



Liberté • Égalité • Fraternité

RÉPUBLIQUE FRANÇAISE

MINISTÈRE DE LA TRANSITION ÉCOLOGIQUE ET
SOLIDAIRE

MINISTÈRE DE LA COHÉSION DES TERRITOIRES

**EXAMEN PROFESSIONNEL DE VÉRIFICATION
D'APTITUDE AUX FONCTIONS DE
PROGRAMMEUR**

Session 2018

Langage : **Java**

Durée : **5 heures**

Coefficient : **4**

Notation : sur **20**

Nombre de pages du sujet : **9** (y compris cette page)

Remarques générales

1. Le sujet comporte trois parties indépendantes :
 - Une série de 5 questions notée sur 5 points
 - Exercice 1 “ WAR of EVA ” noté sur 6
 - Exercice 2 “ Exploitation de données de mesure du trafic ” noté sur 9
2. Si éventuellement il vous manquait des informations pour développer votre sujet, il vous revient de retenir les hypothèses adaptées à votre solution, en les explicitant clairement.
3. Aucun document ou matériel électronique (calculatrice, ordinateur ...) n'est autorisé.

Examen professionnel de vérification d'aptitude aux fonctions de programmeur			Session 2018
Épreuve écrite : langage Java	Durée : 5h	Coefficient : 4	Page : 1/9

1. Questions

Répondez de façon précise et concise aux cinq questions suivantes.

- 1.1 Quelle est l'utilité d'un système de gestion de version. Pouvez-vous en citer au moins deux en précisant leurs spécificités ?*
- 1.2 Pouvez-vous expliquer la méthode test-driven development (TDD) ou en français développement piloté par les tests et préciser ses apports ?*
- 1.3 Quels sont les avantages et inconvénients du langage Java par rapport aux langages C++ et PHP ? Pour quels types de problèmes Java est-il le mieux adapté ?*
- 1.4 Qu'est-ce que l'injection SQL et comment s'en prémunir ?*
- 1.5 Expliquez le rôle du Garbage Collector et précisez ses avantages et inconvénients ?*

Examen professionnel de vérification d'aptitude aux fonctions de programmeur			Session 2018
Épreuve écrite : langage Java	Durée : 5h	Coefficient : 4	Page : 2/9

2. Exercice 1 “ War of EVA ”

2.1. Description



Figure 1: Écran d'accueil

Cet exercice propose une série de questions conduisant à la création du jeu war of EVA.

Ce jeu (Cf Figure 1) est un jeu similaire au Touché-Coulé aussi nommé Bataille Navale.

Il consiste à placer une flotte de bateaux de différentes tailles sur une grille carrée de dimension 10 x 10. Puis tour à tour les joueurs tirent sur une case de la grille de l'adversaire. Si la case est occupée par un navire, l'adversaire annonce qu'il est touché et dans le cas contraire il dit que le tir est loupé. Lorsqu'un bateau entier est touché, le joueur annonce coulé. Le but du jeu est de couler tous les navires de la flotte adverse.

2.2. Initialisation

Pour commencer le jeu, il faut avoir placé les 5 bateaux dans sa grille de jeu. Chaque joueur dispose d'un porte-avion (taille : 5 cases), d'un croiseur (taille : 4 cases), d'un contre-torpilleur (taille : 3 cases), d'un sous-marin (taille : 3 cases) et d'un torpilleur (taille : 2 cases). Deux bateaux ne peuvent pas se trouver sur une même case et leur positionnement doit être horizontal ou vertical (la diagonale est interdite).

2.2.a Proposez une solution pour initialiser la grille d'un joueur et pour stocker les données de la partie.

2.2.b Implémentez la fonction `initJeu()` en Java

Examen professionnel de vérification d'aptitude aux fonctions de programmeur			Session 2018
Épreuve écrite : langage Java	Durée : 5h	Coefficient : 4	Page : 3/9

2.3. Affichage du plateau de jeu

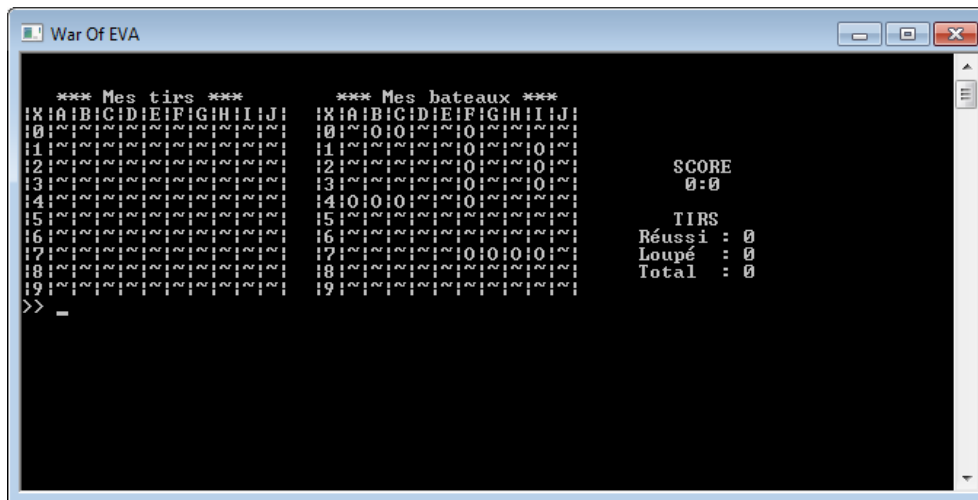


Figure 2: Plateau de jeu

La grille de droite montre la mer avec l'emplacement des bateaux du joueur. La grille de gauche correspond au radar qui garde en mémoire les tirs effectués et leurs résultats en reproduisant la grille de l'adversaire.

- ~ : Représente la mer
- * : Représente un tir loupé
- X : Représente un tir réussi
- O : Indique qu'une case est occupée par un navire

Écrivez, en Java, une fonction `afficheJeu()` qui permet d'afficher le plateau de jeu au cours de la partie tel que présenté sur la Figure 2.

2.4. Déroulement de la partie

La partie ne peut démarrer que lorsque les deux adversaires ont placé leurs 5 bateaux.

Le jeu commence par le tir du joueur qui n'héberge pas la partie. Lorsqu'un joueur aura saisi les coordonnées d'une case pour effectuer son tir, un avertissement indiquera s'il a touché, coulé ou loupé un des bateaux du joueur adverse.

L'affichage du plateau de jeu se met à jour après chaque coup tiré par un joueur.

Le jeu prend fin dès que la flotte d'un des joueurs est entièrement coulée.

Pour la suite de l'exercice vous utiliserez les fonctions `send()` / `receive()` qui ont pour fonctions respectives d'envoyer / recevoir une information de la part du joueur adverse.

La fonction `send()` prend en argument une chaîne de caractères. La fonction `receive()` retourne une chaîne de caractères.

2.4.a Décrivez par un logigramme le déroulement d'une partie.

2.4.b Écrivez, en Java, les fonctions qui mettent à jour les grilles de jeu.

2.4.c Écrivez, en Java, les fonctions `tirReussi()`, `tirLoupé()` et `tirTotal()` qui affichent les statistiques des joueurs.

2.4.d Implémentez, en Java, une fonction `testSaisie()` qui doit détecter les erreurs de saisies au cours du jeu.

2.4.e Écrivez la classe principale de votre jeu qui permet le déroulement du jeu de la phase d'initialisation à la fin du jeu lorsqu'un joueur a tous ses bateaux coulés.

Examen professionnel de vérification d'aptitude aux fonctions de programmeur			Session 2018
Épreuve écrite : langage Java	Durée : 5h	Coefficient : 4	Page : 4/9

3. Exercice 2 : Exploitation de données de mesure du trafic

Cet exercice consiste en la réalisation d'un prototype d'exploitation de données de mesure du trafic.

Pour les parties de code qu'il vous est demandé d'écrire, vous préciserez, le cas échéant, les librairies que vous utilisez.

Le recueil de données de trafic repose sur :

- des capteurs qui réagissent aux passages des véhicules ;
- des stations de regroupement qui centralisent les données transmises par les capteurs ;
- un dispositif d'exploitation des données trafic dont vous allez prototyper certaines fonctions.

Chaque capteur transmet au fil de l'eau à la station de regroupement à laquelle il est rattaché les informations qu'il mesure.

Les stations de regroupement déposent toutes les trois minutes, dans le dossier `"/incoming"` d'un serveur ftp, un fichier reprenant les mesures de leurs capteurs.

Chaque fichier comporte une ligne par mesure. Une mesure correspond au passage d'un véhicule au niveau du capteur.

Chaque ligne de mesure est structurée comme suit :

Format et longueur	Contenu	Exemples
10 chiffres	Identifiant unique du point de mesure	0001282727
1 caractère	Séparateur (code ASCII 124 – Barre verticale)	
1 caractère	Nature de la route correspondant à la mesure : • A → autoroute • N → route nationale • R → route régionale • D → route départementale • B → bretelle • X → autre type de voie	A0014 N0007 D1057 B7521
4 chiffres	Numéro de la route	
1 caractère	Séparateur (code ASCII 124 – Barre verticale)	
3 chiffres	Trois chiffres indiquant le département où est localisé le point de mesure. Exception pour la Corse où le dernier caractère n'est pas un chiffre, les codes étant 02A et 02B	001 020 02A 092 972
1 caractère	Séparateur (code ASCII 124 – Barre verticale)	
3 chiffres	Numéro du point de repère précédant le point de mesure dans le sens des points de repère croissants.	018
4 chiffres	Distance entre le point de mesure et le point de repère, mesurée en mètres dans le sens des points de repère croissants	0129
1 caractère	Séparateur (code ASCII 124 – Barre verticale)	

Examen professionnel de vérification d'aptitude aux fonctions de programmeur			Session 2018
Épreuve écrite : langage Java	Durée : 5h	Coefficient : 4	Page : 5/9

Format et longueur	Contenu	Exemples
1 caractère	Sens de circulation correspondant à la mesure : + → sens des points de repère croissants, signe plus, code ASCII 043 - → sens des points de repère décroissants, signe moins, code ASCII 045 * → mesure les deux sens de circulation, étoile, code ASCII 042	+ - *
1 caractère	Séparateur (code ASCII 124 – Barre verticale)	
12 caractères	Latitude des coordonnées GPS du point de mesure exprimée sous la forme : 2 chiffres : valeur degrés - signe ° (degré, code ASCII 176) 2 chiffres : valeur minutes - signe ' (apostrophe simple, code ASCII 039) 2 chiffres : valeur secondes - signe . (point utilisé comme séparateur décimal, code ASCII 046) 1 chiffres : dixième de secondes - signe " (apostrophe double, code ASCII 034) 1 caractère : N ou S	48°53'32.9"N 14°36'52.3"N 20°53'17.6"S
1 caractère	Séparateur Latitude , Longitude , (virgule, code ASCII 044)	,
12 caractères	Longitude des coordonnées GPS du point de mesure exprimée sous la forme : 2 chiffres : valeur degrés - signe ° (degré, code ASCII 176) 2 chiffres : valeur minutes - signe ' (apostrophe simple, code ASCII 039) 2 chiffres : valeur secondes - signe . (point utilisé comme séparateur décimal, code ASCII 046) 1 chiffres : dixième de secondes - signe " (apostrophe double, code ASCII 034) 1 caractère : E ou W	02°14'09.2"E 61°02'53.9"W 55°26'54.0"E
1 caractère	Séparateur (code ASCII 124 – Barre verticale)	
10 chiffres	Date-heure de la mesure au format unix time code (nombre de secondes depuis le 1 ^{er} janvier 1970)	1537263020 pour le 18/09/2018 à 9h30 et 20 secondes
1 caractère	Séparateur (code ASCII 124 – Barre verticale)	

Format et longueur	Contenu	Exemples
1 chiffre	Nature du véhicule capté : • 1 : véhicule léger • 5 : poids lourd • 9 : deux roues	1 5 9

3.1. Validation de ligne

La fonction `ValideMesure` permet de vérifier la validité d'une ligne de mesure. Elle prend pour unique paramètre la chaîne de caractères correspondant à une ligne de mesure et retourne une valeur booléenne valant `Vrai` si la ligne est valide et `Faux` sinon.

Écrivez le logigramme de la fonction `ValideMesure`.

Écrivez cette fonction en Java

3.2. Validation de ligne nouvelle version

Les premiers tests de la fonction `ValideMesure` ont permis de mettre en évidence que le numéro de la route n'était pas toujours écrit sous la forme de 4 chiffres. Selon les cas, ce numéro peut être :

- 4 chiffres comme précisé dans les spécifications initiales, exemple "0014".
- une chaîne de 4 caractères se terminant par 1 à 3 espaces (code ASCII 032) selon le nombre de chiffres du numéro de la route, exemple "14 · ·" (· représente le caractère espace).
- Une chaîne de 1 à 4 chiffres selon le nombre de chiffres du numéro de la route, exemple "14"

Modifiez la fonction `ValideMesure` pour tenir compte de ces différentes situations valides.

3.3. Validation des fichiers

La fonction `ValideFichiers` permet de vérifier les fichiers présents dans le répertoire `/incoming`. Les fichiers valides sont déplacés dans le répertoire `/Import`, les fichiers invalides sont déplacés dans le répertoire `/Erreur`.

Un fichier est considéré comme non valide si 10 % ou plus des lignes de mesure qu'il contient sont invalides.

Écrivez le logigramme de la fonction `ValideFichiers`.

Écrivez cette fonction en Java. La valeur seuil de 10 % est amenée à évoluer dans le temps, vous ferez en sorte que cette valeur puisse être facilement modifiable.

3.4. Détermination du trafic par point de mesure

En exploitant les lignes valides des fichiers valides, la fonction `MesureTrafic` restitue un tableau proposant pour chaque point de mesure :

- l'identifiant unique du point de mesure
- la localisation complète du point de mesure (Route, Département, Point repère, distance au point repère, latitude, longitude, sens de circulation mesuré)
- la valeur de la mesure du trafic pour les trois dernières minutes.

Un point de mesure n'est présent que dans un seul fichier. Mais il peut être présent plusieurs fois dans ce fichier, sans tri par point de mesure. Les mesures sont ajoutées dans ce fichier sous la forme d'une nouvelle ligne chaque fois qu'une mesure est transmise par un capteur à sa station de regroupement.

Examen professionnel de vérification d'aptitude aux fonctions de programmeur			Session 2018
Épreuve écrite : langage Java	Durée : 5h	Coefficient : 4	Page : 7/9

Chaque station regroupe au plus 100 points de mesure. Chaque fichier comporte autant de lignes que de véhicules détectés pendant la séquence de mesure de trois minutes.

Les fichiers valides à traiter sont ceux présents dans le répertoire `/Import`. Une fois un fichier traité, il est supprimé.

Pour un point de mesure, la valeur de mesure du trafic est égale à la somme des valeurs correspondant à chaque type de véhicule observé :

- Un véhicule léger compte pour 1
- Un poids lourd compte pour 10
- un deux roues compte pour 0

Écrivez le logigramme de la fonction `MesureTrafic`.

Écrivez cette fonction en Java.

3.5. Affichage du trafic mesuré

L'affichage du trafic mesuré s'appuie sur l'objet `Carte`. Cet objet permet d'accéder à une collection de segments de routes et d'affecter un code d'affichage au segment en fonction de la valeur de trafic mesurée pour le point de mesure correspondant à ce segment.

L'objet `Carte` propose les méthodes suivantes :

- `GetPremierSegment()` : retourne le premier segment disponible dans l'objet `Carte`.
- `GetSegmentSuivant(idSegment)` : retourne le segment qui suit le segment ayant pour identifiant `idSegment` ou retourne une valeur nulle si `idSegment` correspond au dernier segment de l'objet `Carte`.
- `GetSegment(idSegment)` : retourne le segment dont l'identifiant est donné en paramètre. `GetSegment` déclenche une erreur de numéro 404 si l'identifiant passé en paramètre ne correspond à aucun segment de l'objet `Carte`.
- `GetSegmentPointMesure(idPointMesure)` : retourne le segment correspondant au point de mesure dont l'identifiant est donné en paramètre. `GetSegmentPointMesure` déclenche une erreur de numéro 404 si l'identifiant passé en paramètre ne correspond à aucun segment de l'objet `Carte`.

L'objet `Segment` propose les méthodes suivantes :

- `GetIdSegment()` : retourne l'identifiant unique du segment
- `GetIdPointMesure()` : retourne l'identifiant unique du point de mesure correspondant au segment.
- `ValideLocalisation(Route, Département, PR, Distance, Sens, Latitude, Longitude)` : retourne la valeur booléenne `Vrai` si les informations de localisation `Route`, `Département`, `Point de repère`, `distance au point de repère`, `Sens mesuré`, `latitude` et `longitude` correspondent au segment, `Faux` sinon.
- `GetLocalisation()` : retourne un tableau avec les informations de localisation `Route`, `Département`, `Point de repère`, `distance au point de repère`, `sens`, `latitude` et `longitude` du point de mesure attaché à ce segment.
- `Masque()` : masque le segment
- `Affiche(Couleur)` : affiche le segment dans la couleur donnée en paramètre. Les valeurs de paramètre utilisables sont données par les constantes `cc_Gris`, `cc_Noir`, `cc_Vert`, `cc_Orange` et `cc_Rouge`

Le traitement du lot de fichiers présents dans le répertoire `/Import` doit permettre de :

- produire dans un fichier journal d'erreurs la liste des points de mesure pour lesquels aucun segment n'est disponible dans l'objet `Carte`,
- produire dans un fichier journal d'erreurs la liste des points de mesure dont la localisation ne correspond pas aux informations du segment auquel le point de mesure est rattaché,
- afficher en gris les segments pour lesquels la localisation du point de mesure ne correspond pas aux informations du segment auquel le point de mesure est rattaché,

Examen professionnel de vérification d'aptitude aux fonctions de programmeur			Session 2018
Épreuve écrite : langage Java	Durée : 5h	Coefficient : 4	Page : 8/9

- afficher en noir les segments pour lesquels aucun véhicule n'a été détecté,
- afficher en vert les segments pour lesquels la valeur de mesure du trafic est supérieure à 0 et inférieure à `cmt_SeuilOrange`,
- afficher en orange les segments pour lesquels la valeur de mesure du trafic est supérieure ou égale à `cmt_SeuilOrange` et inférieure à `cmt_SeuilRouge`,
- afficher en rouge les segments pour lesquels la valeur de mesure du trafic est supérieure ou égale à `cmt_SeuilRouge`.

Pour faciliter l'exploitation du journal d'erreurs, un point de mesure en erreur ne doit apparaître qu'une seule fois dans le journal d'erreurs.

Tous les segments sont masqués au début de chaque séquence de traitement du lot de fichiers.

Écrivez le logigramme de traitement du tableau retourné par `MesureTrafic` pour afficher les données de mesure du trafic avec l'objet `Carte`

Écrivez ce traitement en Java.

3.6. Amélioration de l'affichage

La mise à jour de l'affichage de l'objet `Carte` segment par segment provoque un scintillement de l'écran qui nuit à sa lisibilité. Ne pas masquer tous les segments au début de chaque séquence ne suffit pas pour éliminer ce scintillement.

Quelle évolution proposez-vous pour résoudre ce problème ?

Précisez les méthodes des objets `Carte` et `Segment` que vous modifiez ou créez.

3.7. Passage en production

Les fonctions précédentes que vous avez prototypées doivent permettre de construire une application qui exploitera les fichiers de recueil de données trafic pour afficher en temps réel l'état de la circulation d'une zone sur une page internet accessible au public.

Quelles sont les qualités logicielles qui vous semblent importantes pour cette application ?

Quelle architecture vous semble adaptée pour répondre à ce besoin ?

Écrivez le logigramme de cette application.

Examen professionnel de vérification d'aptitude aux fonctions de programmeur			Session 2018
Épreuve écrite : langage Java	Durée : 5h	Coefficient : 4	Page : 9/9