Nom : .....

Prénom : .....

Groupe : .....

Identifiant : .....

/ 50

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

2014 – 2015

Développement – 1ère

Examen de janvier

On peut jouer au loto pour un examen

Lundi 19 janvier 2015

Le Loto est un jeu de hasard assez populaire sous contrôle de l'État. En Belgique, les joueurs doivent essayer de trouver les 6 numéros différents entre 1 et 45 tirés au hasard. Pour être visuel, le tirage officiel se fait en faisant rouler 45 boules numérotées de 1 à 45.

Un numéro ne peut donc apparaitre qu'une seule fois. L'ordre dans lequel les numéros sont tirés n'a pas d'importance.

Vous allez dans cet examen écrire quelques modules pour mettre ce jeu en œuvre.

Un exemple de session que votre programme final pourrait proposer est fourni ci-dessous :

```
Entrez 6 nombres différents entre 1 et 45 :
Nombre 1 : 18
Nombre 2 : 2
Nombre 3 : 18
Entrez un autre bon nombre : 21
Nombre 4 : 0
Entrez un autre bon nombre : 5
Nombre 5 : 8
Nombre 6 : 9
Vous avez proposé : [18,2,21,5,8,9]
Le tirage était : [18,14,44,21,28,3]
Les nombres trouvés sont 18 21
Vous avez trouvé 2 nombres.
```

Il faudra notamment

1. créer et initialiser des tableaux ;
2. tirer au hasard et enregistrer les 6 nombres différents compris entre 1 et 45 ;
3. jouer, c'est-à-dire demander à l'utilisateur d'entrer 6 valeurs différentes autorisées ;

Version du document : 29 janvier 2015

4. afficher le tirage et les propositions ;
5. compter les nombres de la proposition du joueur qui font partie du tirage ;
6. En Java, on vous demandera d'assembler ces modules pour mettre en œuvre le jeu complet.

On se lance.

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.
- Les noms en gras (variables et types) doivent être respectés.
- Veillez à travailler de manière modulaire.

1 Initialisation (5 points)

Écrivez un module **initialiser** qui reçoit une taille **taille**, crée et retourne un tableau d'entiers de cette taille, dans lequel toutes les valeurs sont initialisées à la valeur 0.

On supposera que **taille** est un entier strictement positif.

2 Tirage (5 points)

Écrivez un module **tirer** qui reçoit un tableau d'entiers initialisé à zéro, **tirage** et retourne ce tableau rempli d'entiers *différents*, tirés au hasard, compris entre 1 et 45.

Les numéros seront obtenus en appelant le module **hasard(n)** qui fournit un entier aléatoire compris entre 1 et n. La difficulté de l'algorithme réside dans la recherche de nombres *différents*. Le tableau ne doit *pas* être trié.

Par exemple, si `taille = 6`, l'algorithme retournera un tableau comme `[43,5,8,12,14,32]`.

Un tel algorithme efficace est subtil. Nous ne vous demandons **pas** qu'il soit efficace, mais seulement qu'il fonctionne. Continuer à tirer un nouveau nombre si celui précédemment tiré est déjà dans la liste sera adéquat.

3 Affichage (4 points)

Écrivez un module **afficher** qui affiche sur une ligne le tableau d'entiers passé en paramètre, en séparant les valeurs par une virgule, en commençant par [et terminant par] . Par exemple, le tableau contenant les valeurs 7, 4 et 8 est affiché comme

```
[7,4,8]
```

Ce module servira tant à afficher le tableau du tirage du loto que le tableau des valeurs introduites par le joueur.

4 Entrée d'une proposition (6 points)

Écrivez un module **jouer** qui reçoit un tableau d'entiers `jeu` préalablement initialisé à zéro, lit les 6 numéros différents et valides choisis par le joueur et renvoie en sortie ce tableau `jeu` contenant ces valeurs lues et proposées par le joueur. Les valeurs entrées doivent être comprises entre 1 et une valeur maximum constante, `VALEUR_MAX` ici 45.

Si le joueur entre un numéro erroné (inférieur à 1, supérieur à `VALEUR_MAX` ou déjà entré), il devra corriger son entrée et entrer une valeur valide.

5 Évaluation de la proposition (5 points)

Écrivez un module **évaluer** qui reçoit en paramètre les tableaux **tirage** des 6 numéros tirés et **jeu** de la proposition, identifie les numéros de **jeu** qui figurent parmi les numéros de **tirage**, les affiche, et retourne le nombre de ceux-ci.

Par exemple, si **tirage** est [18,14,44,21,28,3] et **jeu** est [18,2,21,5,8,9], le module affichera : `Les nombres trouvés sont 18 21` et renverra 2. La valeur renvoyée servira pour les tests.

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 (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.

7 Travailler dans un package (1 points)

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

Votre programme s'appellera `Loto.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*.

8 Coder les algorithmes (16 points)

Écrivez les méthodes suivantes, préparées dans la partie algorithmique :

— `public static void tirer(int[] tirage)`
— `public static int[] initialiser(int taille)`
— `public static void afficher(int[] tab)`
— `public static void jouer(int [] jeu)`
— `public static int evaluer(int[] tirage, int[] proposition)`

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

1. initialiser un tirage et un jeu,
2. créer le tirage,
3. demander à l'utilisateur d'entrer les 6 nombres, en imposant éventuellement les corrections,
4. afficher le jeu proposé et le tirage,
5. afficher les nombres trouvés,
6. afficher le nombre de nombres trouvés

Un exemple de session est fourni en première page.

9 La javadoc (4 points)

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

Écrivez ici la commande utilisée pour générer cette javadoc.

10 Test JUnit (4 points)

Écrivez au moins 5 tests unitaires pour la méthode `evaluer` et exécutez ces tests en utilisant les commandes qui vous écrivez ci-dessous :

— la commande permettant de lancer les tests JUnit [1],

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

— une commande permettant de rediriger les résultats des tests dans le fichier `~/evaluations/janvier/tests.log`.

1. Inutile d'écrire ici un alias

III
Algorithmes

```
module initialiser(t ↓ : entier ) → tableau [1 à t] d'entiers
    tab : tableau [1 à t] d'entiers
    i : entier
    pour i de 1 à t faire
        tab[i] ← 0
    fin pour
    retourner tab
fin module
```

```
module tirer(tirage ↓↑ : tableau [1 à t] d'entiers)
    i : entier
    nombre : entier
    pour i de 1 à t faire
        faire
            nombre ← hasard(45)
        jusqu'à ce que NON estDansListe(tirage,nombre)
        tirage[i] ← nombre
    fin pour
fin module

module estDansListe(tab : tableau [1 à m] d'entiers, num : entier) → booléen
    estDansListe : booléen
    i : entier
    i ← 1
    estDansListe ← false
    tant que NON estDansListe ET i <= m faire
        estDansListe ← tab[i] = num
        i ← i + 1
    fin tant que
    retourner estDansListe
fin module
```

```
module afficher(tab : tableau [1 à m] d'entiers)
    i : entier
    afficher "["
    pour i de 1 à m-1 faire
        afficher tab[i] ","
    fin pour
    afficher tab[m]
    afficher "]"
fin module
```

```
module jouer(jeu ↓↑ :tableau [1 à m] d'entiers)
    constante VALEUR_MAX = 45
    i : entier
    pour i de 1 à m faire
        lire valeur
        tant que valeur<=1 OU valeur>=VALEUR_MAX OU estDans-
Liste(tab,valeur) faire
            lire valeur
        fin tant que
        jeu[i] ← valeur
    fin pour
fin module
```

```
module évaluer(tirage : tableau [1 à m] d'entier, proposition : tableau [1 à m]
d'entier) → entier
    nbBonnesValeurs :entier
    valeur :entier
    i : entier
    nbBonnesValeurs ← 0
    afficher "Les nombres trouvés sont "
    pour i de 1 à 6 faire
        valeur ← proposition[i]
        si estDansListe(tirage,valeur) alors
            nbBonnesValeurs ← nbBonnesValeurs +1
            afficher valeur
        fin si
    fin pour
    retourner nbBonnesValeurs
fin module
```

IV
Programme en java

```
1
2  package evaluations.janvier;
3
4  import java.util.Scanner;
5  import java.util.Random;
6
7  /*
8   * Loto
9   * Une version du jeu de Loto dans le cadre de l'examen d'algorithmique
10  * et laboratoire java de DEV1
11  * Janvier 2015
12  * @author npx
13  */
14 public class Loto {
15
16     public static final int VALEUR_MAX = 45;
17     public static final int TAILLE = 6;
18
19     /**
20      * @param args the command line arguments
21      */
22     public static void main(String[] args) {
23
24         int[] tirage = new int[TAILLE];
25         int[] jeu = new int[TAILLE];
26         int nbBonnesValeurs = 0;
27
28         tirage = initialiser(TAILLE);
29         tirer(tirage);
30
31         // pour tester en voyant les nombres tirés, décommenter la ligne suivante
32         //afficher(tirage);
33
34         jeu = initialiser(TAILLE);
35         jouer(jeu);
36
37         System.out.print("Vous avez proposé : ");
38         afficher(jeu);
39
40         System.out.print("Le tirage était : ");
41         afficher(tirage);
42
43         nbBonnesValeurs = evaluer(tirage, jeu);
44
45         System.out.print("Vous avez trouvé ");
46         System.out.print(nbBonnesValeurs);
47         System.out.println(" nombres");
48
49     }
50
51     /**
52      * remplit en les tirants au hasard un tableau de valeurs
53      * entières différentes comprises entre 1 et la
```

```
54          * constante VALEUR_MAX
55          * @param tab le tableau préalablement initialisé qui contiendra
56          * les valeurs tirées au hasard
57          */
58         public static void tirer(int[] tab) {
59
60             Random r = new Random();
61             int taille = tab.length;
62
63             int value;
64
65             for (int i = 0; i < taille; i++) {
66                 // int tot = 0; // permet de compter le nombre de tirages inutiles
67                 value = r.nextInt(VALEUR_MAX);
68                 while (estDansListe(tab, value)) {
69                     value = r.nextInt(VALEUR_MAX);
70                     // tot++;
71                 }
72                 tab[i] = value;
73                 // System.out.println("en " + tot + " fois : ");
74                 // afficher(tab);
75             }
76
77         }
78
79         /**
80          * crée et initialise à 0 un tableau d'entier de taille donnée
81          *
82          * @param taille , la longueur du tableau
83          * @return le tableau créé et initialisé
84          */
85         public static int[] initialiser (int taille) {
86             int[] tab = new int[taille];
87             for (int i = 0; i < taille; i++) {
88                 tab[i] = 0;
89             }
90             return tab;
91         }
92
93         /**
94          * affiche un tableau d'entiers
95          *
96          * @param tab le tableau d'entiers
97          */
98         public static void afficher(int[] tab) {
99             System.out.print("[");
100            for (int i = 0; i < tab.length - 1; i++) {
101                System.out.print(tab[i] + ",");
102            }
103            System.out.print(tab[tab.length - 1]);
104
105            System.out.println("]");
106        }
107
108        /**
109         * lit les 6 numéros valides choisis par un joueur
110         * Pour être valides, les valeurs entrées doivent être
111         * différentes et comprises entre 1 et une valeur maximum
```

```
112         * constante ici 45.
113         * @param le tableau préalablement initialisé, contenant ces valeurs.
114         */
115        public static void jouer(int[] jeu) {
116            int taille = jeu.length;
117            Scanner clavier = new Scanner(System.in);
118            int valeur;
119
120            System.out.println("Entrez 6 nombres entre 1 et 45 : ");
121
122            for (int i = 0; i < taille; i++) {
123                System.out.print("Nombre " + (i + 1) + " : ");
124                valeur = clavier.nextInt();
125                while ((valeur < 1) || (valeur > VALEUR_MAX)
126                        || estDansListe(jeu, valeur)) {
127                    System.out.print("Entrez un autre bon nombre : ");
128                    valeur = clavier.nextInt();
129                }
130                jeu[i] = valeur;
131            }
132        }
133
134        /**
135         * vérifie si le nombre num est dans le tableau tab ou pas
136         *
137         * @param tab un tableau d'entiers
138         * @param num un entier
139         * @return vrai si le nombre est dans le tableau
140         */
141        public static boolean estDansListe(int[] tab, int num) {
142            boolean existe = false;
143            int i = 0;
144            while (!existe && i < tab.length) {
145                existe = tab[i] == num;
146                i++;
147            }
148            return existe;
149        }
150
151    /**
152     * identifie et affiche les valeurs trouvées, et retourne le nombre
153     *  de celles-ci
154     *
155     * @param tirage le tableau des valeurs tirées au hasard par
156     *       l'ordinateur
157     * @param proposition le tableau des valeurs proposées
158     * @return le nombre de valeurs trouvées
159     */
160        public static int evaluer(int[] tirage, int[] proposition) {
161            int nbBonnesValeurs = 0;
162            int valeur = 0;
163
164            System.out.print("Les nombres trouvés sont ");
165
166            for (int i = 0; i < TAILLE; i++) {
167                valeur = proposition[i];
168                if (estDansListe(tirage, valeur)) {
169                    nbBonnesValeurs++;
```

```
170                     System.out.print(valeur+ " ");
171                 }
172             }
173             System.out.println("");
174 
175             return nbBonnesValeurs;
176 
177         }
178 
179 }
```