IUT2 Université Grenoble Alpes
Département INFO

R3.02 – Développement efficace

TP 3 *(associé au Cours 3)*

Arbres n-aires (Mondial)

Pour ce TP

- Télécharger et compléter le projet CLion « R302-TP3 »
- Il n'y a qu'une configuration d'exécution.

Note : Les méthodes `int getNbAirports() const` et `void printCountiesCode() const` sont fournies.

TP individuel noté

Ce TP sera noté. Cette semaine, vous disposez de deux séances encadrées. La semaine prochaine, vous disposez d'une la séance encadrée et la seconde séance sera consacrée à une démonstration notée.

Attention On vous rappelle qu'il s'agit d'un travail individuel et que nous disposons de tous les outils nécessaires pour contrôler que votre code est bien original. De plus les questions qui vous seront posées lors de la démonstration permettront de cerner votre compréhension du travail effectué.

Excercie 1 : Nombre de déserts (itératif)

Dans la classe **Mondial**, implanter la méthode itérative publique `int getNbDeserts() const` (algorithme itératif) qui compte le nombre de déserts référencés dans la base de donnée, et la tester.

Excercie 2 : Nombre d'éléments d'une catégorie (itératif)

Dans la classe **Mondial**, implanter la méthode itérative `int getNbElemCat(string categoryName)` qui compte le nombre d'entrées de type `categoryName` présents dans la base de données, et la tester.

Excercie 3 : Code d'un pays identifié par son nom (worker auxiliaire récursive)

Dans la classe **Mondial**, implanter la méthode privée suivante qui retourne un pointeur de type `XMLElement*` sur l'**élément** `<country>` correspondant à un pays identifié par son nom (`countryName`) :

```
XMLElement* getCountryXmlelementFromNameRec(string countryName) const;
```

Note : Si le pays n'existe pas la méthode retourne `nullptr`.

Cette méthode utilise le worker récursif suivant :

```
XMLElement* getCountryXmlelementFromNameRecWorker(XMLElement* currentCountryElement,
                                                  string countryName) const;
```

Pour la tester, implanter la méthode publique suivante qui retourne le code d'un pays identifié par son nom (`countryName`) :

```
string getCountryCodeFromName(string countryName) const throw (PrecondVioleeExcep);
```

Excercie 4 : Population d'un pays identifié par son nom

Dans la classe **Mondial**, implanter la méthode publique suivante qui retourne la valeur de la dernière mesure de population d'un pays identifié par son nom (`countryName`), et la tester :

```
int Mondial::getCountryPopulationFromName(string countryName) const;
```

Note : Si le pays n'existe pas la méthode retourne `-1` et si aucune population n'est renseignée, la méthode retourne 0.

Cette méthode utilise la méthode iterative privée suivante qui retourne un pointeur de type `XMLElement*` sur l'**élément** `<country>` correspondant à un pays identifié par son nom (`countryName`) :

```
XMLElement* getCountryXmlelementFromNameIter(string countryName) const;
```

Note : Si le pays n'existe pas la méthode retourne `nullptr`.

Excercie 5 : Pays frontaliers d'un pays identifié par son nom

Dans la classe **Mondial**, implanter puis tester la méthode publique suivante qui affiche, pour un pays identifié par son nom (`countryName`), le nom de chaque pays frontalier et la longueur de la frontière entre ce dernier et le pays identifié :

```
void Mondial::printCountryBorders(string countryName) const;
```

Voir le sujet de TD 2.2 pour plus d'informations.

Note 1 : Dans si le pays n'existe pas ou si le pays n'a pas de pays frontalier, la méthode affichera un message approprié pour ces deux situations.

Note 2 : Cette méthode utilisera la méthode privée suivante , à implanter, qui retourne un pointeur de type `XMLElement*` sur l'**élément** `<country>` correspondant à un code d'identification d'un pays (`countryCode`) :

```
XMLElement* Mondial::getCountryXmlelementFromCode(string countryCode) const;
```

Excercie 6 : Pays traversés par un fleuve

Un fleuve est défini par un **élément** <river>. Aller voir la définition dans la dtd. On donne ci-dessous un exemple :

```
<river id="river-Rhein" country="D CH FL A F NL">
  <name>Rhein</name>
  <located country="D" province="prov-Germany-2 prov-Germany-8 prov-Germany-11 prov-Germany-12"/>
  <located country="F" province="prov-France-2"/>
  <located country="A" province="prov-Austria-4"/>
  <located country="CH" province="prov-Switzerland-2 … prov-Switzerland-27"/>
  <located country="NL" province="prov-Netherlands-7 prov-Netherlands-10 prov-Netherlands-12"/>
  <through lake="lake-Bodensee"/>
  <to watertype="sea" water="sea-Nordsee"/>
  <area>198735</area>
  <length>1324</length>
  <source country="CH">
    <mountains>Alps</mountains>
    <located country="CH" province="prov-Switzerland-11"/>
    <latitude>46.6</latitude>
    <longitude>8.7</longitude>
    <elevation>2345</elevation>
  </source>
  <estuary country="NL">
    <located country="NL" province="prov-Netherlands-10"/>
    <latitude>51.9</latitude>
    <longitude>4.00</longitude>
    <elevation>0</elevation>
  </estuary>
</river>
```

Comme on le voit dans l'**élément** ci-dessus concernant le Rhin (Rhein en anglais), les pays que traversent un fleuve sont renseignés de deux manières dans un **élément** `<river>`.

1) Sous la forme d'une liste des pays traversés par le fleuve fournie par l'**attribut** `country` de l'**élément** `<river>`. C'est une chaîne de caractères séparés par des blancs dont les « mots » sont des codes d'identification de pays, (`<river id="river-Rhein" country="D CH FL A F NL">`).
2) **S**ous la forme d'éléments `<located>` (un par pays traversé) qui possèdent deux **attributs :**
   - `country` qui est le code d'identification du pays traversé, et
   - `province` qui est la liste des régions administratives du pays que le fleuve traverse. Cette liste est une chaîne de caractères séparés par des blancs dont les « mots » sont des codes d'identification de régions administratives (`<located country="D" province="prov-Germany-2 prov-Germany-8 prov-Germany-11 prov-Germany-12"/>`).

Dans la classe **Mondial**, implanter puis tester la méthode publique suivante qui affiche, pour un fleuve identifié par son nom (`riverName`), sa longueur ainsi que le nom de tous les pays qu'il traverse en utilisant la représentation 1)[1], l'**attribut** `country` de l'**élément** `<river>`, des pays traversés ci-dessus.

```
void Mondial::printAllCountriesCrossedByRiver(string riverName) const;
```

Dans la classe **Mondial**, implanter puis tester la méthode publique suivante qui affiche, pour un fleuve identifié par son nom (`riverName`), sa longueur ainsi que le nom de tous les pays qu'il traverse en utilisant la représentation 2)[2], des **éléments** `<located>` fils de l'**élément** `<river>`, des pays traversés ci-dessus.

[1] Cf. le sujet du TD 2.2 Tâche 4.

```
void Mondial::printCountriesWithProvincesCrossedByRiver(string riverName) const;
```

Note : Ces deux méthodes utiliseront la méthode privée, à implanter, qui retourne un pointeur de type `XMLElement*` sur l'**élément** `<river>` correspondant au nom de fleuve `riverName` :

- `XMLElement* Mondial::getRiverXmlelementFromName(string riverName) const;`

Excercie 7 : Pays et provinces traversées par un fleuve *(difficile)*

On rappelle que l'**élément** `<river>` ne possède pas autant de fils `<located>` que de pays traversés. Seuls les pays avec division administrative traversés par le fleuve apparaissent dans un **élément** `<located>`.

On rappelle qu'un **élément** `<located>` possède deux **attributs** comme on peut le voir dans l'exemple ci-dessous :

```
<located country="D" province="prov-Germany-2 prov-Germany-8 prov-Germany-11 prov-Germany-12"/>
```

L'**attribut** `country` est le code d'identification du pays traversé et l'**attribut** `province` est une chaîne de caractères dont les mots sont les codes d'identification des divisions administratives du pays traversé.

Dans la classe **Mondial**, implanter puis tester la méthode publique suivante qui affiche, pour un fleuve identifié par son nom (`riverName`), sa longueur, le nom de tous les pays qu'il traverse, et pour chaque pays traversé qui possède des divisions administratives, le nom de la ou des divisions administratives qu'il traverse.

```
void Mondial::printCountriesAndProvincesCrossedByRiver(string riverName) const;
```

Note : On veillera à minimiser le nombre d'éléments des données visités.

Excercie 8 : Informations sur une ville *(difficile)*

Dans la classe **Mondial**, implanter puis tester la méthode publique suivante qui affiche, pour une ville identifiée par son nom (`cityName`), le nom du pays dans laquelle elle se trouve, sa latitude, sa longitude, sa dernière population renseignée et le nom de la division administrative dans laquelle elle se trouve.

```
void Mondial::printCityInformation(string cityName) const;
```

Note : On veillera à minimiser le nombre d'éléments des données visités.

Excercie 9 : À vous de jouer !

Proposer l'implantation d'autres méthodes qui interrogent les données…