UFR ST Besançon - Licence Informatique - L2 - Année 2014/15

POO - Contrôle du vendredi 07 novembre 2014

Durée : 2H00 – Aucun Document Autorisé

Exercice 1. Une classe `Etudiant` (∼ 5 points)

Temps maximum conseillé : 30 minutes.

Un étudiant sera vu de manière simple comme possédant un nom et un prénom (deux chaînes de caractère) et possédant au maximum 5 notes (enregistrées dans un tableau de réels).

Une méthode `moyenne()` doit permettre de calculer et renvoyer la moyenne de l'étudiant (chacune des notes est sur 20), et la méthode `enChaine()` (équivalent en PDL++ du toString() de java) doit représenter l'étudiant sous la forme

Prénom Nom - Notes [*note1*, *note2*, *note3*, *note4*, *note5*]

comme dans l'exemple

Jean Dupont - Notes [8, 12.5, 17, 5.25, 11]

N.B. Vous devrez toujours utiliser *une boucle* pour parcourir le tableau de notes.

Enfin, la classe devra comprendre 3 constructeurs.

1. le premier constructeur n'a pas de paramètres et il définit un étudiant dont le nom et le prénom contiennent la chaîne `"non defini"`, et dont le tableau de notes n'est pas instancié ;
2. le deuxième constructeur prend en paramètres le nom et le prénom de l'étudiant (sous la forme de chaînes de caractères), et instancie le tableau de notes ;
3. le troisième constructeur prend en paramètres le nom et le prénom de l'étudiant (sous la forme de chaînes de caractères), ainsi qu'un tableau (pas forcément complet) de réels pour définir les notes de l'étudiant. **Attention.** Les notes du tableau paramètre devront être *recopiées* une à une dans le tableau attribut. Attention à bien gérer les éventuelles différences de taille entre les deux tableaux.

Attention à bien encapsuler en déclarant privé ce qui doit l'être.

Question 1. Donnez un modèle de la classe `Etudiant` sous la forme d'un diagramme UML.

Question 2. Codez la classe en PDL++.

Exercice 2. Mots de passe en nombre limité (∼ 4 points)

Temps maximum conseillé : 25 minutes.

On demande de créer une classe (nommée Password) dotée d'une méthode **statique** (nommée getPassword()) générant des mots de passe aléatoires.

L'invocation Password.getPassword() ; renvoie donc une chaîne contenant un mot de passe aléatoire.

Les mots de passe générés ont une longueur aléatoire, de 5 caractères minimum à 16 caractères maximum. Les caractères sont soit des minuscules (entre 'a' et 'z'), soit des majuscules (entre 'A' et 'Z'), soit des chiffres (entre '0' et '9').

Pour des raisons de sécurité, la classe ne peut délivrer que 100 mots de passe. Au delà, Password.getPassword() ; renvoie une chaîne vide.

N.B. L'instruction PDL++ hasard(N) renvoie un entier aléatoire compris entre 0 et $N-1$ (inclus). Par exemple, hasard(20) renvoie un nombre de 0 à 19.

N.B. En PDL++, on autorise à ajouter un nombre à un caractère. Le résultat est un caractère, correspondant à la position ainsi désignée dans la table ASCII.

Par exemple, 'a'+5 donne le caractère 'F' ; 'a'+3 donne le caractère 'D' ; '0'+3 donne le caractère '3' ; etc.

Question 1. Codez cette classe en PDL++.

Exercice 3. Dates et périodes (∼ 7 points)

Temps maximum conseillé : 45 minutes.

On considère dans cet exercice des dates et des périodes.

Une date est définie par trois entiers pour représenter le jour, le mois et l'année. *Pour simplifier le problème, on ne cherchera pas à vérifier que les dates ainsi définies sont cohérentes.* Les constructeurs permettent de créer une date soit en fournissant les trois numéros (jour, mois et année) en paramètres, soit sans paramètres (la date construite est alors celle du 1er janvier 1970).

Un *setter* doit permettre d'ajuster une date aux valeurs d'une autre date fournie en paramètre. La méthode enChaine() (équivalent PDL++ du toString() de java) doit permettre de représenter une date sous la forme JJ/MM/AAAA : jour et mois sur **deux chiffres** dans tous les cas, année sur quatre chiffres, et la barre oblique '/' comme séparateur. Exemples : "01/01/1970", "07/11/2014", etc.

Enfin, une méthode **prenant deux dates en paramètre** doit retourner la « plus petite » de ces deux dates, c'est à dire celle qui est antérieure (pas forcément strictement) à l'autre.

Question 1. Donnez un modèle de la classe Date. Attention à utiliser correctement les notions vues en cours (privé, statique, etc.), et à bien nommer les attributs et les méthodes.

Question 2. Codez la classe Date en PDL++.

On considère maintenant une *période*, qui est définie par une date de début et une date de fin. On construit une période en fournissant deux dates en paramètre, et la date de début est celle qui est antérieure à l'autre. On doit pouvoir savoir si une période est vide (ses dates de début et de fin sont les mêmes). Et une méthode doit permettre de savoir si deux périodes fournies en paramètre se chevauchent ou pas.

Question 3. Donnez un modèle de la classe Periode (mêmes recommandations qu'à la question 1).

Question 4. Codez la classe en PDL++.

Exercice 4. Questions de cours (∼ 4 points)

Temps maximum conseillé : 20 minutes.

Question 1. Dans quel cas doit-on utiliser un bloc try/catch ?

Question 2. *Les vélos sont des véhicules à 2 roues, de même que les motos. Par contre, les voitures sont des véhicules à 4 roues. Les voitures peuvent être soit des utilitaires, soit des berlines. Les camping cars sont eux aussi des véhicules à 4 roues.* Proposez une hiérarchie des classes pour les objets cités dans l'énoncé précédent.

Question 3. À quoi sert de déclarer *privé* un attribut pour lequel un *getter* permet de consulter sa valeur, et un *setter* de modifier sa valeur ?

Question 4. Dans quel cas un constructeur est-il implicite ?

Question 5. Que signifie *surcharger* une méthode ? Donner un exemple.

Question 6. Qu'est-ce qui ne va pas dans le code java suivant ?

```
public class Toto {
    private int n;

    public static int getN() {
        return n;
    }
}
```

Question 7. Et dans celui-ci ?

```
public class Titi {
    private int x;

    public Titi() {
        x = 0;
    }

    public Titi(int x) {
        if (x > 0)
            this.x = x;
        else
            this();
    }
}
```

Question 8. Qu'est-on obligé de faire quand on décide d'implanter une interface ?