HAI715I – Ordres, treillis, induction (Examen)

30 mars 2022 - 13h-15h

Université de Montpellier – Faculté Des Sciences - Master informatique (Génie Logiciel)

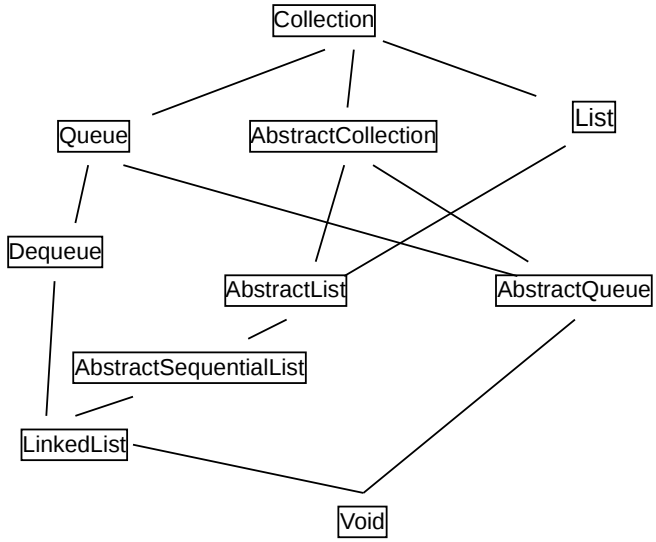
Tous les supports de cours et vos notes personnelles sont autorisés. Les ordinateurs portables, tablettes, calculatrices, etc. sont interdits. **Les 2 parties doivent être rédigées sur 2 copies séparées**.

Partie 1 : Ordres partiels et treillis

Q1. Le schéma suivant représente le diagramme de Hasse de la relation de spécialisation entre des types (classes et interfaces) de Java.

Q1.1. Indiquer quels sont les super-types de *LinkedList*, puis les super-types de *AbstractQueue*, et enfin quels sont leurs super-types communs. L'ensemble de leurs super-types communs admet-il un unique plus petit élément ou plusieurs (indiquez ce ou ces plus petits éléments) ?

Q1.2. Le diagramme de Hasse est-il celui d'un treillis ? Expliquer pourquoi. Si ce n'est pas le cas, comment pourriez-vous le compléter avec un sommet supplémentaire pour le transformer en treillis. On rappelle que la complétion ne doit pas modifier (ni ajouter, ni retirer) les relations d'ordre entre les sommets initiaux.

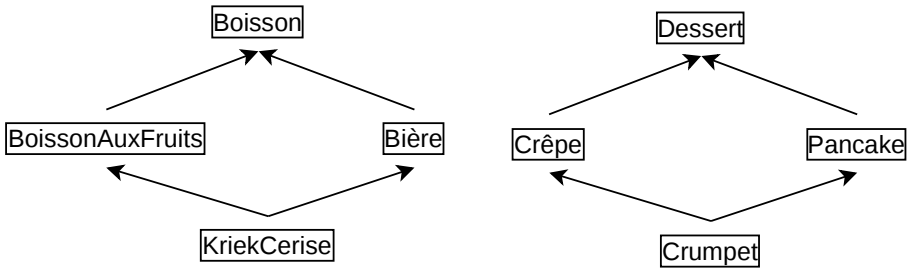


Q2. On s'intéresse à la manière dont les méthodes d'une classe accèdent aux attributs de cette même classe. On pose la relation $\mathbf{R_a}$ suivante : soient *m1* et *m2* deux méthodes, $m1 \mathbf{R_a} m2$ si *m2* accède au moins aux attributs auxquels *m1* accède.

Exemple : *m1* accède aux attributs *a* et *b* ; *m2* accède aux attributs *a*, *b*, *c* ; *m3* accède aux attributs *c* et *d*. $m1 \mathbf{R_a} m2$, mais on n'a ni $m2 \mathbf{R_a} m1$, *ni* $m1 \mathbf{R_a} m3$, ni $m2 \mathbf{R_a} m3$.

Indiquer si cette relation est : réflexive, irréflexive, symétrique, anti-symétrique, transitive, une relation d'équivalence ? un préordre ? une relation d'ordre ?

Q3. On vous donne ci-dessous le diagramme de Hasse de deux ordres partiels, respectivement $(P1,<=_{P1})$ à gauche et $(P2,<=_{P2})$ à droite.



Q3.1 Dessinez le diagramme de Hasse de leur produit direct $(P, <=_P) = P1,<=_{P1}) X (P2,<=_{P2})$.

Q3.2 Dans l'ordre $(P, <=_P)$, entourez les sommets (BoissonAuxFruits, Crêpe), (Bière, Crumpet), (KriekCerise, Crumpet). Puis appuyez-vous sur les relations qui les connectent dans $(P, <=_P)$ pour indiquer quelles sont les substitutions possibles entre les signatures d'opération suivantes :

S1 = servir(BoissonAuxFruits, Crêpe)
S2 = servir (Bière, Crumpet)
S3 = servir(KriekCerise, Crumpet)

pour les trois cas suivants :

A. Le langage demande de l'invariance sur les paramètres (ex. Java)
B. Le langage demande de la covariance sur les paramètres (ex. UML)
C. Le langage demande de la contravariance sur les paramètres (théorie des types).

Partie 2 : Induction

Exercice 1

On considère le type inductif $\mathcal{N}$des entiers naturels vu en cours.

Q1. Écrire la fonction *mult*, qui étant donnés deux entiers naturels n_1 et n_2, réalise la multiplication de n_1 par n_2. Pour ce faire, on utilisera la fonction d'addition *plus* vue en cours. La définition inductive de cette fonction devra se faire par rapport au premier argument n_1.

Q2. Démontrer que : *(mult 2 n) = (plus n n)*, pour tout entier naturel *n*.
A-t-on besoin de réaliser une induction structurelle pour faire cette preuve ?

Q3. Démontrer que : *(mult n 2) = (plus n n)*, pour tout entier naturel *n*.
A-t-on besoin de réaliser une induction structurelle pour faire cette preuve ?

Exercice 2

On considère le type inductif $\mathcal{N}$des entiers naturels vu en cours.

Q1. Définir la relation inductive *is_fact* qui spécifie le comportement de la fonction factorielle. Cette relation prendra deux arguments : l'entier dont on veut calculer la factorielle et le résultat.

Q2. Démontrer que : *(is_fact 3 6)*.

Q3. Écrire la fonction factorielle. On appellera cette fonction *fact*.

Q4. Écrire le schéma d'induction fonctionnelle correspondant à la fonction *fact*.

Q5. Démontrer la correction de la fonction *fact* vis-à-vis de la relation *is_fact*.