Nom : _____

Prénom : _____

Identifiant : _____ Groupe : _____

/50

HE2B ESI
Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

17 janvier 2022
DEV1
BEJ

DEV1 – Laboratoire Java

Examen de Laboratoire Java

Consignes

- L'examen dure 3 heures.
- Créez **maintenant** un projet NetBeans
  - Nom du projet : `dev1-janvier-<votreLogin>`.
  - Nom du package `gxxxxx.dev1.javal.janv2022`
  - Nom de la classe `LeJustePrix`
- La remise se fait via **gitlab** — votre professeur·e de Java vous a créé un dépôt — et nous vous demandons de faire des *commit* réguliers.

"Jeu : Le Juste Prix"
On veut simuler en version simplifiée le jeu TV "Le Juste Prix". Il est inutile de connaître le jeu, nous allons décomposer le problème pas à pas.

1 Lecture robuste d'un entier borné (3 points)

Écrivez une méthode qui permet de lire au clavier, de manière robuste, un entier compris entre 2 valeurs entières positives données min et max inclus. La méthode affiche un message d'invitation à l'utilisateur.
Ne pas oublier de gérer le cas où $min > max$, en permutant les 2 valeurs.

```
public static int lireEntierBorné(int min, int max, String message)
```

2 Génération d'un nombre aléatoire borné (3 points)

Écrivez une méthode qui permet de générer un nombre entier aléatoire compris entre 2 valeurs entières positives données min et max (les 2 valeurs comprises). Cette méthode lance une IllegalArgumentException si `max` est strictement inférieur à `min`.

```
public static int générerEntierAléatBorné(int min, int max)
```

3 Jeu : Phase 1 - Inscription des 14 candidats (3 points)

Il s'agit de créer, d'initialiser et de retourner ce tableau de 14 chaînes de caractères. Ce tableau appelé **panel** contient les noms des 14 joueurs. Ces noms seront hardcodés à votre choix.
Dans le jeu TV les candidats sont dans le public du jeu ; ici par facilité les noms des candidats sont fixés.
Écrivez une méthode qui permet de créer, initialiser et retourner un tableau de 14 chaînes de caractères.

```
public static String[] inscrireCandidats()
```

Exemple :
panel = {"J01","J02","J03","J04","J05","J06","J07","J08","J09","J10","J11","J12","J13","J14"}

4 Jeu : Phase2.1 - Choisir un candidat du panel (3 points)

Écrivez une méthode qui récupère au hasard une chaîne non `null` parmi celles du **panel** et la remplace par `null` dans le tableau **panel**. La méthode retourne la chaîne qui a été récupérée du **panel**.

```
public static String choisirCandidat(String[] panel)
```

5 Jeu : Phase2.2 - Formation du pupitre de 4 joueurs qui devront faire tourner la roue (4 points)

Écrivez une méthode qui permet de sélectionner 4 noms de joueurs différents à partir du **panel**, et qui retourne un tableau (le **pupitre**) constitué des 4 noms des joueurs sélectionnés.
La méthode `choisirCandidat()` peut être appelée plusieurs fois pour constituer le **pupitre**.

```
public static String[] formerPupitre(String[] panel)
```

Exemple :
pupitre = {"J08","J05","J01","J12"}
panel = {null,"J02","J03","J04",null,"J06","J07",null,"J09","J10","J11",null,"J13","J14"}

6 Jeu : Phase 3.1 - Vérifier qu'une estimation est valide (5 points)

Écrivez une méthode qui permet de vérifier si une estimation donnée d'un prix secret donné est valide ou pas.
Une estimation est jugée valide, si elle est positive et inférieure ou égale au prix secret.
La méthode lance une `IllegalArgumentException` dans le cas d'une estimation négative ou nulle.
Écrivez les tests unitaires de cette méthode en utilisant l'outil `JUnit`.

```
public static boolean estEstimationValide(int prixSecret, int estimation)
```

7 Jeu : Phase 3.2 - Déterminer la position de la meilleure estimation (5 points)

Écrivez une méthode qui permet de déterminer la meilleure estimation parmi toutes les estimations données. Les estimations données peuvent être valides ou pas. La meilleure estimation est celle qui est la plus proche mais inférieure ou égale au prix secret.
La méthode doit retourner la position de la meilleure estimation, ou bien -1 si toutes les estimations sont invalides.
Écrivez les tests unitaires de cette méthode en utilisant l'outil `JUnit`.

```
public static int positionMeilleureEstimation(int prixSecret, int[] estimations)
```

Exemple :
prixSecret = 155
estimations = {156,250,150,154}
posMeilleure = 3.

8 Jeu : Phase 3.3 - Déterminer l'ordre de passage des 4 candidats du pupitre pour faire tourner la roue (6 points)

Parmi les 4 joueurs du **pupitre**, il faut déterminer l'ordre de leur passage pour faire tourner la roue. Pour cela, ils doivent chacun à son tour deviner un prix secret. Ce prix secret est compris

entre deux (2) valeurs données.
Le joueur dont l'estimation se rapproche le plus de ce prix sans jamais le dépasser sera le premier dans l'ordre de passage.
On refait la même opération avec les joueurs restant et ceci en autant de fois que nécessaire, pour déterminer les 2ème, 3ème et 4ème dans l'ordre de passage.
Écrivez une méthode qui détermine l'ordre de passage des 4 candidats du **pupitre** sur la roue.
La méthode retourne un tableau des noms de joueurs du **pupitre** mais dans l'ordre de passage sur la roue.
Cette méthode lit les estimations des candidats au clavier.

```
public static String[] ordreDePassage(String[] pupitre, int prixSecret, int min, int max)
```

Indication : Les valeurs lues seront comparées au prix secret et on prend celle qui est la plus proche du prix secret sans le dépasser. En cas d'égalité, il faut reprendre la première des deux estimations.
Exemple :
pupitre = {"J08","J05","J01","J12"}
ordrePassage = {"J12","J05","J08","J01"}

9 Jeu : Phase 4 - Le finaliste (6 points)

On doit déterminer le finaliste parmi les 4 joueurs qualifiés. Pour ce faire, chacun, dans un ordre préétabli, fait tourner une seule fois une roue de 20 cases. Les cases de la roue correspondent aux multiples de 5 jusque 100 : "5, 10, etc. . ., 90, 95, 100"
Le joueur qui le premier dans l'ordre de passage obtient le score maximum est le finaliste.
Écrivez une méthode qui permet de déterminer le finaliste du jeu (la position du finaliste dans le tableau des 4 qualifiés dans l'ordre de passage).

```
public static int positionDuFinaliste(String[] ordrePassageCandidats)
```

Exemple :
pupitre = {"J08","J05","J01","J12"}
ordrePassage = {"J12","J05","J08","J01"}
résultatRoue = {75,85,60,80}
posDuFinaliste = 1, donc le finaliste est "J05"

10 Jeu : Phase 5 - Gain du finaliste (6 points)

Il faut déterminer le gain du finaliste. Pour ce faire, le finaliste (et uniquement le finaliste) dont la position est donné, doit deviner un nombre entier secret (le gain secret) donné compris entre 40000 et 50000 en 8 tentatives au maximum.
Le joueur (le finaliste) reçoit, après chaque tentative, une indication : `"votre proposition est plus haute"` ou `"votre proposition est plus basse"`.

- ▷ Si le gain secret est découvert, il sera ajouté au gain par défaut du finaliste. Le message suivant sera affiché : "Vous avez gagné **gain secret** en plus".
- ▷ Si le finaliste tombe à court de tentatives, son gain sera, comme pour les 3 autres joueurs égal au gain par défaut. Le message suivant sera affiché : "Vous n'avez pas découvert le gain secret".

À titre de rappel : le premier qualifié à faire faire tourner la roue gagne par défaut 2600, le 2ème gagne 1300, le 3ème gagne 650 et le 4ème gagne 325 par défaut.
Ce qui donne : `gainParDéfaut` = {2600,1300,650,325}
Écrivez une méthode qui permet deviner le gain secret et de mettre à jour en conséquence le tableau des gains reçu.

```
public static void gainDuFinaliste(int gainSecret, int[] gainsParDéfaut, int positionDuFinaliste)
```

Indication : Pensez au jeu de la fourchette.
Exemple :
pupitre = {"J08","J05","J01","J12"}
ordrePassage = {"J12","J05","J08","J01"}
posDuFinaliste = 1, donc le finaliste est "J05"
gainSecret = 43426 et il est découvert par le finaliste après 7 tentatives
Message : "Vous avez gagné 43426 en plus".
gainsParDéfaut mis à jour = {2600, 44726, 650, 325}
Donc le gain du finaliste le "J05" est de 44726 ; celui de "J12" est 2600 ; celui de "J08" est 650 ; et celui de "J01" est 325.

11 Jeu : Conclusion - Simulation d'une partie complète (6 points)

Nous devons simuler une partie complète du jeu "Le Juste Prix".
Dans une méthode principale, écrivez un algorithme qui réalise une simulation du jeu "Le Juste Prix".
On partira de l'inscription des candidats, en passant par la sélection des quatre (4) candidats appelés à faire tourner la roue, l'établissement de l'ordre de passage sur la roue et la détermination du finaliste. On termine par la détermination et la publication des gains du finaliste ainsi que ceux des 3 autres joueurs qualifiés.

```
public static void main(String[] args){
   /*
   – initialiser un tableau de 14 noms de joueurs: inscription des 14 joueurs;
   – déterminer le tableau des 4 joueurs qui vont faire tourner la roue;
   – générer de manière aléatoire un prix secret entre 150 et 300;
   – établir l'ordre de passage des 4 joueurs précédents en leur faisant deviner
     le prix secret précédemment généré;
   – initialiser le tableau des gains par défaut des 4 joueurs;
   – déterminer le finaliste en faisant tourner la roue aux 4 joueurs dans l'ordre;
   – générer de manière aléatoire le gain secret entre 40000 et 50000 à deviner;
   – faire deviner ce gain aléatoire par le finaliste et mettre à jour les gains ad hoc;
   – publier les noms et les gains du finaliste ainsi que ceux des 3 autres joueurs.
   */
}
```

Indication : Pensez à faire appel à toutes les méthodes implémentées précédemment.