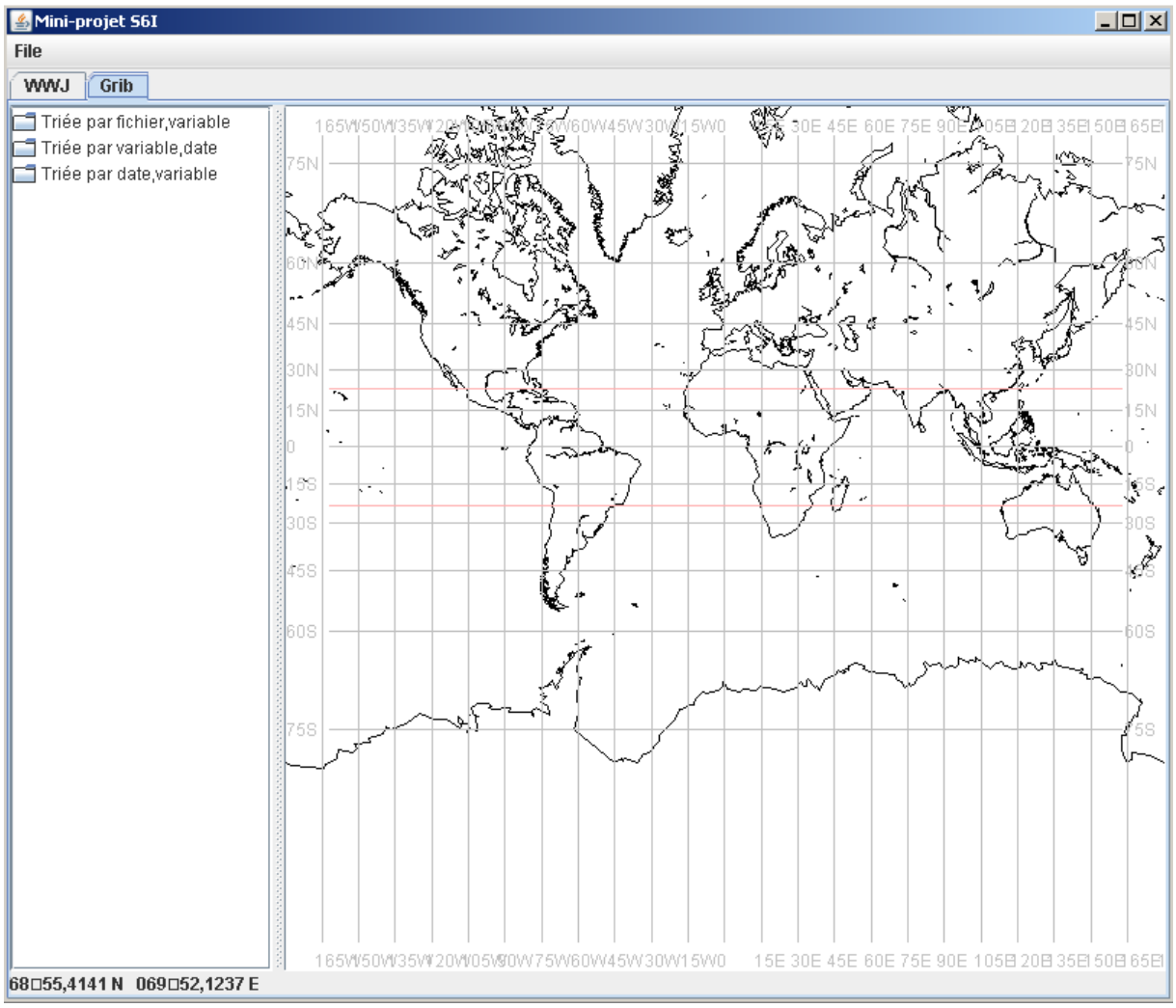
Encadré par Monsieur Serge Morvan

14/06/2009

Table des matières

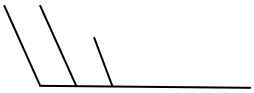
I. Expressions des besoins

Dans le cadre de notre formation, nous avons eu à réaliser un projet sous la tutelle de Monsieur Morvan que nous tenons à remercier. Cette application doit permettre la visualisation de prévision météo, au format Grib en 3D, en utilisant l'API WorldWindJava développée par la Nasa. Cette API est équivalente aux projets Virtual Earth de Microsoft ou GoogleEarth de Google, à la différence de la licence complètement open source. Les données Grib seront prises sur le site UGrib. Un programme de décryptage des données Grib est fourni: ViperFish. L'application combinera Viperfish et WorldWindJava, l'affichage dans l'un permettra l'affichage dans l'autre.



1. Les Barbules

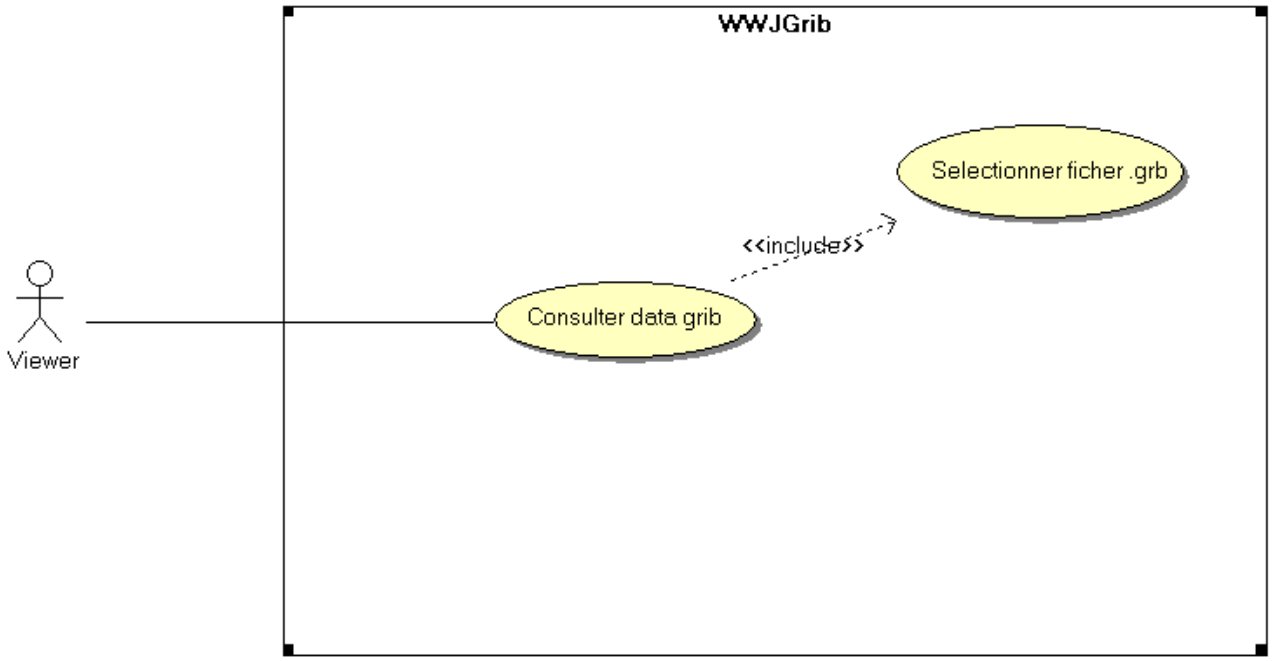
Sur une carte météorologique, les météorologues utilisent un symbole pour représenter à la fois la vitesse et la direction du vent. Ce symbole est la barbule.

Ouest  Est

La tête de la barbule pointe dans la direction d'où vient le vent. Sur l'image, le vent souffle donc de l'ouest vers l'est. C'est un vent d'ouest. La vitesse du vent est donnée par le nombre de barres attachés à la barbule :

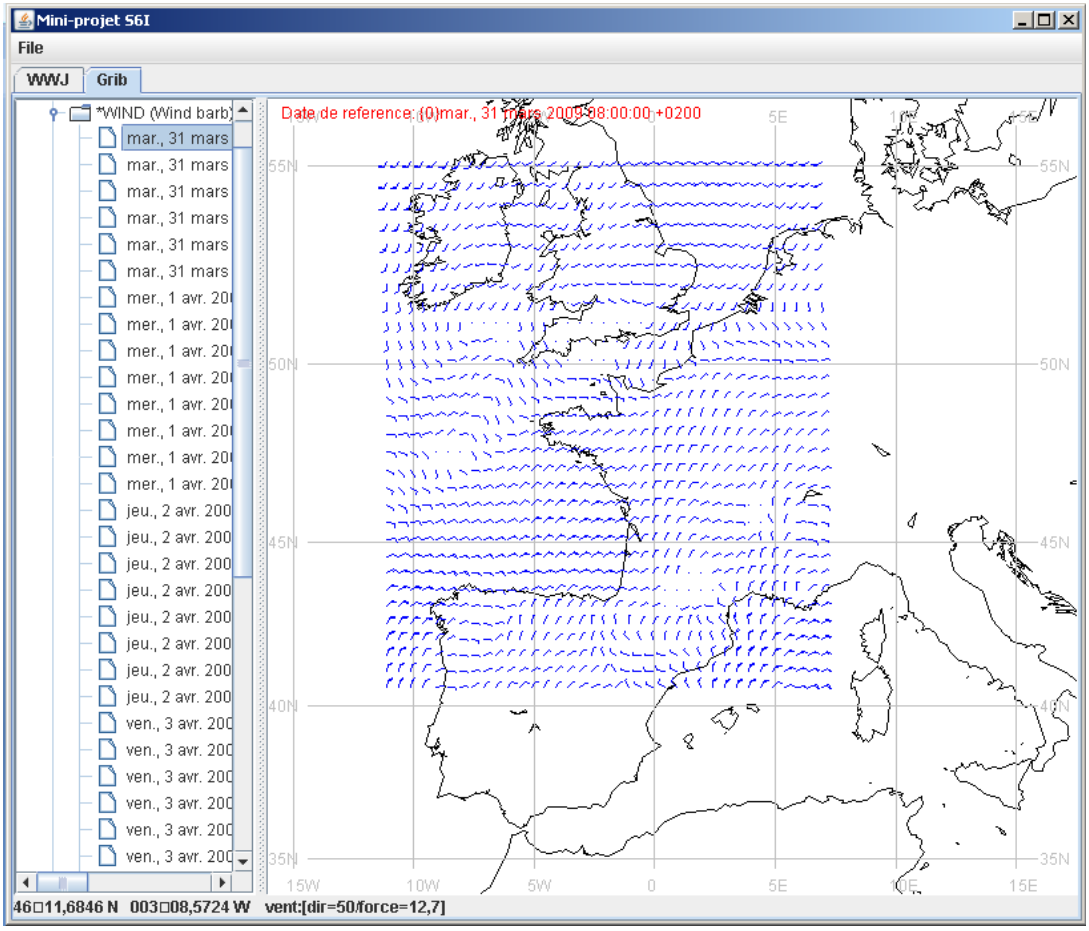
- 1 longue barre = 10 nœuds
- 1 petite barre = 5 nœuds

2. Lotissement du projet

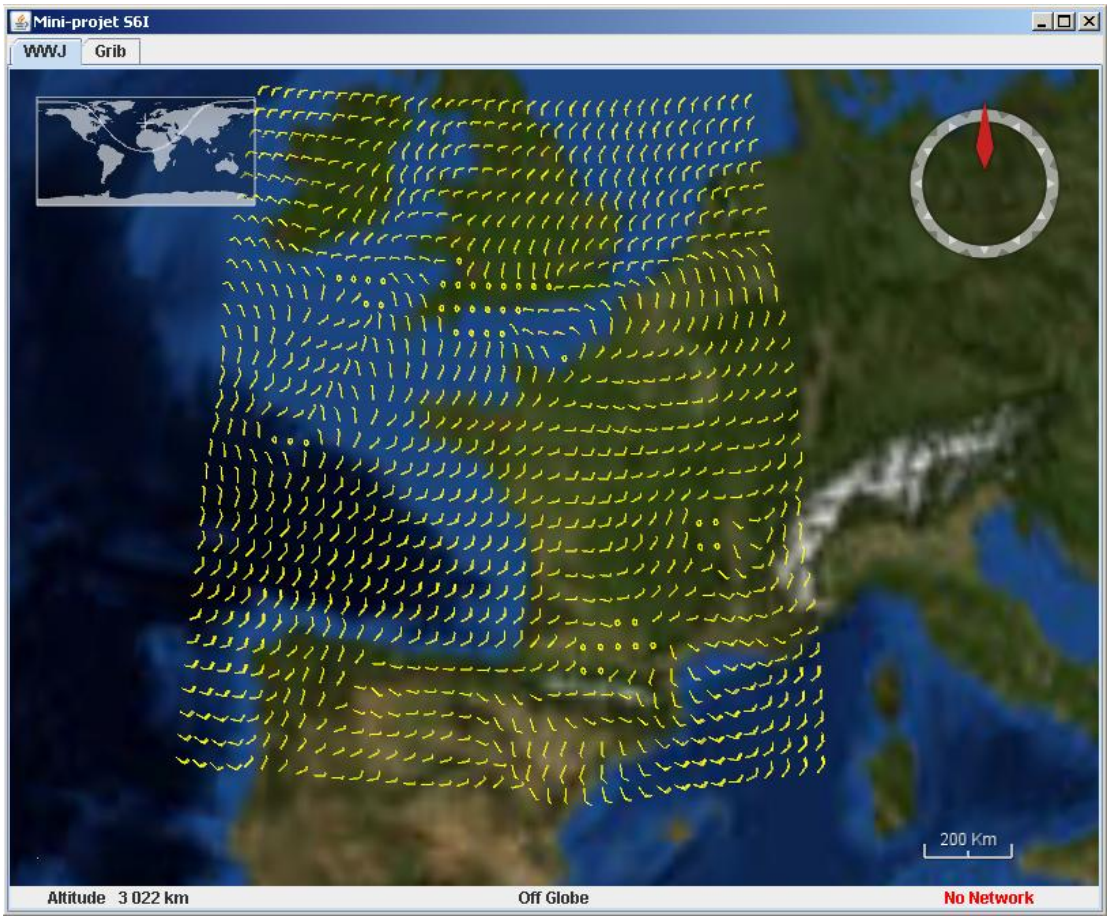


Les fichiers peuvent être téléchargés à partir du site : http://grib.us ou http://meteo-marine.com/grib.htm. Suite aux téléchargements, il est possible via Viperfish d'afficher les données correspondantes (File>Open):

Sous ViperFish :

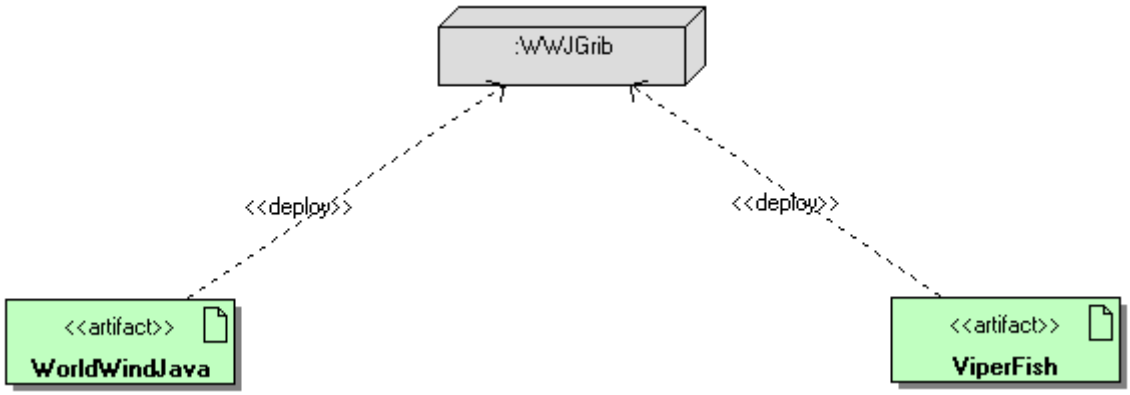


Sous World Wind Java :



II. Analyse du sujet

1. Architecture de l'application



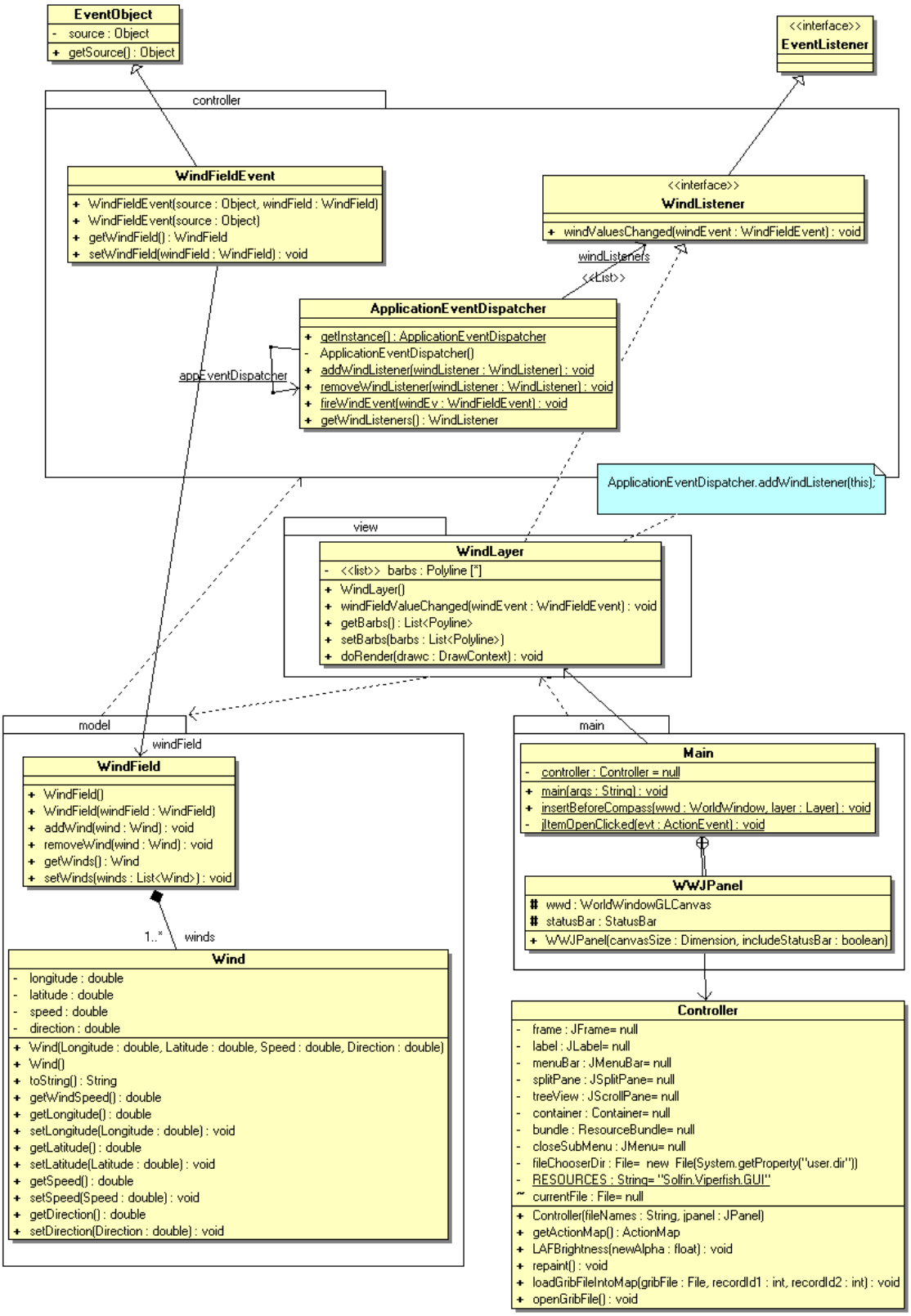
2. Diagramme de classes

L'application a été développée selon le modèle MVC *(Model View Controller)* :

Packages	Classes
Package Model :	-Wind (Représente un vent)
	-WindField (Représente un champ de vent)
Package Controller :	-WindFieldEvent (Evènement déclenché lorsqu'un windField est instancié.
	-WindListener (Interface a implémenté afin d'être à l'écoute d'un vent).
	-ApplicationEventDispatcher (Singleton/Emetteur de WindFieldEvent).
Package View:	-WindLayer (Représente une couche sur laquelle sera inscrite ou plusieurs champs de vent).

Le développement de cette application a contribué à la réutilisation des packages développés par la NASA ainsi que ceux de Viperfish. Néanmoins l'analyse de cette application ne portera pas sur l'intégralité des ressources utilisées étant donnée l'abondance de classes…Elle portera essentiellement sur une partie qui a fait l'objet de programmation évènementielle.

Voici donc le diagramme de classes de la partie « programmation évènementielle » :



3. Explication des structures de données utilisées

Comme le démontre le diagramme utilisé ci-dessus, l'application nous a obligés à mettre en application de la programmation évènementielle. La raison de cette utilisation vient du fait, que le choix du fichier grib à afficher se fait à l'exécution et non à la compilation. En suivant le modèle de Viperfish, une fois le fichier sélectionné, celui-ci est affiché. Le cahier des charges souhaite que ces données soient en plus affichées dans WorldWindJava.

1. Le patron Observer

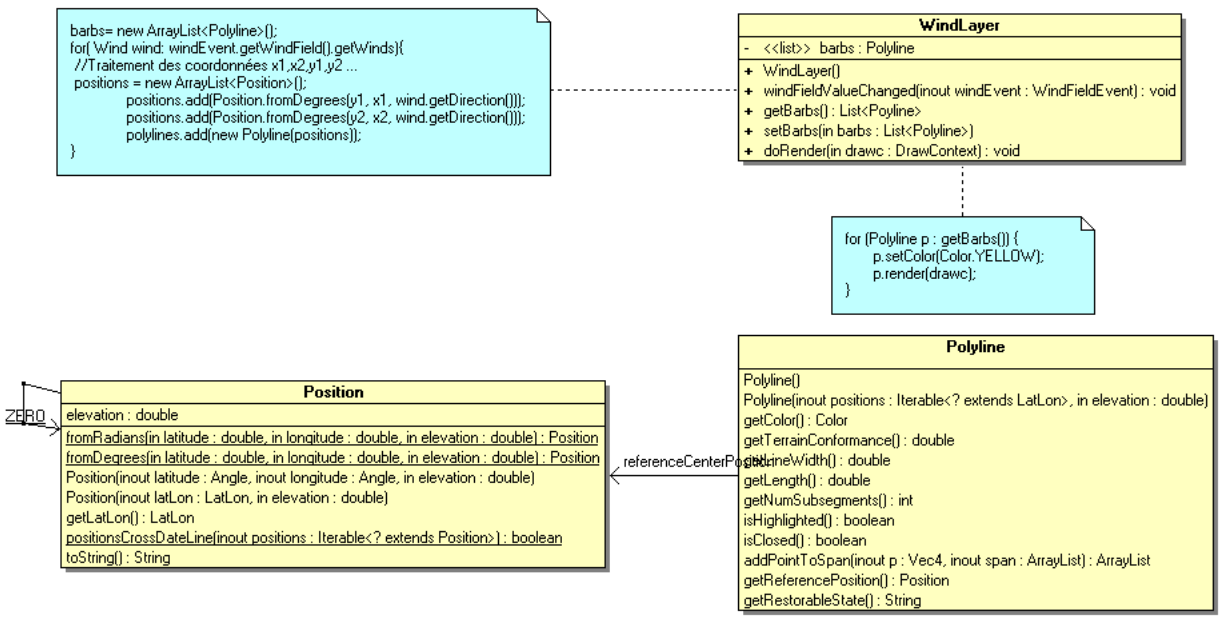
L'analyse du patron s'appuie sur le modèle classique de la programmation évènementielle, c'est pourquoi nous nous appuierons sur les points essentiels de ce patron. Les classes étudiées seront l'émetteur de l'évènement *(ApplicationEventDispatcher)* et le souscripteur de l'évènement *(WindLayer)* ainsi que sa fonction de traitement des vents : « WindFieldValueChanged() ».

1.1 Emetteur du WindFieldEvent

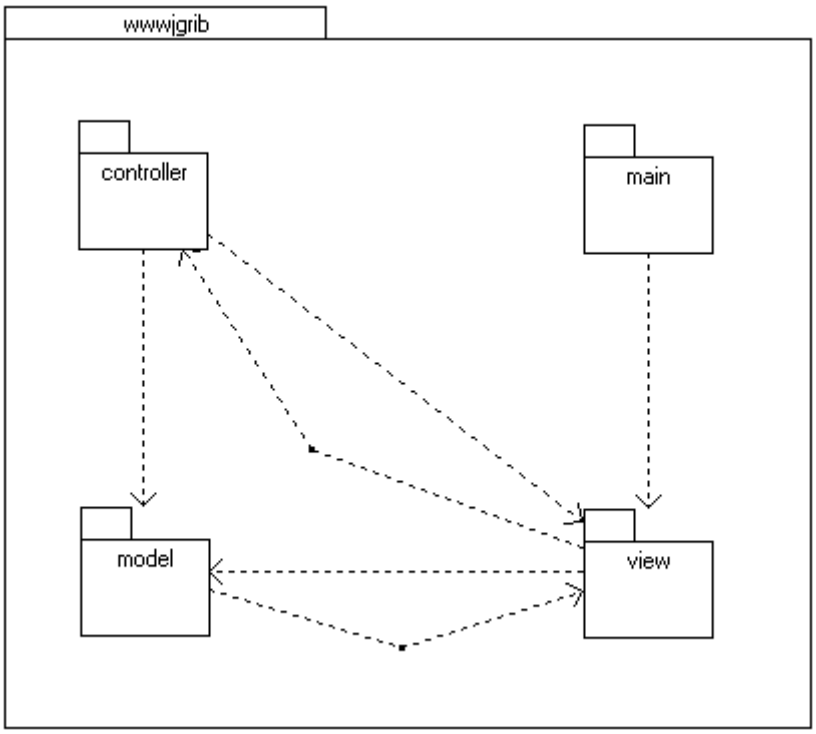
L'objet ApplicationEventDispatcher est un singleton afin de s'assurer qu'une seule instance de l'objet sera instanciée pendant toute la durée de l'application, s'assurer que c'est la même instance à laquelle on se réfère quelque soit le temps écoulé entre deux appels. Celui-ci permet d'émettre un objet WindField une fois tous les vents dessinés.

1.2 Récepteur du WindFieldEvent

L'objet WindLayer est une couche qui traitera le champ de vent reçu, c'est-à-dire une fois la méthode WindFieldValueChanged appelée. Ce traitement consiste à transformer les coordonnées cartésiennes des vents de ViperFish en coordonnées mercatoriennes pour WorldWindJava. Il consiste aussi à construite la barbule du vent correspondant; En objet, elle est une liste de polyline : une succession de lignes. Une polyline pour chaque branche de la barbule. Une polyline se construit à partir d'une liste de positions :



4. Packages en conception



III. Conclusion

Ce mini-projet a été très enrichissant du fait qu'il nous a non seulement permis d'approfondir et d'appliquer nos connaissances en programmation Java, mais aussi à respecter les conditions d'un cahier des charges. Ce mini-projet a été d'autant plus intéressant qu'il s'agissait de réaliser un système concret utilisé par de nombreux secteurs de recherches tel que la NASA. Cependant, les sources mis à disposition par les développeurs de ViperFish restent bâclée du fait que la javadoc n'a pas été créée, ce qui a donc ralenti notre progression au sein du projet et nous a causés des difficultés de compréhension. Néanmoins, ce projet a permis de nous rendre compte de la difficulté à développer une application de ce type, et donc à établir une analyse rigoureuse et soigneuse. Une étude sur un projet de ce type a donc permis de rassembler les connaissances de programmation objet et évènementielle dans le langage Java, et celles de modélisation pour l'ingénieur système, qui nous a été très utilise pour l'analyse du sujet.

IV. Bibliographie

Site de NASA WorldWind Java : http://worldwind.arc.nasa.gov/java/

Open Source ViperFish : http://french.osstrans.net/software/viperfish.html

Analyse WWJGrib : WWJGribModeling

JavaDoc WWJGbrib : WWJGribJavadoc

Liens vers les fichiers grib: http://grib.us
http://meteo-marine.com/grib.htm