LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Ministère de l'Éducation nationale,
de l'Enfance et de la Jeunesse

EXAMEN DE FIN D'ÉTUDES SECONDAIRES CLASSIQUES
2020

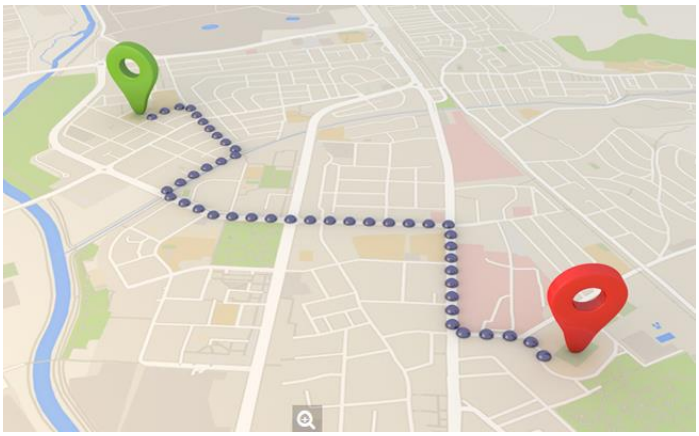
BRANCHE	SECTION(S)	ÉPREUVE ÉCRITE
SCIPR	**CI /CLI**	*Durée de l'épreuve :* *180 minutes* *Date de l'épreuve :* 21/09/2020

Partie modélisation

10 points

- La partie modélisation doit être rendue avant que la partie implémentation puisse être débutée. Cette partie est à réaliser sur papier.
- La partie modélisation est à rendre après 35 minutes au plus tard, mais vous être libre de la rendre plus tôt.

Afin de pouvoir déterminer la distance parcourue lors d'une randonnée, beaucoup de gens emmènent leur traceur GPS lors d'une excursion. Après chaque randonnée la trace GPS ainsi générée est transférée et sauvegardée sur ordinateur.



Un traceur GPS enregistre à des intervalles réguliers sa position. Une trace GPS est donc un ensemble ordonné de points dans l'espace. La distance totale d'une trace GPS correspond à la somme des distances d'un point au prochain, du premier au dernier point de la trace.

Vous devez écrire une application permettant de réaliser une évaluation d'une trace GPS. Pour débuter, vous devez mettre au point un programme qui permet de calculer la distance totale d'une trace GPS.

1. Développez donc le diagramme de classe UML pour un modèle permettant les choses suivantes (vous n'avez pas besoin d'y mettre les accesseurs et manipulateurs) :

   - Lecture d'une trace GPS (EN : track) à partir d'un fichier (le format ne joue pas de rôle pour l'instant…).
   - Un point se compose d'une coordonnée 2D. Il doit aussi être possible de sélectionner un point afin de le mettre en évidence.
   - Calcul de la distance d'un point à un autre.
   - Calcul de la distance totale d'une trace GPS.

2. Expliquez en détail votre choix pour la structure de donnée que vous avez choisi pour les différents points de la trace GPS !

LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Ministère de l'Éducation nationale,
de l'Enfance et de la Jeunesse

EXAMEN DE FIN D'ÉTUDES SECONDAIRES CLASSIQUES
2020

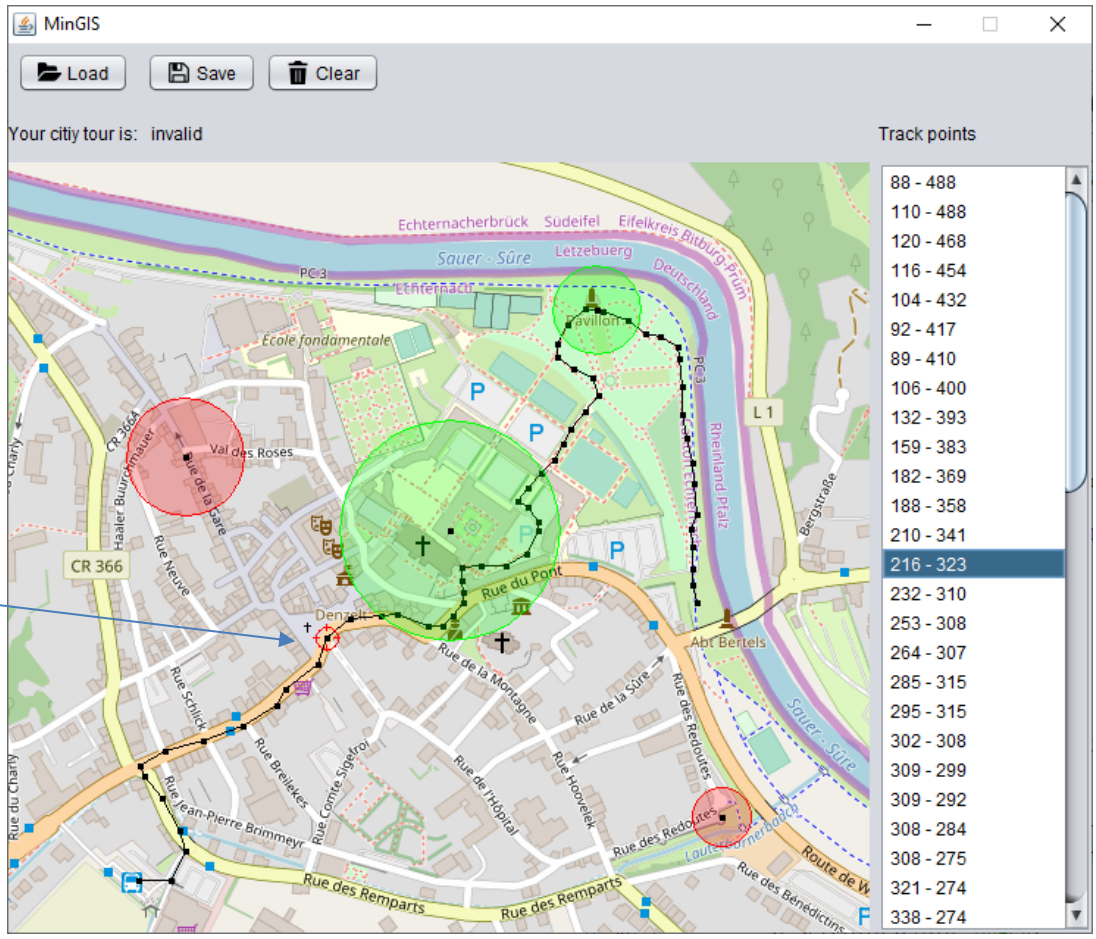
BRANCHE	SECTION(S)	ÉPREUVE ÉCRITE
SCIPR	**CI /CLI**	*Durée de l'épreuve :* *180 minutes* *Date de l'épreuve :*

Partie pratique

50 points

- Renommez le dossier **`NomPr123`** de manière qu'il porte votre code d'examen (exemple de notation : **`LXY_CI2_05`**). Le dossier que vous venez de renommer est celui qui sera évalué. Nous l'appellerons 'votre dossier' dans la suite.

Dans la partie pratique vous allez développer une application GIS (Geographic Information System) très simpliste pour la ville d'Echternach. Des milliers de touristes passent chaque année par la ville, sans pour autant visiter tous les points d'intérêt. C'est pour cette raison que la commune a décidé de créer une application dans laquelle on peut injecter une trace GPS et qui détermine alors si tous les points d'intérêt ont été visités ou non.



L'application, telle que présentée sur la capture ci-dessus, affiche la trace GPS (EN : track), les points d'intérêt (EN : points of interest) ainsi que la liste des points dont se compose la trace. Le point sélectionné dans la liste est mis en évidence par une petite cible rouge.

Un point d'intérêt possède un certain rayon qui définit sa surface. Il est reconnu comme étant visité si un point de la trace GPS se trouve à l'intérieur de sa surface. Les points d'intérêts visités sont dessinés en vert tandis que les autres sont dessinés en rouge.

Exemples

Point d'intérêt visité car il
y a un point de la trace à l'intérieur
de la surface du point d'intérêt.

Point d'intérêt ***non*** visité car il
n'y a pas de point de la trace à l'intérieur
de la surface du point d'intérêt.

Ouvrez le projet NetBeans « MiniGIS » qui se trouve dans votre dossier et complétez-le !

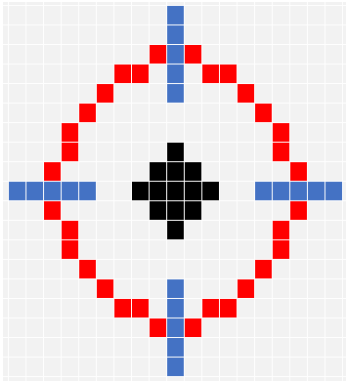
La classe « GisPoint » ***7 points***

Cette classe représente un point d'une trace. Elle possède les attributs suivants : **0.5 point**

- `x,y` sont les coordonnées du point
- `selected` indique si le point est sélectionné ou non

A côté du constructeur, des accesseurs et manipulateurs (voir schéma UML), les méthodes suivantes sont à implémenter : **0.5 point**

- La méthode `distanceTo(…)` calcule et retourne la distance entre le point actuel et celui passé en tant que paramètre. **2 points**
  Formule : $dist = \sqrt{(x_{from} - x_{to})^2 + (y_{from} - y_{to})^2}$ (→ Pythagore)
- La méthode `draw(…)` dessine le point comme disque noir avec un diamètre de 5 pixels centré aux coordonnées du point. Si le point est sélectionné, une cible avec un diamètre de 15 pixels est tracée en plus du point lui-même. **3 points**



Toute la cible est dessinée en couleur rouge, mais afin de mieux mettre en évidence les différentes parties, la figure ci-contre utilise plus de couleurs :

- En rouge le cercle avec un diamètre de 15 pixels.
- En bleu le réticule dont chaque élément a une longueur de 5 pixels.
- En noir le centre de la cible.

- La méthode `toString()` retourne une représentation textuelle du point de la forme suivante : **1 point**

`<X> - <Y>`

La classe « Track » *19 points*

Cette classe représente une trace GPS. Elle possède les attributs suivants : **0.5 point**

- `alPoints` qui est la liste des points dont se compose la trace

A côté des méthodes auto-générées (`add(…)`, `clear()`, `get(…)`, `toArray()`, `isEmpty()`) déléguant une partie du travail à la liste des points, les méthodes suivantes sont à implémenter : **1 point**

- La méthode `deselect()` désélectionne tous les points de la trace. **1.5 points**
- La méthode `loadFromFile(…)` remplace la liste des points actuels par les points contenus dans un fichier texte. Chaque ligne du fichier à importer contient une paire de coordonnées X et Y séparées par une virgule (voir le fichier fourni). **4 points**
- La méthode `saveToFile(…)` sauvegarde les points de la trace dans un fichier texte suivant le format indiqué ci-dessus, c'est-à-dire une paire de coordonnées X et Y séparées par une virgule par ligne. **3 points**
- La méthode `getClosestTo(…)` détermine et retourne le point appartenant à la trace qui est le plus proche du point passé en tant que paramètre. Si la liste est vide, la valeur `null` est retournée. **5 points**
- La méthode `draw(…)` dessine la trace complète suivant les indications suivantes : **4 points**
  - Tous les points de la trace sont dessinés.
  - Les points de la trace sont interconnectés par des segments de couleur noire.

La classe « PointOfInterest » *4 points*

Cette classe représente un point d'intérêt. Elle possède les attributs suivants : **0.5 point**

- `radius` le radius de la zone qui délimite le point d'intérêt
- `validated` indique si le point d'intérêt a été visité ou non

A côté des différents constructeurs, des accesseurs et manipulateurs (voir schéma UML), la méthode suivante est à implémenter : **0.5 point**

- La méthode `draw(…)` dessine le point d'intérêt. Si le point a été visité, il est dessiné en vert, sinon en rouge. La circonférence est tracée avec cette couleur. Pour la couleur de remplissage il faut utiliser la même couleur que celle de la circonférence mais dont la transparence est réduite à ¼. **3 points**

La classe « Node » *6 points*

Les points d'intérêt sont organisés dans un arbre binaire. Cette classe représente donc un élément de cet arbre. Elle possède les attributs suivants : **0.5 point**

- `left` la sous-branche gauche
- `right` la sous-branche droite
- `poi` une référence vers le point d'intérêt

A côté des différents constructeurs, des accesseurs et manipulateurs (voir schéma UML), les méthodes suivantes sont à implémenter : **0.5 point**

- La méthode `add(…)` ajoute un nouveau nœud avec le point d'intérêt passé en tant que paramètre au nœud actuel. L'insertion se fait sur base du rayon du point d'intérêt, c'est-à-dire que des points d'intérêt avec un rayon plus petit sont ajoutés à gauche tandis que les autres sont ajoutés à droite du nœud actuel (→ arbre binaire de recherche). **3 points**

- La méthode `draw(…)` dessine le point d'intérêt ainsi que, si possible, ceux de la sous-branche gauche et de la sous-branche droite. **2 points**

La classe « CityTour » *7 points*

Cette classe représente un tour à travers une ville. Un tour se compose de différents points d'intérêt qui sont stockés dans un arbre binaire de recherche. La classe possède l'attribut suivant : **0.5 point**

- `root` le nœud de base de l'arbre binaire

La classe possède les méthodes suivantes qui sont à implémenter :

- La méthode `add(…)` ajoute un nouveau nœud avec le point d'intérêt passé en tant que paramètre à l'arbre. **1 point**
- La méthode `draw()` dessine, si possible, les points d'intérêt de l'arbre sur le canevas passé en tant que paramètre. **0.5 point**
- Les méthodes `checkStatusFor(…)` testent une trace GPS par rapport au tour actuel : **5 points**
    - Si un point de la trace GPS passée en tant que paramètre se trouve à l'intérieur du de la surface d'un point d'intérêt, ce dernier est marqué comme étant visité. (→ voir dessin en haut de la page #2)
    - La méthode retourne `true` si tous les points d'intérêt ont été visités.

La classe « DrawPanel » *2 points*

Cette classe est responsable de la visualisation des éléments graphiques. Elle possède les attributs suivants : **0.5 point**

- `track` une trace GPS, `null` par défaut
- `cityTour` un tour de ville, `null` par défaut

A côté des manipulateurs des deux attributs, il faut modifier le code de la classe afin qu'elle dessine, si possible, la trace GPS ainsi que le tour de la ville. **1.5 points**

La classe « MainFrame » *5 points*

Cette classe est responsable de la visualisation des éléments graphiques. Elle possède les attributs suivants :

- `track` une trace GPS, à initialiser avec une nouvelle trace
- `cityTour` un tour de ville, à initialiser avec un nouveau tour de ville

La classe est déjà partiellement codée. À vous de compléter les éléments suivants :

- La méthode `updateView()` indique dans le libellé `validLabel` si la trace GPS actuelle valide le tour de ville (« valid » » ou non (« invalid ») et met à jour le dessin. **2 points**
- En cliquant avec la souris dans le dessin, un nouveau point est ajouté à la trace GPS et la vue est mis à jour. **1.5 points**
- En cliquant sur une coordonnée dans la liste des points de la trace GPS il faut que le point correspondant soit sélectionné et que donc la petite cible rouge doit y apparaitre afin de le mettre en évidence. **1.5 points**

Schéma UML du modèle à implémenter