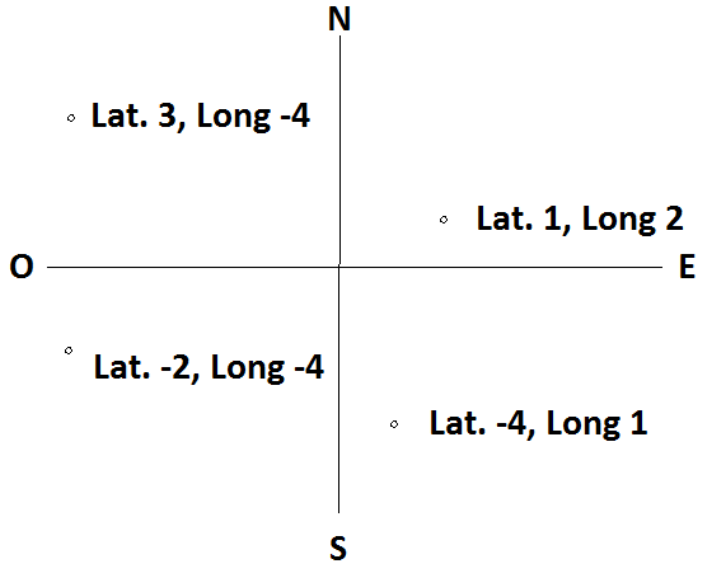
Projet d'Algorithmique : « Parapente »

1. Description du contexte

Tommy est un passionné de parapente. Chaque fois qu'il effectue un vol, il emporte un gps. Cet appareil enregistre à intervalle de temps régulier sa position. Pendant toute la durée de son vol, son parcours est ainsi mémorisé sous forme d'une suite de coordonnées gps.
Une fois à la maison, Tommy transfère toutes les données du gps dans un fichier sur son ordinateur. Chaque enregistrement de ce fichier est donc constitué des coordonnées gps du lieu survolé à un moment donné. Par exemple : 7°23'45'' de latitude N et 51°12'23'' de longitude E.

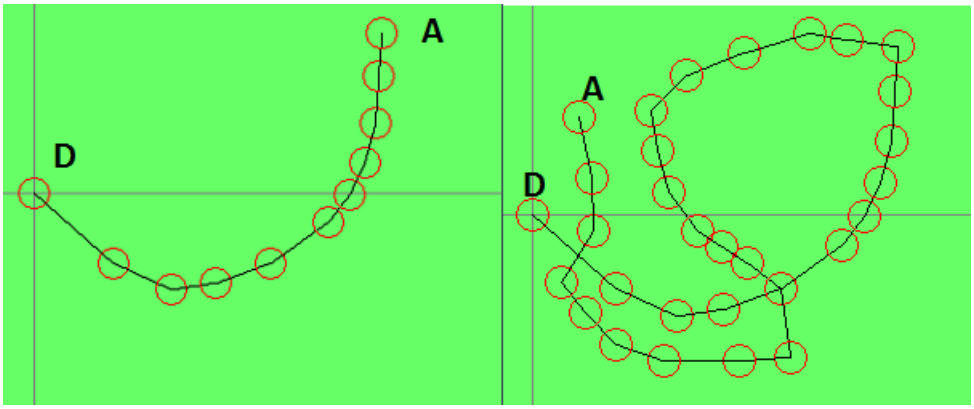
Exemples de coordonnées gps (système simplifié)



Exemples de vols

Dans les schémas suivants, chaque point représente un lieu qui a été survolé et mémorisé par le gps. Le point D représente le point de départ. Le point A, le point d'arrivée.

Vol 1 : Vol 2 :

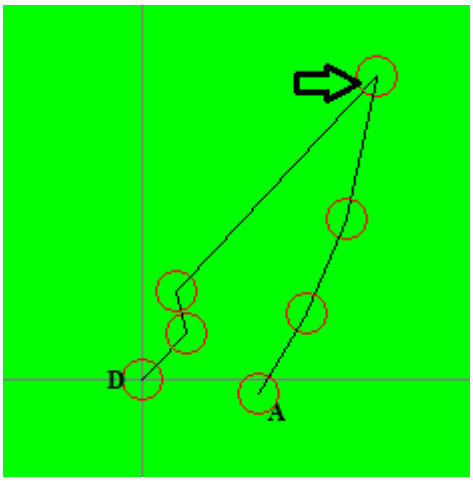


Tommy participe à de nombreux championnats. Ceux-ci proposent différentes épreuves.
Il peut s'agir d'aller le plus loin possible, de parcourir la plus longue distance ou même d'atteindre des cibles.

Tommy aimerait obtenir quelques statistiques sur son vol :

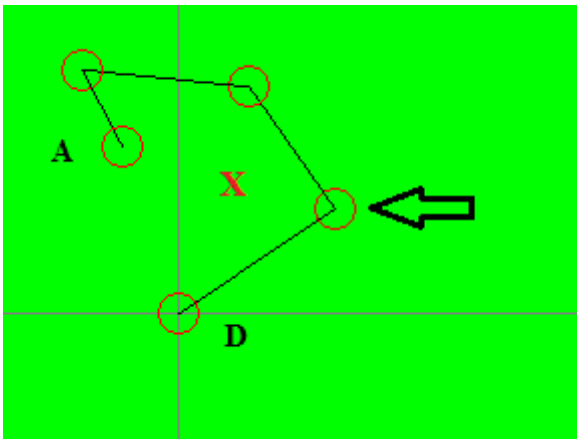
1) La durée du vol
Une unité de temps correspond au temps écoulé entre 2 mesures de position du gps.

2) Le **lieu** enregistré **le plus éloigné** du lieu de départ à vol d'oiseau
Ce lieu ne doit pas nécessairement correspondre au lieu d'arrivée.
En cas d'ex-aequos, c'est le premier lieu qui sera donné.



3) Les 4 **lieux extrêmes,**
Il s'agit du lieu enregistré le **plus au nord**, celui le **plus à l'est**, celui le **plus au sud** et celui le **plus à l'ouest** du parcours.

4) Le **lieu** enregistré le **plus proche** d'une **cible** à atteindre.
En cas d'ex-aequos, c'est le premier lieu qui sera donné.

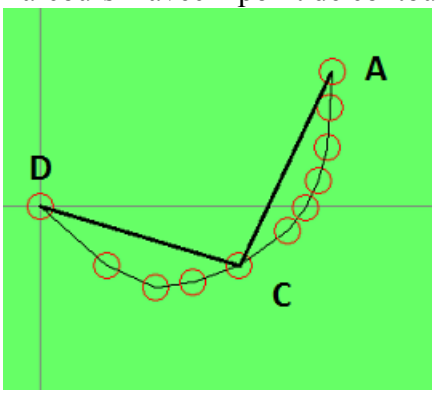


5) La **distance** parcourue.
Cette distance sera obtenue en additionnant les distances des segments du vol mémorisé.

6) La **distance maximale** parcourue en tenant compte d'un nombre donné de points de contournements.
Elle correspond à la distance entre le point de départ et celui d'arrivée en passant par quelques points du parcours.
Ces points de contournement ne sont pas fixés avant le vol. Ils sont déterminés par le logiciel afin de maximiser la distance.

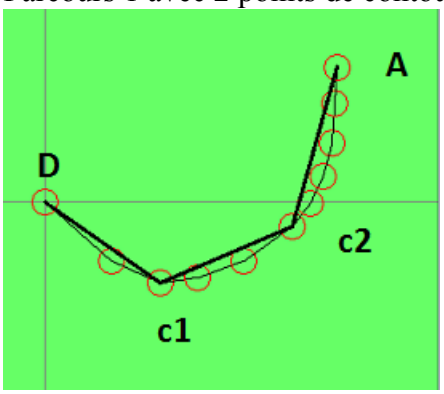
Supposons 1 point de contournement C dans le parcours 1. C'est un point de contournement qui a été choisi par la méthode afin de maximiser la distance. Celle-ci est obtenue en additionnant les longueurs des segments de droites DC et CA.

Parcours 1 avec 1 point de contournement :



Supposons 2 points de contournement dans le parcours 1. La distance calculée sera obtenue en additionnant les longueurs des segments de droites DC_1, C_1C_2 et C_2A. C_1 et C_2 étant des points de contournement qui ont été choisis par la méthode afin de maximiser cette distance.

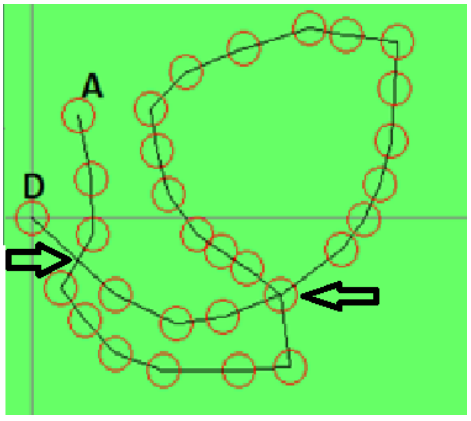
Parcours 1 avec 2 points de contournement :



Le nombre de points de contournement est de 1 ou 2.
Pour ceux qui le souhaite, mais ce n'est pas obligatoire et ce ne sera pas coté, vous pouvez prévoir jusqu'à 3 points de contournement (ou même k points de contournements !)

Le document *ptsContournement* essaye de montrer l'intérêt de cette statistique à partir d'1 exemple.

7) Le **nombre** de **croisements**.
Il s'agit du nombre de fois qu'il a croisé sa route.
Si le lieu de départ est le même que celui d'arrivée, il faut le compter.
Il y a croisement lorsque 2 lieux enregistrés sont les mêmes, mais aussi lorsque 2 segments s'entrecroisent.



2 segments se sont croisés

2 lieux enregistrés sont les mêmes

8) Les **cibles atteintes** parmi une liste donnée de cibles à atteindre.
Dans la liste donnée, il n'y a pas de doublon.
L'ordre dans lequel il faut atteindre les cibles n'a pas d'importance.
L'ordre dans lequel les cibles atteintes vont être affichées n'a pas d'importance.
Si une cible a été atteinte plusieurs fois, il ne faut la donner qu'une fois.

9) Le **nombre** de **cibles atteintes** lors d'un **parcours** imposé.
L'ordre dans lequel il faut atteindre les cibles a de l'importance.
Une cible peut se retrouver plusieurs fois dans e parcours.
Si une cible est survolée alors qu'une des précédentes ne l'a pas été, elle n'est pas comptée.
Elle devra être à nouveau survolée.

10) Une statistique au choix proposée par votre groupe

2. Travail à réaliser

2.1. Classe *TraitementVol*

Nous vous demandons de compléter la classe *TraitementVol*.
Il ne s'agit pas d'une classe de tests. Ce programme s'adresse directement à Tommy.
Ce programme commence par charger un fichier contenant les données du vol à analyser.
Ensuite via menu, il proposera d'afficher une des 10 statistiques demandées.
C'est dans cette classe qu'on va trouver les lectures clavier et les affichages.
Complétez méthode qui permet de remplir une table avec des coordonnées gps ainsi qu'une méthode qui permet d'afficher une telle table.
Pensez à utiliser les méthodes de votre classe Utilitaires !

2.2. Classe *Vol*

Nous vous demandons de compléter la classe *Vol*.
Cette classe contient un tableau de coordonnées gps.
Vous ajouterez toutes les méthodes qui vont permettre de récolter les statistiques demandées.
Les lectures au clavier et les affichages seront effectués uniquement dans le programme principal (*TraitementVol*) et pas dans cette classe.
Lorsque plusieurs coordonnées doivent être fournies ou renvoyées, cela se fera via une table (PAS d'ArrayList ou classes similaires)!
Toute table renvoyée aura toujours la taille physique égale à la taille logique.
Pensez à ajouter des méthodes (*private*) du style : *contient(Coordonnees c)* ou *donnerIndice(Coordonnees c)* ou même d*onnerIndice(Coordonnees c, int indiceDebut, int indiceFin)*

2.3. Remarques concernant l'implémentation

- Nous vous fournissons un programme de création de fichiers de données : *PgmCreationFichiersParcours*. Il vous permettra de créer des fichiers afin de tester vos classes.

- Ne modifiez pas la classe *Coordonnees*.
Cette classe contient les méthodes *distance(), equals() et segmentsCroises()*.

- N'introduisez pas d'autres classes. Chaque méthode « *public* » de la classe *Vol* ne renvoie qu'une seule information : un type simple (*int*, *double*, …), une coordonnée ou un tableau de coordonnées.

2.4. Tests

On ne vous demande pas d'écrire une classe de tests pour les différentes méthodes de la classe *Vol* !
Par contre veillez à bien tester chaque épreuve en exécutant la classe *TraitementVol* avec des vols préparés pour vérifier différents cas.
Dans le dossier, vous allez numéroter les vols et les visualiser. (S'ils sont bien conçus, 2-3 devraient suffire.)

Ensuite vous allez passer en revue chaque méthode qui ne renvoie pas une distance.
Pour chaque méthode, vous allez répertorier les différents cas à tester.
Vous allez remplir un tableau reprenant pour chaque cas :

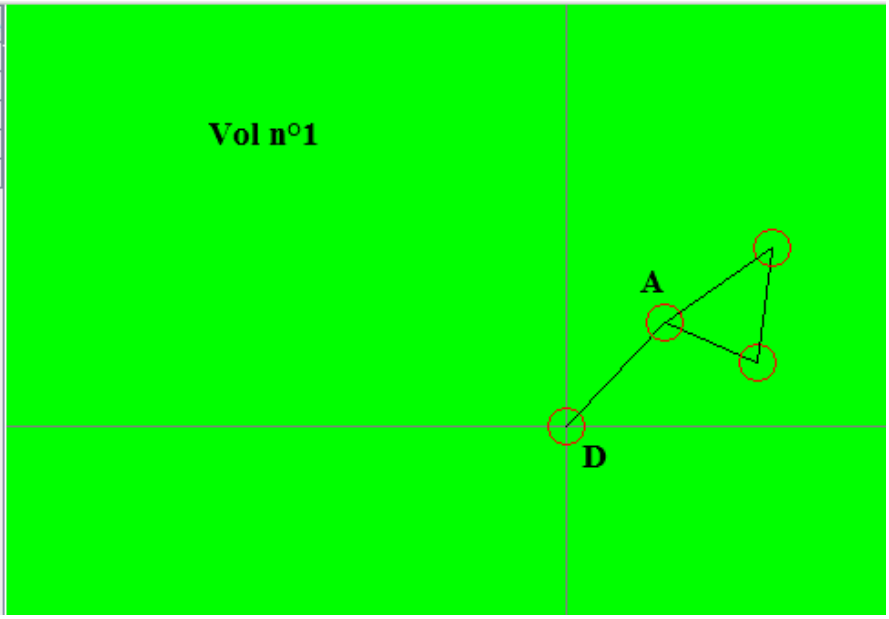
- Un descriptif
- Le n° du vol
- Les « IN » : données introduites par Tommy
- Les « OUT » : résultats attendus

Pour bien faire, ces tableaux devraient être remplis avant de programmer !!!

Voici 1 exemple (limité à 1 vol et 1 méthode) de ce que nous attendons dans votre dossier :

Vol n°1

N°	Latitude	Longitude
1	0	0
2	57	54
3	35	105
4	98	113
5	57	54



Méthode *ciblesAtteintes()*

Cas n°	Explication	Numéro du vol	IN	OUT
1	Aucune cible atteinte	Vol n°1	cibles à atteindre 77 77 99 99	cibles atteintes (table vide)
2	Quelques cibles atteintes pas nécessairement dans l'ordre	Vol n°1	cibles à atteindre 98 113 77 77 35 105 99 99	cibles atteintes 98 113 35 105
3	Quelques cibles atteintes pas nécessairement dans l'ordre Une des cibles a été atteinte plusieurs x	Vol n°1	cibles à atteindre 57 54 77 77 35 105 99 99	cibles atteintes 57 54 35 105
4	Toutes les cibles sont atteintes	Vol n°1	Cibles à atteindre 98 113 57 54 77 77 35 105 99 99	cibles atteintes 57 54 98 113 35 105

Pour tester les 2 méthodes qui renvoient des distances vous pouvez utiliser le vol 77 fourni.
Méthode *distance()*

Cas n°	Explication	Numéro du vol	IN	OUT
1		Vol n°77		9.668875159187094

Méthode *distanceMaximale(int nombrePointsContournement)*

Cas n°	Explication	Numéro du vol	IN	OUT
1		Vol n°77	1	8.937266254458097
2		Vol n°77	2	9.448117432955836
3	Vol avec 2 coordonnées		1	IllegalArgumentException
4	A compléter			

3. Dossier à remettre

Les séances d'exercices en algorithmique des semaines 11 et 12 seront consacrées à ce projet.
La **présence** est **obligatoire**.
Le travail est à réaliser par **groupe de deux étudiants**. Il prend fin le lundi **19 décembre à 16h**. On demande une impression « papier » d'un dossier et une soumission.

Le dossier papier est à remettre dans le casier portant le numéro de votre série**.** Les casiers sont situés près de la machine à café.

Vous devez remettre un dossier contenant :

- les noms, prénoms et le numéro de série des 2 étudiants sur la première page de garde

- une table des matières

- une introduction

Cette introduction reprend le sujet du projet et présente le contenu du dossier. Elle s'adresse à quelqu'un qui n'a aucune idée du projet. Celui-ci doit pouvoir se faire une bonne idée de celui-ci sans devoir aller consulter les sources.
N'oubliez pas de décrire clairement (avec schémas) la statistique que vous avez ajoutée !

- le listing (impression papier!) des classes *TraitementVol* et *Vol*

Vous veillerez à réaliser une mise en page correcte facilitant la lecture.
La *Javadoc* de la classe Vol.

- les tests

- une conclusion

  Dans cette conclusion, vous donnerez l'état d'avancement : tout fonctionne ?
  Si ce n'est pas le cas, dites clairement ce qui fonctionne et ce qui ne fonctionne pas !
  Dans la conclusion, vous pourrez aussi indiquer les difficultés rencontrées, ce que ce dossier vous a apporté, l'apport du travail en groupe et ses difficultés, les améliorations qui pourraient être apportées au programme,…

Sur l'extranet, vous devrez avoir posté, **au plus tard pour le lundi 19 16h** dans **un** fichier compressé (.zip) **portant vos deux noms** : les classes *Utilitaires, TraitementVol* et *Vol,* vos vols et une version électronique de votre dossier.
La version électronique du dossier **doit** correspondre à la version papier !

4. Evaluation

Voici à titre indicatif les critères d'évaluation des méthodes qui seront corrigées :

Code fonctionnel

F1 Le code est correct.

F2 Les paramètres sont testés (*IllegalArgumentException* en cas de paramètres non valides)

F3 Le code tient compte des cas rares mais cependant possibles.
(Vous pouvez supposer que la table du vol contient toujours au moins 2 coordonnées.)

F4 Il n'y a pas d'absurdité en terme de performance.
(On n'exige pas pour autant la version la plus performante.)

F5 Le code n'est pas inutilement compliqué.

Code clair et documenté

C1 Le code est auto-commenté

C2 Une découpe en sous-méthodes a été faite de façon à clarifier le code

C3 La JavaDoc est présente et intéressante

C4 Les passages difficiles à comprendre sont commentés de façon pertinente

C5 La statistique proposée est bien expliquée dans l'introduction

Tests fournis

T1 Les cas de tests sont bien explicités

T2 Les cas sont pertinents (pas trop, pas trop peu)

Voici une liste contenant certaines causes de sanction.
Selon la gravité, le projet pourrait se voir attribuer une note équivalente à 0 ou ne pas être corrigé !

S1 Le plagiat

S2 Le travail a été fait seul sans la permission préalable d'un professeur.

S3 L'étudiant n'a pas été présent à toutes les séances d'exercices consacrées au projet.

S4 L'étudiant ne s'est pas montré actif lors de ces séances.

S5 La soumission n'a pas été faite dans le délai imparti.

S6 Le dossier a été remis en retard.

S7Le dossier ne contient pas tout ce qui a été demandé.

S8 L'orthographe et/ou la langue française sont reprochables.

S9 L'impression est peu lisible (encre, …).

S10 Le format d'écriture est mal choisi (trop petit ou trop grand).

S11 Les pages ne sont pas numérotées.

S12 La table des matières est peu fournie. On ne trouve pas aisément le code ou le test d'une statistique.

S13 La présentation des sources ne facilite pas la lecture.

Par contre, 1 point bonus est prévu pour les dossiers particulièrement bien présentés !