Devoir surveillé

Durée de l'épreuve : 1 heure.

Tout document interdit, de même que l'usage de la calculatrice.

Exercice 1 : Points

On s'intéresse à la conception d'une classe `Point2D` représentant un point dans un repère à deux axes. Les contraintes de conception sont les suivantes :

- un objet `Point2D` est caractérisé par son couple de coordonnées `x` et `y` (valeurs entières bornées par des constantes `XMIN`, `XMAX`, `YMIN`, `YMAX`)
- les coordonnées sont fixées à la création de l'objet et ne peuvent plus être modifiées par la suite
- la création d'un objet `Point2D` avec des coordonnées en dehors des bornes génère une erreur typée par la classe `InvalidCoordinatesException`
- les méthodes `getX` et `getY` permettent d'obtenir les valeurs des coordonnées
- la méthode `translate` retourne un nouvel objet `Point2D` dont les coordonnées sont translatées respectivement de `deltaX` et `deltaY` pour les coordonnées `x` et `y`. Si la translation conduit à un dépassement des bornes, une erreur est générée (du même type que lors de la création de l'objet)
- les méthodes `toString`, `equals` et `hashCode` de la classe `Object` sont redéfinies de sorte que :
  - `toString` conduit à l'affichage `(5,3)` pour l'objet `Point2D` lorsque sa coordonnée `x` vaut 5 et sa coordonnée `y` vaut 3
  - deux objets `Point2D` sont équivalents s'ils possèdent les mêmes valeurs pour chacune de leurs coordonnées

Q 1 . Ecrire la classe `Point2D`

Q 2 . Ecrire la classe `Point3D`, représentant un point dans un repère à trois axes, avec des contraintes de conception similaires

Q 3 . Ecrire une application `TestPoint`, créant un objet `Point2D`, le translatant et l'affichant sur la sortie standard

Exercice 2 : Parcours

Un parcours peut se définir comme une suite de points visités, possédant un départ et une arrivée, et se construisant incrémentalement en ajoutant des points visités.

Q 1 . Ecrire la classe `Track2D`, en considérant que :

- lors de la création d'un parcours, il n'existe pas encore de point visité
- un parcours ne pas contenir plus de 20 points visités
- l'ajout d'un nouveau point visité en fin de parcours s'effectue via la méthode `addPointToVisit`, et soulève une exception de type `TooMuchPointsException` en cas d'échec
- la redéfinition de la méthode `toString` conduit à l'affichage du parcours, où les points visités sont affichés du départ à l'arrivée séparés par des signes supérieurs.
- la méthode `isLoop` permet de savoir si le parcours est une boucle
- la méthode `containsLoop` permet de savoir si le parcours possède une boucle