Examen de rattrapage
(Durée 1h30)

Exercice 1 :

On modélise une application devant servir à l'inventaire d'une bibliothèque. Elle devra traiter des documents de nature diverse : des livres, des dictionnaires, et autres types de documents qu'on ne connaît pas encore précisément mais qu'il faudra certainement ajouter un jour (articles, bandes dessinées...). Tous les documents possèdent un numéro d'enregistrement et un titre. A chaque livre est associé, en plus, un auteur et un nombre de pages, les dictionnaires ont, eux, pour attributs supplémentaires une langue et un nombre de tomes. On veut manipuler tous les articles de la bibliothèque au travers de la même représentation : celle d'un document.

(a) Définissez les classes Document, Livre et Dictionnaire. Définissez pour chacune un constructeur permettant d'initialiser toutes ses variables d'instances.

(b) Définissez une classe Bibliothèque réduite à une méthode main permettant de tester les classes précédentes (ainsi que les suivantes).

(c) Définissez la classe ListeDeDocuments permettant de créer une liste vide de documents, puis y adjoindre une fonction permettant d'ajouter un document.

(d) Dans la classe ListeDeDocuments définissez une méthode tousLesAuteurs() qui affiche la liste des numéros des documents de la liste avec, pour chacun, l'éventuel auteur.

(e) Redéfinnisez la méthode toString() dans la classe Document ainsi que dans les classes Livre et Dictionnaire et qui renvoie une chaîne de caractères décrivant un document, un livre ou un dictionnaire. Ajoutez alors dans la classe ListeDeDocuments une méthode tousLesDocuments() qui affiche consécutivement la description de tous les documents.

(f) Proposez quelques lignes de codes à ajouter à la classe Bibliothèque afin de tester la classe ListeDeDocuments.

Exercice 2 :

Un intervalle d'entiers est défini par sa borne minimale et sa borne maximale,

par exemple [-3..7].

(a) Définir la classe ***Intervalle*** pour représenter un intervalle d'entiers avec des attributs ***private***.

(b) Ecrire le constructeur avec paramètres qui initialise l'intervalle.

(c) Ecrire la méthode ***intervalVide()*** qui retourne 1 si l'intervalle =$\emptyset$, (c-à-d min > max) et 0 sinon.

(d) Ecrire la méthode ***cardinalite*** qui retourne le nombre d'éléments de l'intervalle.

(e) Etant donnée x une valeur entière, écrire la méthode ***appartient*** qui retourne 1 si x $\in$ à l'intervalle et 0 sinon.

Soient deux intervalles I1 et I2,

(a) écrire la méthode ***inclus*** qui retourne 1 si I1$\subseteq$ I2 et 0 sinon.

Par exemple : [-2..5] $\subseteq$ [-3..8].

(b) écrire la méthode ***memeInterval*** qui vérifie si I1 est égal ou non à I2.

(c) écrire la méthode ***intersection*** qui retourne l'intervalle I1$\cap$ I2.

Par exemple : si I1 = [-3..7] et I2 = [-2..10] alors *intersection* retourne I = [-2..7].

(d) écrire la méthode ***chevauche*** qui retourne 1 si I1$\cap$ I2 $\neq \emptyset$ et 0 s'ils sont disjoints.

(e) écrire la méthode ***union*** qui retourne l'intervalle $I_1 \cup I_2$. On suppose que I_1 et I_2 chevauchent (I1 $\cap$ I2 $\neq \emptyset$).

Par exemple : si I1 = [-3..7] et I2 = [-2..10] alors *union* retourne I = [-3..10].

Bon courage