IUT Paris Descartes – Base de la Programmation Objet
DST 2015 – 2016

3 heures – tous documents autorisés – *barème indicatif*

Sur la route

Dans le cadre du développement d'une application de logistique pour le transport routier, il vous a été demandé de programmer le composant logiciel chargé de sélectionner les meilleurs itinéraires.

Après une rapide analyse, il a été décidé de représenter une route comme étant une succession de tronçons. Chacun de ces tronçons est caractérisé par sa distance (en km) et la vitesse maximale autorisée (en km/h). Une première version de la classe `Tronçon` a été définie. Elle est donnée en annexe.

1. (1 point) Ajoutez une méthode calculant le temps minimal de parcours (en h) du tronçon (i.e. lorsqu'il est parcouru à la vitesse maximale autorisée).
2. (3 points) Définissez la classe `Route`. Vous ferez en sorte que toute route soit composée d'au moins un tronçon et qu'il soit toujours possible d'agrandir une route par l'ajout d'un nouveau tronçon. De plus, il devra être possible de connaître la distance totale d'une route, son temps minimal de parcours et la vitesse moyenne que représente ce temps minimal.
3. (1 point) Écrivez un court programme définissant une route composée de trois tronçons successifs de 13, 11 et 26km sachant que le premier et le dernier ont une vitesse maximale autorisée de 130km/h alors que le deuxième est limité à 110km/h. Votre programme affichera la distance de cette route, son temps de parcours à vitesse maximale et la vitesse moyenne correspondant à ce temps de parcours.
4. (2 points) Vous vous apercevez que les tronçons et les routes partagent plusieurs caractéristiques : il est possible de connaître leur distance, leur temps de parcours minimal et une vitesse associée. Faites en sorte que ces spécificités soient décrites au sein d'une interface nommée `Trajet` et que les deux classes `Tronçon` et `Route` implémentent celle-ci. Cette interface doit être employée partout où elle peut l'être. En particulier, vous vous assurerez qu'une route doit être composée d'une succession de trajets et non pas de tronçons. Ne reproduisez pas tout le code des classes `Tronçon` et `Route` mais décrivez les modifications devant être apportées.
5. (3 points) La sélection des trajets va dépendre de critères qui pourront évoluer dans le futur. Pour permettre ces évolutions, il a été décidé d'introduire l'interface suivante :

```
public interface Critère {
  boolean estSatisfaitPar(Trajet t);
}
```

   Écrivez une implémentation de cette interface réalisant la conjonction (i.e. un "et" logique) entre deux critères donnés à la construction.
6. (2 points) Ajoutez à la classe précédente, une méthode de classe (i.e. statique) qui retourne un critère étant la conjonction de critères reçus sous la forme d'un tableau.

7. (4 points) Le développement de critères de sélection concrets nécessite que nous puissions accéder aux tronçons composant un trajet. Plus particulièrement, nous souhaitons pouvoir écrire des boucles de la forme suivante :

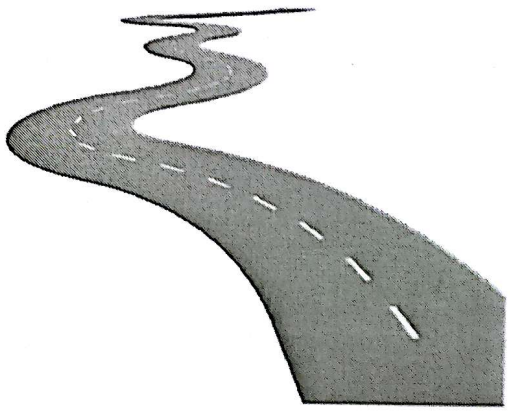
```
Trajet trajet = ...;
for (Tronçon t : trajet)
  System.out.println(t);
```

Pour pouvoir être parcouru à l'aide d'une boucle *foreach* comme ci-dessus, tout trajet doit implémenter l'interface standard `Iterable<Tronçon>`. Modifiez ce qui doit l'être pour que tout trajet soit ***itérable***. Les définitions des interfaces qui vous sont nécessaires (`Iterable<T>` et `Iterator<T>` – une interface associée) sont données en annexe.

Si vous ne savez pas comment implémenter `Iterable<Tronçon>`, vous êtes libres de proposer une solution alternative du moment qu'elle rend possible l'accès aux tronçons composant un trajet. Toutefois, la solution précédente devrait être privilégiée.

8. (2 points) Développez un critère (i.e. une classe implémentant `Critère`) qui soit satisfait par tout trajet dont au moins la moitié du parcours est de l'autoroute (i.e. la vitesse autorisée y est supérieure ou égale à 130km/h).

9. (2 points) Nous disposons d'au moins 7 classes et interfaces : (par ordre d'apparition) `Tronçon`, `Route`, `Appli`, `Trajet`, `Critère`, `Conjonction` et `AutorouteMajoritaire`. Dessinez le diagramme de classe correspondant en regroupant au sein de paquetages distincts ce qui doit l'être.



Annexe

```java
public class Tronçon {
  private double distance;
  private double vitesseMaximale;

  public Tronçon(double distance, double vitesseMaximale) {
    this.distance = distance;
    this.vitesseMaximale = vitesseMaximale;
  }

  public double getDistance() {
    return distance;
  }

  public double getVitesseMaximale() {
    return vitesseMaximale;
  }

  @Override
  public String toString() {
    return "[dist=" + distance + ",␣vitesseMax=" + vitesseMaximale + "]";
  }
}
```

Les commentaires avant chaque méthode sont ceux de la documentation officielle.

```java
public interface Iterable<T> {
  // Returns an iterator over a set of elements of type T.
  Iterator<T> iterator();
}
```

```java
public interface Iterator<T> {
  // Returns true if the iteration has more elements.
  boolean hasNext();
  // Returns the next element in the iteration (throwing
  // NoSuchElementException if the iteration has no more elements.
  T next();
  // Removes from the underlying collection the last element returned by
  // the iterator (optional operation throwing UnsupportedOperationException
  // if the remove operation is not supported by this Iterator).
  void remove();
}
```