Algorithmique et structures de données

TD / TP 1

C. Sempé

Les objectifs de ces premières séances de TD / TP sont :

- rappels sur la syntaxe JAVA
- reprendre en main l'environnement NetBeans
- refaire les algorithmes traditionnels sur les tableaux
- structurer les données au sein d'un objet.

Les smartphones et objets connectés peuvent générer des données que l'on peut ensuite manipuler et interpréter.

C'est le cas lorsqu'on fait de la course à pied et que l'on veut tracer son parcours pour évaluer les distances, vitesse, rythme cardiaque...

Des applications comme RunKeeper utilise le GPS du téléphone pour tracer et analyser votre course.

Nous avons récupérer les données GPS d'une course et nous allons dans ce TD/TP les stocker, manipuler, interpréter et tracer

Les données

L'application nous fournit la latitude et la longitude en degrés ainsi que l'altitude en mètres
Chaque relevé de point se fait dans un intervalle de temps exprimé en secondes

lat	lon	temps	alt
46,056425	-1,083614	0	12
46,056455	-1,083547	5	12
46,056377	-1,083457	4	11,9
46,056393	-1,083334	4	11,8
46,056356	-1,083223	5	11,7

Ici entre le premier point et le second il s'est écoulé 5 secondes

TD1

Comment stocker les données sur un ordinateur ?
Plusieurs possibilités. Quel choix de structure de données ?

1 - Ecrire en Java une classe Point qui stockera les informations relatives à un point.

La classe possédera un constructeur avec tous les paramètres nécessaires à la construction d'un point.
La classe possédera aussi tous les assesseurs de type get afin de pouvoir récupérer les données lorsqu'on le souhaite.
La classe aura une méthode toString() qui permettra de retourner la chaîne de caractère représentant le mieux un Point.

2 - Testez la classe Point en créant 2 points et utilisant un maximum les méthodes écrites.

3 - Stocker des Points

Ecrire la classe Parcours qui stockera l'ensemble des points d'une course.
Comment stocker l'ensemble des points fournis par l'application ? Faites la déclaration de la structure la plus adaptée.
Le constructeur construira la structure de données vide.
Une méthode chargement() permettra de remplir la structure par des points à partir d'un fichier.(conception en TP)

4 - Algorithmes

Nous allons ajouter à la classe Parcours les outils nécessaires pour bien exploiter les données.

- Ecrire la méthode afficher() qui affiche directement à l'écran tous les points.
- Ecrire la méthode altitudeMax() qui retourne l'altitude maximum atteinte sur le parcours.
- Ecrire la méthode temps() qui retourne le temps total en seconde du parcours.

5 - Ecrire la méthode distance(P) dans la classe Point, capable de retourner la distance en km entre deux points de coordonnées GPS.

Comment accéder aux données des deux points ?

Calcul à effectuer en utilisant Math :
Distance = R ArcCos(Sin(a) Sin(b) + Cos(a) Cos(b) Cos(c-d))

R est le rayon de la terre soit 6347 km
a est la latitude du point A en radian
b est la latitude du point B en radian
c la longitude du point A en radian
d la longitude du point B en radian
Attention : la latitude et la longitude fournie par l'application sont en degrés

TP1

Mode DEBUG (utilisation obligatoire)
L'enseignant ne sera pas toujours là pour vous corriger les erreurs. Vous DEVEZ savoir utiliser le debugeur OBLIGATOIREMENT. Si ce n'est pas le cas, demandez à votre enseignant de vous montrer, en préparant au préalable l'appel unique dans le main() à une méthode comportant une boucle.

Complétez et testez les classes et outils conçus en TD.

- Récupérez le projet sur Moodle. Ouvrir le projet avec NetBeans, observez le programme principal main() ainsi que les autres classes.
  Repérez les différentes classes vues en TD ainsi que leur contenu. Certaines méthodes sont déjà présentes.
- Testez dans le main la distance entre deux points que vous construirez.
- Complétez la classe Parcours puis testez le chargement des données et les méthodes du TD.

Nouvelles méthodes

- Ecrire la méthode *afficherVitesses()* qui affiche l'évolution de la vitesse sur le parcours
- Ecrire la méthode *vitesseMoy()* qui retourne la vitesse moyenne sur tout le parcours.
- Ecrire la méthode *split(intervalleDist)* qui retourne un tableau de Double.
  On calcule la vitesse tous les intervalleDist. Par exemple si intervalleDist=1 alors on placera dans le tableau le calcul des vitesses tous les 1 km.

Il vous faudra cumuler les distances et le temps puis dès que l'on atteint 1 km alors on calcule la vitesse moyenne sur l'intervalle, on la stocke puis on continu avec des compteurs remis à zéro.

- Ecrire la méthode afficherLesMax*()* qui affiche les points sur lesquels on a détecté un pic. Un pic sera défini ainsi :
  Ecart(pi , pi+1) > 0 ET Ecart(pi+1 , pi+2) < 0

Graphiques

On vous fourni une fenêtre graphique dans laquelle on peut tracer des points sur un plan orthonormé.

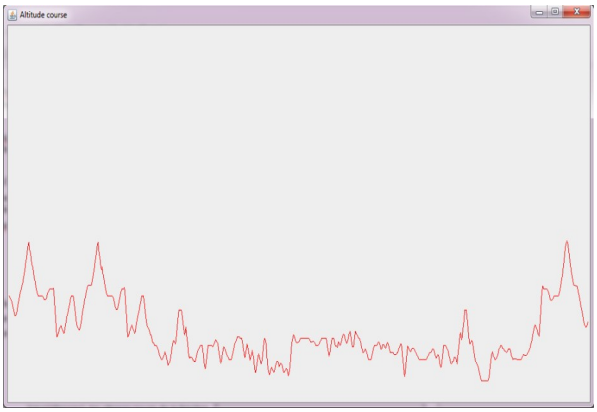
```
Fenetre maFenetre = new Fenetre("Altitude course", 1500, 500);
```
On peut créer une fenêtre de dimension 1500 pixels en largeur par 500 pixels en hauteur.

```
maFenetre.tracerPoint( 10, 30 ) ;
```
On trace ensuite un point en x=10 y=30

ATTENTION, l'origine (0, 0) est en haut à gauche de la fenêtre

remarque : si vous voulez transformer des réels en entier pour la fenêtre

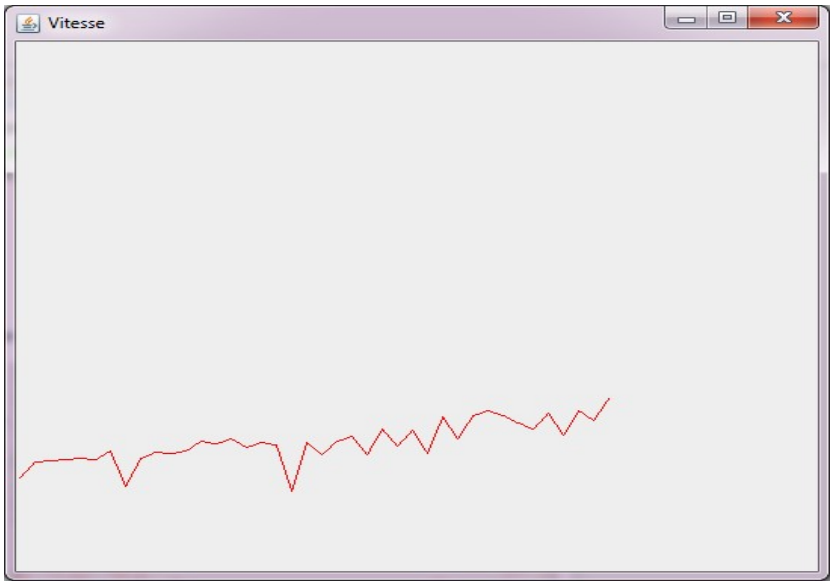
```
int y = (int) (vitesse)
```



- Tracez grâce à la méthode tracerAltitude() la courbe de l'altitude du parcours.

Faites en sorte que la courbe soit lisible, à l'endroit, dans des proportions correctes. Vous pouvez aussi vous contenter d'afficher 1 point sur 2.

- Tracez dans une autre fenêtre grâce à la méthode tracerVitesses(), les vitesses issues de la fonction split(0.5).



- Placez les points GPS dans l'application Google myMaps en important le fichier PointGPS.cvs afin de voir le vrai trajet effectué.