Contrôle Machine

6 Avril 2022

Consignes :

- Le contrôle dure 2h15.
- Vous avez droit à une feuille A4, la documentation Java ainsi qu'aux TP (sujets+votre travail).
- Le contrôle doit être fait sur une machine de l'IUT, sous OS Unix.
- La communication est évidement interdite.
- Chaque exercice sera dans son propre dossier avec les classes correspondantes.
- Vos réponses doivent être envoyée à luc.dartois@u-pec.fr avec le sujet :
  [ControleMachine] votre_Prenom votre_Nom
- Vous y joindrez une unique archive s'appelant *reponse.tar* contenant les dossiers de vos exercices. Je rappelle la commande :

  ```
  tar cvf reponses.tar *
  ```

- Le non-respect des consignes exactes entraîne une perte de 2 points (vous aurez directement 0 si vous communiquez).

Exercice 1. Primalité (4 points).
On rappelle qu'on nombre est premier s'il n'est divisible que par 1 et lui-même.

1) Écrire un programme simple qui teste si l'entier positif donné en argument est premier.
A défaut de réussir à lire l'entier donné en argument, vous effectuerez le test sur une variable n de valeur 42.

2) Modifiez votre programme pour afficher la liste des diviseur de l'argument s'il n'est pas premier.

3) Enfin, améliorez votre programme pour pouvoir tester un ensemble d'entiers donnés en argument à la suite.

Exercice 2. Retour des Entiexte (4 points).
On rappelle qu'un Entiexte est une chaîne de caractères contenant un Entier représenté en bit de poids faible en premier.

1) Écrire une classe Entiexte.java permettant d'encoder les entiextes. Cette classe devra comporter :

- Un constructeur prenant une chaîne de caractères contenant un entier et.
- Un constructeur prenant un entier.
- Une fonction toString() permettant l'affichage correct d'un entiexte.

2) Créer une classe pour tester votre classe Entiexte et ses méthodes.

3) Ajoutez une fonction toInt() renvoyant la valeur d'un entiexte sous forme d'entier.

4) Écrire une fonction plusGrand(Entiexte e) retournant vrai si l'entiexte est strictement plus grand que e. La comparaison devra se faire sur les chaînes de caractères et non les entiers associés.

5) Testez vos nouvelles méthodes dans votre classe de tests.

Exercice 3. Jeux de hasard (6 points).
On souhaite avoir un outil pour simuler le hasard pour différents jeux. Il a été décidé de créer l'interface `ObjetAleatoire` décrite ci-dessous :

```
public interface ObjetAleatoire{
    void lancer();
    int lire();
    }
```

La méthode lancer modifie la valeur de l'objet selon la loi de probabilité, et lire renvoie la valeur actuelle de l'objet.

1) Créer une classe Piece réalisant l'interface ci-dessus. Vous pourrez utiliser la classe Random. On rappelle qu'une pièce peut prendre deux valeurs (ici 0 ou 1) et que les deux valeurs sont équiprobables.

2) Créer une nouvelle classe PiecePipee réalisant la même interface. La particularité d'une pièce pipée est que la loi de distribution est faussée. Elle a 70% de chances de tomber sur pile.

3) Créer dans une nouvelle classe une méthode statique genererOA() qui renvoie une pièce ou une pièce pipée de façon équiprobable.

4) Dans le main de cette classe, utilisez votre méthode statique pour générer une pièce. Sans utiliser le mot-clé `instance of`, créez une procédure permettant de déterminer si la pièce créée est pipée ou non.

5) Vérifiez votre test en utilisant `instance of`.

Exercice 4. Plan de classe (6 points).
On souhaite une application permettant de dessiner un plan de classe. L'utilisation de cette application sera via une fenêtre de la bibliothèque Swing pour une utilisation et une clarté suffisante.

1) Créez un *JComponent* qui affiche un champ de texte et contenant un grille de JLabel dont les dimensions ont été données en argument (ou 3x2 par défaut).

2) En utilisant une interface d'écoute, modifiez votre programme pour qu'en entrant entrée lors de la saisie, le texte du champ soit réinitialisé.

3) Enrichissez votre programme : appuyer sur entrée doit ajouter le nom tapé dans le champ à la première case vide de la grille.

4) Faites en sorte qu'une fois la grille complète, on ne puisse plus essayer d'en ajouter d'autres.