HE2B ESI

Nom : .....

Prénom : .....

Groupe : .....

Identifiant : .....

/ 50

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

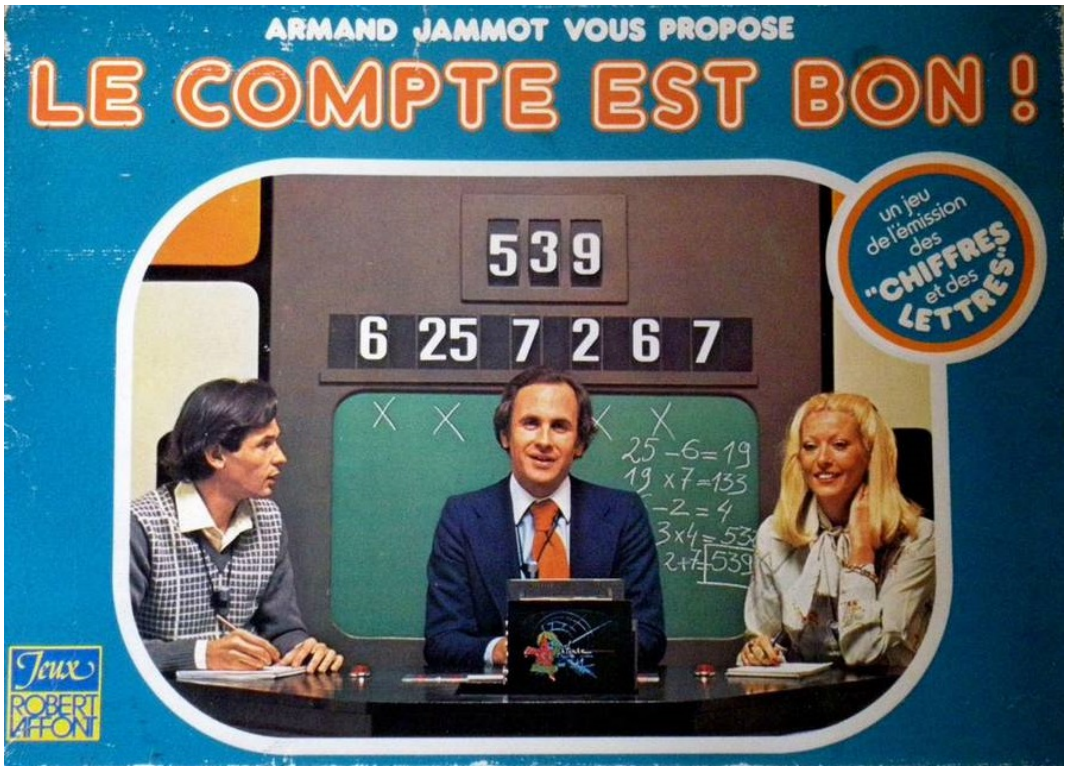
2017 – 2018

DEV1 – Développement

Examen de Janvier

Le compte est bon

janvier 2018



Le compte est bon. En 1965 Armand Jammot créa "le Mot le plus Long", la partie lettre de l'émission "des Chiffres et des Lettres" de France 2. Voyons donc comment résoudre la partie chiffres (introduite en 1972).

Voici les règles simplifiées de ce jeu.

Il se joue seul face à l'ordinateur. Il y a donc qu'un seul joueur. A l'aide de six nombres entiers différents choisis aléatoirement entre 1 et 20 par l'ordinateur et des opérations d'arithmétique élémentaire, (addition ou soustraction), il s'agit de

Version du document : 17 décembre 2017

retrouver exactement le total proposé. Le joueur dispose pour ces opérations d'un temps illimité. Le joueur n'est pas tenus d'utiliser tous les nombres et peut utiliser un nombre plusieurs fois.

Exemple 1 :

L'ordinateur propose les nombres 1 ; 5 ; 10 ; 12 ; 15 ; 20 ; et le total a obtenir est de 32

Le joueur propose 10 + 15 + 20 -12 -1

L'ordinateur répond "Bravo"

Exemple 2 :

L'ordinateur propose les nombres 2 ; 4 ; 10 ; 11 ; 16 ; 19 ; et le total a obtenir est de 31

Le joueur propose 10 + 11 + 16

L'ordinateur répond "Le total n'est pas égal à 31" "Voulez-vous faire une nouvelle proposition ? (vrai/faux)"

Le joueur répond "vrai"

Le joueur propose 11 + 11 + 11 - 2

L'ordinateur répond "Bravo"

I Algorithmique

Consignes

Pour la partie algorithmique,

- Vous travaillez sur papier, vous ne pouvez pas utiliser de notes ni d'ordinateur.
- 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.
- Veillez à travailler de manière **modulaire**.

1 Générer les 6 nombres (points)

Écrivez un algorithme :

algorithme *générerNombres*() → tableau de n entiers

qui crée un tableau `nombres` de n entiers et l'initialise avec des valeurs différentes générées aléatoirement entre 1 et 20.

Deux exemples :

— nombres = | 15 | 12 | 10 | 20 | 1 | 5 |

— nombres = | 4 | 19 | 16 | 11 | 10 | 2 |

2 Trier le tableau de 6 nombres (points)

Écrivez un algorithme

algorithme *trier*(nombres↓↑ : tableau de n entiers)

qui trie le tableau `nombres` reçu en paramètre et le renvoie trié en paramètre de sortie.

Exemple :

— Le tableau reçu en paramètre : nombres = | 15 | 12 | 10 | 20 | 1 | 5 |

— Ce même tableau trié : nombres = | 1 | 5 | 10 | 12 | 15 | 20 |

3 Générer un total (points)

Écrivez un algorithme :

algorithme *générerTotal*(nombres↓ : tableau de n entiers) → entier

qui reçoit le tableau `nombres` trié et retourne le total qui sera proposé au joueur. Ce total sera calculé comme suit :

Générer aléatoirement le nombre de nombres utilisés (entre 2 et 6) et calculer le total en

— choisissant aléatoirement dans le tableau `nombres` les nombres

— choisissant aléatoirement entre les deux opérateurs + et - pour chaque opération

Exemple :

— Le tableau reçu en paramètre : nombres = | 2 | 4 | 10 | 11 | 16 | 19 |

— génération du nombre de nombres qui composera le calcul : 3 (donc nous devons choisir 3 nombres et deux opérateurs)

— choix aléatoire du premier nombre : 16

— choix aléatoire du premier opérateur : +

— choix aléatoire du deuxième nombre : 19

— choix aléatoire du deuxième opérateur : -

— choix aléatoire du troisième nombre : 16

donnera le total 51

4 Lecture de la proposition du joueur (points)

Écrivez un algorithme

algorithme *lireProposition*() → entier

qui lit la proposition de calcul du joueur et retourne le total obtenu par cette proposition de calcul. Le joueur encodera une suite d'opérandes et d'opérateurs terminée par une valeur sentinelle '*'. Vous pouvez supposer que l'utilisateur n'encode que des bonnes valeurs, c'est à dire, des nombres compris dans les nombres proposés et des opérateurs + ou -, le tout dans le bon ordre.

Exemple :

— Le joueur propose le calcul suivant

— 11

— +

— 11

— +

— 16

— *

— l'algorithme retournera 38

5 Jouer (points)

Écrivez un algorithme :

algorithme *jouer*()

qui permet de jouer au compte est bon.

Il faudra

— créer le tableau `nombres`

— trier le tableau `nombres`

— générer le total

— afficher le tableau `nombres` et le total généré

— demander au joueur sa proposition

— si le joueur n'a pas obtenu le total proposé lui demander s'il veut continuer

— le jeu s'arrête quand soit le joueur a trouvé un calcul égal au total proposé soit quand il ne veut plus continuer.

Exemple 1 :

L'ordinateur propose les nombres 1 ; 5 ; 10 ; 12 ; 15 ; 20 ; et le total a obtenir est de 32

Le joueur propose 10 + 15 + 20 -12 -1

L'ordinateur répond "Bravo"

Exemple 2 :

L'ordinateur propose les nombres 2 ; 4 ; 10 ; 11 ; 16 ; 19 ; et le total a obtenir est de 31

Le joueur propose 10 + 11 + 16

L'ordinateur répond "Le total n'est pas égal à 31" "Voulez-vous faire une nouvelle proposition ? (vrai/faux)"

Le joueur répond "vrai"

Le joueur propose 11 + 11 + 11 - 2

L'ordinateur répond "Bravo"

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.*

6 Question préalable (sine qua non)

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.

7 Travailler dans un package (1 point)

Dans la suite de l'examen, vos classes feront partie du package `g12345.dev1.janvier`.

Vos programmes s'appelleront `LeCompteEstBon.java`, `GenerateurNombres.java`, `Calcul.java`, `LeCompteEstBonTest.java` et ils seront situés dans `~/evaluations/janvier/`. Vous placerez les versions compilées dans un dossier classes : `~/evaluations/janvier/classes`.

Écrivez ici :

— l'instruction que doit contenir votre classe `GénérateurNombres.java` pour faire partie du package demandé ;

— la commande (complète et précise) que vous allez utiliser pour **compiler** cette classe ;

— la commande (complète et précise) que vous allez utiliser pour **exécuter** cette même classe ;

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

8 Générer les 6 nombres (points)

Dans la classe `GénérateurNombres`, écrivez la méthode `GénérerNombres` que vous avez préparée dans la partie algorithmique.

Si votre programme compile, ce n'est pas pour autant qu'il est correct. Afin de le tester, on vous demande d'écrire également une méthode principale qui :

— génère et affiche 10 tableaux contenant des nombres aléatoires et différents compris entre 1 et 20.

Pour vous aider dans la méthode principale, vous pouvez utiliser toute méthode écrite pendant les laboratoires mais vous devez nous en fournir le code.

9 Trier le tableau de 6 nombres (points)

Toujours dans la classe `GénérateurNombres`, écrivez la méthode `trier`, préparée dans la partie algorithmique.

10 Générer un total (points)

Dans la classe `Calcul`, écrivez la méthode `GénérerTotal`, préparée dans la partie algorithmique.

11 Lecture de la proposition du joueur (points)

Dans la classe `LeCompteEstBon`, écrivez la méthode `lireProposition`, préparée dans la partie algorithmique.
Modifiez la méthode principale afin de tester cette méthode.

12 Jouer (points)

Toujours dans la classe `LeCompteEstBon`, écrivez la méthode `jouer`, préparée dans la partie algorithmique. Modifiez la méthode principale afin de lancer une partie complète du jeu « le compte est bon ».

13 Exceptions (points)

Modifiez la méthode `lireProposition` afin qu'une `IllegalStateException` soit lancée si l'opérande entrée n'est pas« + » ou « - ».

Adaptez la méthode principale pour qu'un message soit affiché dans ce cas.

14 Javadoc (3 points)

Écrivez la javadoc de la classe `GénérateurNombre` et écrivez ici la commande utilisée pour générer cette javadoc.

15 Les tests (2 points)

Dans la classe `LeCompteEstBonTest.java`, écrivez des tests unitaires pour la méthode `trier` en utilisant deux assert différents ainsi que des tests pour la méthode `lireProposition()`. On vous demande de vérifier que :

- la méthode `trier` trie bel et bien le tableau.
- `lireProposition` évalue correctement la proposition. Pour tester la méthode `lireProposition`, redirigez l'entrée de votre programme vers un fichier *test* qui contiendra au moins trois propositions.
  Certains changements devront être apportées à votre méthode `lireProposition`. Il peut être utile d'aller lire l'API de la classe Integer. Vous pouvez utiliser plusieurs asserts dans votre méthode de test afin de tester la réponse des différentes propositions contenues dans le fichier test. Pour vous aider voici un exemple de contenu d'un fichier test :

12
+
13
*
14
*
1
+
2
-
3
*

Écrivez ici :

— la commande permettant d'afficher le résultat des tests JUnit ;

— le contenu de votre variable d'environnement *CLASSPATH* pour pouvoir exécuter ces tests ;

— la commande permettant d'exécuter les tests en redirigeant les erreurs dans le fichier `Test.log`.