Nom : .....

Prénom : .....

Groupe : .....

Identifiant : .....

/ 50

Haute École de Bruxelles
École Supérieure d'Informatique
Bachelor en Informatique

2014 – 2015

Développement – 1ère

Examen de Janvier

Panneau de progression des métros

Vendredi 16 Janvier 2015

Nous allons simuler le panneau montrant la progression des rames [1] de métro station par station sur une seule ligne de métro.

Voici un exemple obtenu par notre simulation où les tirets (-) montrent l'absence de rame de métro et les croix (X) l'emplacement des rames de métro. La première à la station Parc, la deuxième à la station De Brouckère et la troisième à St Catherine...

Art Loi	Parc	Gare centrale	De Brouckère	St Catherine
-	X	-	X	X

Nous utiliserons deux tableaux à une dimension :

- Tableau **noms** de chaines qui reprend les noms des stations dans l'ordre des stations sur la ligne.
- Tableau **rames** de caractères qui reprend l'emplacement des rames de métro sur la ligne.

Attention, il ne peut avoir deux rames de métro dans une seule station. Une rame de métro ne peut dépasser une autre. Dans ces deux cas, la rame de métro sera mise à l'arrêt dans une station afin de ne pas provoquer un accident. Une rame de métro met 2 minutes pour passer d'une station à la suivante.

1. rame : Groupe de voitures ou de wagons de chemin de fer effectuant un même parcours (Larousse)

Version du document : 11 janvier 2015

I
Algorithmique

Consignes

Pour la partie algorithmique,
- Vous ne pouvez pas utiliser de notes.
- Vos réponses se feront au bic bleu ou noir sur la feuille de réponses.
- Sauf spécification du contraire, les données lues ou reçues ne comportent pas d'erreurs.
- Les noms en gras (variables et types) doivent être respectés.
- Veillez à travailler de manière modulaire.

1 Position d'une station (5 points)

Écrivez un module

module *positionStation*(noms↓ : tableau [1 à n] de chaines, station↓ : chaine) → **entier**

où la chaine **station** est le nom de la station recherchée et le tableau **noms** contient déjà les noms des stations. Ce module renvoie l'indice dans le tableau correspondant à la station donnée en paramètre. Si la station n'existe pas dans **noms**, le module retourne la valeur -1.

Exemple : le tableau **noms** égal à

Art Loi	Parc	Gare centrale	De Brouckère	St Catherine

si **station** = Parc, le module retournera 2

si **station** = Alma, le module retournera -1

2 Créer le panneau (8 points)

Écrivez un module

module *créerPanneau*(noms↓ : tableau [1 à n] de chaines) → **tableau** [1 à n] de caractères

où **noms** contient les **n** noms des stations et renvoie le tableau **rames** de **n** caractères. (Rappel : Dans le tableau **rames**, le caractère X marque la présence d'une rame de métro, le caractère - marque l'abscence de rame de métro dans la station.) Pour construire ce tableau **rames**, ce module lit les noms des stations où on veut placer une rame de métro. Vous utiliserez une chaine vide comme valeur sentinelle de fin de lecture. Vous permettez à l'utilisateur de se tromper : s'il encode une station inconnue ou une station déjà occupée par une rame de métro, vous lui signalez l'erreur et il peut continuer son encodage.

Exemple : le tableau **noms** égal à

Art Loi	Parc	Gare centrale	De Brouckère	St Catherine

et les noms de station lues suivantes :De Brouckère, St Catherine, Parc, ""

le module retournera le tableau :

-	X	-	X	X

3 Afficher le panneau (4 points)

Écrivez un module

module *afficherPanneau*(noms↓ : tableau [1 à n] de chaines, rames↓ : tableau [1 à n] de caractères)

qui affiche le panneau de progression des rames de métro. Ce module reçoit en paramètre les deux tableaux **noms** et **rames** et affichera le panneau suivant l'exemple ci-dessous (Nous l'affichons verticalement par facilité) :

Exemple : Pour les tableaux **noms** et **rames** suivants

Art Loi	Parc	Gare centrale	De Brouckère	St Catherine
-	X	-	X	X

Le module affichera :

-	Art Loi
X	Parc
-	Gare centrale
X	De Brouckère
X	St Catherine

4 Temps d'attente (8 points)

Écrire un module

module *tempsAttente*(noms↓ : tableau [1 à n] de chaines, rames↓ : tableau [1 à n] de caractères, station↓ : chaine) → **entier**

qui retourne le nombre de minutes qu'il faudra attendre pour qu'une rame de métro arrive dans une station donnée. Ce module reçoit les tableaux **rames** et **noms** et la chaine **station** reprenant le nom de la station où attend le passager. (Rappel : une rame de métro met 2 minutes pour passer d'une station à la suivante) Si la station passée en paramètre est inconnue, l'algorithme provoquera une erreur. Si une rame de métro est déjà dans la station, le temps d'attente est naturellement de zéro. S'il n'y a aucune rame de métro dans toutes les stations précédentes, le temps d'attente sera inconnu et le module retournera la valeur -1.

Exemple : Pour ces tableaux **noms** et **rames** :

Art Loi	Parc	Gare centrale	De Brouckère	St Catherine
-	X	-	-	X

Le temps d'attente à la station Art Loi est inconnu (-1) ; à la station Parc de 0 minute et à la station De Brouckère de 4 minutes.

II
Java et laboratoire

Consignes

Pour la partie java,

- *Vous réaliserez votre travail sur* `linux1` *et le déposerez dans le casier linux de votre professeur par la commande* `casier`.
- *Vous disposez de toutes vos notes ainsi que de l'aide en ligne.*
- *Il ne suffit pas que votre code compile. Testez-le pour identifier d'éventuelles erreurs à l'exécution.*
- *La cotation tiendra compte aussi du style de programmation que vous avez acquis.*
- *Respectez bien les noms de package, classe, méthodes demandés dans l'énoncé.*
- *Vous remplacerez bien sûr g12345 par votre numéro d'étudiant.*

5 Question préalable (0 point)

Créez un répertoire `evaluations/janvier`. Changez les droits sur votre répertoire `janvier` pour donner les permissions de lecture et d'exécution aux professeurs mais aucun droit aux autres étudiants. Appelez votre professeur pour lui montrer que vos permissions ont bien été changées.

Vous ne continuerez pas l'examen tant que cette question n'a pas été validée par votre professeur.

6 Travailler dans un package (2 points)

Dans la suite, votre classe fera partie du package `g12345.evaluation.janvier`

Votre programme s'appellera `Metro.java`

Ses sources seront dans `~/evaluations/janvier/src`, et la version compilée sera dans `~/evaluations/janvier/` (ou un sous-répertoire adéquat).

Écrivez dans les cadres correspondants les réponses aux questions suivantes :

- l'instruction que doit contenir votre classe pour faire partie du package demandé ;
- la commande (complète et précise) que vous allez utiliser (à partir du répertoire dans lequel se trouvent les sources) pour **compiler** votre classe ;

– la commande (complète et précise) que vous allez utiliser pour **exécuter** votre classe ;

– le contenu minimal de votre variable d'environnement *CLASSPATH*.

7 Coder les algorithmes (14 points)

Écrivez les méthodes suivantes, préparées dans la partie algorithmique :

- `public static int positionStation(String[] noms, String station)`
- `public static char[] créerPanneau(String noms)`
- `public static void afficherPanneau(String[] noms, char[] rames)`
- `public static int tempsAttente(String [] noms, char[] rames, String station)`

Procédez pas à pas, et écrivez une méthode principale pour tester votre code. Celle-ci devra :

1. créer le panneau afin d'obtenir les tableaux **noms** et **rames** initialisées à

Art Loi	Parc	Gare centrale	De Brouckère	St Catherine
-	X	-	-	X

2. afficher le panneau,
3. calculer le temps d'attente aux stations Art Loi, Parc et De Brouckère.
   Vous devriez trouver que le temps d'attente à la station Art Loi est inconnu (-1) ; à la station Parc de 0 minute et à la station De Brouckère de 4 minutes.
   Votre professeur passera pour vérifier ce point.

8 La javadoc (4 points)

Si vous ne l'avez pas fait au fur et à mesure, écrivez la javadoc pour les méthodes `positionStation` et `tempsAttente` et générez la dans le répertoire `~/evaluations/janvier/doc`.

Écrivez ici la commande utilisée pour générer cette javadoc.

9 Test JUnit (5 points)

Écrivez 1 test unitaire pour la méthode `tempsAttente` et exécutez ce test en utilisant les commandes que vous écrivez ci-dessous :

– la commande permettant de lancer votre test JUnit [2],

2. Inutile d'écrire ici un alias

– le contenu de la variable d'environnement CLASSPATH pour pouvoir exécuter ces tests,

```

```

– une commande permettant de rediriger le résultat du test (sans écraser les éventuels précédents résultats) dans le fichier `~/evaluations/janvier/tests.log`.

```

```