Examen Ecrit

Tous documents sur support papier autorisés. Durée 2h.

Ce sujet vous propose de traiter quelques éléments pour un logiciel destiné au traitement d'informations relatives aux réseaux routiers. Nous étudierons la représentation de routes et de portions de routes. Vous pouvez utiliser des noms raccourcis pour les identifiants proposés, à condition de ne pas introduire d'ambiguïté.

Question 1. Savoir écrire un type-interface.

a- Ecrivez le code Java d'une interface `IPortionRoute` pour représenter une portion de route munie : (1) d'une méthode `coordGPSdep` permettant de connaître les coordonnées GPS du point de départ de la portion de route (sous forme d'une chaîne de caractères), (2) d'une méthode `coordGPSarr`, permettant de connaître les coordonnées GPS du point d'arrivée de la portion de route (sous forme d'une chaîne de caractères), (3) d'une méthode `longueur`, permettant de connaître la longueur (en km) de la portion de route. (4) d'une méthode `description`, retournant une chaîne de caractères composée des coordonnées GPS des points de départ et d'arrivée et de la longueur de la portion de route. Lorsque c'est possible, écrivez le code des méthodes.

b- (a) Pensez-vous que l'on puisse ajouter à l'intérieur de l'interface `IPortionRoute` le code suivant pour détenir un tableau associant des années et les coûts d'entretien au km de portions de routes pour ces années et pour initialiser ce tableau ? (b) Le contenu de ce tableau pourra-t-il être modifié par la suite, par exemple dans une méthode d'une classe implémentant l'interface ? Justifiez vos réponses.

```
HashMap<Integer,Double> coutKmAnnee = new HashMap<>();
static void init(){
    coutKmAnnee.put(2017, 50.0);
    coutKmAnnee.put(2018, 52.0);
}
```

Question 2. Savoir implémenter une interface.

a- Ecrivez une classe `PortionRoute` qui implémente l'interface `IPortionRoute`. Ecrivez les attributs et les méthodes utiles pour implémenter l'interface et une méthode `toString` retournant la description construite dans la question précédente (dans l'interface). La classe ne doit pas pouvoir être instanciée : quel élément introduisez-vous dans son entête pour le manifester ?

b- Ajoutez à la classe `PortionRoute` un constructeur initialisant tous les attributs, déclarant et signalant une exception si le point de départ et le point d'arrivée de la portion de route sont égaux (on suppose qu'il n'y a pas de portion de route formant une boucle). Vous devez créer votre propre classe pour représenter l'exception.

c- Ajoutez à la classe `PortionRoute` une méthode `equals`, redéfinissant celle de la classe `Object`, qui retourne vrai lorsque deux portions de routes ont le même point de départ et le même point d'arrivée.

d- Puis écrivez l'en-tête d'une classe `PortionRouteDepartementale` qui est sous-classe de `PortionRoute` et un constructeur permettant d'initialiser les attributs hérités. Ce constructeur peut-il être interrompu par l'exception prévue ci-dessus ? Si oui, quelle conséquence cela a-t-il sur son entête dans la classe `PortionRouteDepartementale` ? Pour la suite, les accesseurs des deux classes sont tous supposés exister avec les conventions classiques.

Question 3. Ecrire une classe générique

L'objectif est de représenter des routes composées de portions d'un même type (instances d'une même classe, par exemple uniquement des portions de routes départementales ou uniquement des portions de routes nationales, ou uniquement des portions d'autoroutes).

a- Ecrivez le code Java d'une classe **générique** `Route`. Dans cette question, écrivez un embryon de classe générique `Route` avec :

— son entête,

— un attribut permettant de stocker la liste des portions de la route (utiliser une `ArrayList` est recommandé ici). Initialisez l'attribut lors de sa déclaration afin que la liste soit créée et vide au départ.

b- Ecrivez dans la classe **générique** `Route` une méthode d'ajout d'une portion de route. Si la portion de route ajoutée est déjà présente, un message d'erreur est affiché. De plus écrivez le code permettant de munir la méthode d'ajout d'une assertion qui vérifie qu'après ajout, la liste contient zéro ou un élément de plus.

Question 4. Savoir écrire un main

Le programme suivant comprend (1) la création de trois portions de routes (2) la création d'une route et l'ajout des portions à cette route.

a- Complétez le `main` pour préciser les paramètres de généricité et pour capturer les exceptions identifiées à la question plus haut qui pourraient potentiellement y surgir.

```
public static void main(String[] args) {
   ......  à compléter ......
   PortionRouteDepartementale p1 =
         new PortionRouteDepartementale(" 48.8,2.35", "48.99,2.39",12.0,2.0);
   PortionRouteDepartementale p2 =
         new PortionRouteDepartementale(" 48.8,2.35", "48.99,2.39",12.0,2.0);
   PortionRouteDepartementale p3 =
         new PortionRouteDepartementale(" 49.85,3.35", "49.99,3.39",34.0,1.5);

   Route< ......  à compléter ......> route1 = new Route< ......  à compléter ......>();
   route1.ajout(p1);
   route1.ajout(p2);
   route1.ajout(p3);
   ......  à compléter ......
}
```

b- Indiquez combien `route1` contient de portions à la fin du programme.

c- Ajoutez une instruction déclenchant l'exception de la question 2b.

d- Peut-on écrire dans le programme cette instruction (Justifiez) :

```
Route<PortionRoute> route2 = new Route<PortionRouteDepartementale>();
```

Question 5. Comprendre les streams et ajouter une méthode manipulant la structure (question optionnelle, à faire seulement si vous avez fini le reste à temps)

En supposant que la liste des portions d'une route se nomme `portions`. On peut écrire la méthode suivante basée sur les `streams` dans la classe `Route`.

```
   public double methodeMystere(){
      return this.portions
                        .stream()
                        .mapToDouble(p -> p.longueur())
                        .average()
                        .getAsDouble();
   }
```

Expliquez ce qu'elle fait et réécrivez cette méthode avec une itération classique (`for` ou `while`).