Cours PROG : Introduction à la programmation
L3 informatique, parcours Recherche et innovation
Examen final, 15 décembre 2015

Luc Bougé, Martin Quinson

Durée : 1 h 50 + 10 mn de relecture

Cette épreuve écrite a pour objectif de faire la synthèse de l'ensemble des sujets étudiés en cours. La rapidité n'est pas un critère majeur d'évaluation. Au contraire, l'accent est mis sur le soin, la précision, la pédagogie, l'inventivité, la capacité à bien exploiter l'ensemble des documents que vous avez apportés, etc. Vous êtes invités à utiliser l'ensemble des documents disponibles : notes de cours, polycopiés, livres, etc. Seuls les PC et Internet ne sont pas les bienvenus.

La forme de l'épreuve est inspirée de celle de l'épreuve de leçon à l'Agrégation. Vous recevez un sujet et vous imaginez que vous avez à faire un cours de 2 heures à une classe de L3 sur ce sujet. Que proposeriez-vous ? Comme pour la préparation d'un exposé, un temps de relecture est prévu pour la correction de l'orthographe et le soin de la présentation. Ces aspects seront notés sur 20% du barème. Il est attendu deux productions.

Un plan de cours, qui devrait tenir sur 1 page, 2 éventuellement, sûrement pas plus. Ce plan n'est pas une dissertation, c'est plutôt une liste structurée de points à aborder dans ce cours, avec quelques mots pour chacun des points.
Vous pouvez imaginer par exemple que c'est ce que vous auriez préparé sur vos notes de cours, ou ce que vous écririez au tableau ou sur vos diapos. Pas de longues phrases, mais un style télégraphique comme dans les diapos de soutenance.

Un point développé, choisi par vous, pour lequel vous rédigez soigneusement ce que vous proposez de dire aux étudiants. La longueur recommandée est 2 pages, 3 éventuellement, sûrement pas plus.

Critères d'évaluation des productions

Plan : 1–2 pages, pas plus
- **Complétude** de la couverture des aspects traités
- **Organisation** de l'exposé, graduation, équilibre
- **Précision** de la rédaction, soin, orthographe
- **Qualité** des exemples, pédagogie, originalité, inventivité

Développement : 2–3 pages, pas plus
- **Intérêt** du point choisi pour le développement
- **Organisation** du développement, pédagogie
- **Précision** de la rédaction, soin, orthographe
- **Qualité** des exemples, pédagogie, originalité, inventivité

Sujets proposés au choix

Il vous est demandé de choisir l'un des deux sujets et d'y consacrer toute votre attention. Si vous traitez plus d'un sujet, seul le premier sera corrigé.

1 Programmation fonctionnelle

Le cours a été constitué de deux parties qui se sont toutes les deux appuyées sur des langages fonctionnels, Caml et Scala. Il est donc intéressant aujourd'hui de réfléchir à ces deux approches de la programmation fonctionnelle et de les comparer. C'est d'autant plus intéressant que Caml est l'aboutissement d'une longue tradition de langages fonctionnels de plus en plus typés (Lisp, Scheme, etc.) alors que Scala est le résultat de l'introduction tardive d'un aspect fonctionnel dans une longue tradition de langages impératifs de plus en plus orientés vers la programmation à objets (C, C++, Java, etc.)

Principes fondamentaux. Présentez la notion de programmation fonctionnelle et montrez comment elle se distingue de la programmation impérative. Discutez le statut des structures de contrôle et des affectations dans cette approche.

Mise en oeuvre dans Caml et Scala. Comparez les syntaxes proposées par Caml et Scala pour définir et manipuler les fonctions. Détaillez en particulier le cas des fonctions récursives et des ensembles de fonctions mutuellement récursives.

Ordre supérieur dans Caml et Scala. Présentez la notion d'ordre supérieur et donnez quelques exemples typiques en Caml et Scala. Discutez la notion de curryfication et de fonction partielle dans les deux approches. Présentez l'utilisation de l'ordre supérieur pour la définition de *continuations* en Caml ou de *futurs* en Scala. Comparez avec d'autres langages que vous pouvez connaître.

Fonctionnel vs itératif : le cas de la récursion terminale. Présentez la notion de récursion terminale et l'enjeu de son optimisation par quelques exemples typiques en Caml et en Scala. Détaillez les optimisations possibles, même si elles ne sont pas forcément implémentées. Comparez avec d'autres langages que vous pouvez connaître.

2 Programmation à objets

La seconde partie du cours a été consacrée à l'étude de la programmation à objets en appui sur le langage Scala. Par son aspect fonctionnel, Scala se rapproche de Caml, même si sa synthèse de types reste plus limitée par rapport à Caml. Il est donc intéressant de réfléchir à la gestion du typage en Scala en particulier au niveau de son interaction avec la manipulation des classes et des objets.

Principes fondamentaux. Caractérisez la programmation à objets au travers des différents concepts fondamentaux de cette approche et rappelez pour chaque concept sa syntaxe en Scala. Discutez également l'intérêt et présentez la syntaxe des notions de *paquets*, *modules*, et *exception*. Discutez les interactions entre le mécanisme de polymorphisme et la variance des paramètres.

Types et classes. Comparez les avantages et inconvénients de ces deux notions, en donnant des exemples typiques d'usage. Quand et comment peut-on combiner programmation à objets et programmation fonctionnelle ? Quel est le rapport avec la mutabilité des données ?

Héritage, interfaces et polymorphisme. Présentez comment ces notions permettent de factoriser du code au sein d'une application en vous appuyant sur des exemples. Discutez les notions de *co-variance*,*non-variance*, *contra-variance* et *sous-typage*. Étendez votre comparaison au cas des types génériques. Rappelez l'intérêt de l'héritage multiple et le problème que pose cette notion, puis discutez la solution proposée par Scala par rapport à d'autres langages que vous pouvez connaître.

Composer des types, des objets ou des fonctions. Comparez les mécanismes pour déclarer de nouveaux types et ceux de typage automatique d'expressions en Caml et Scala. Discutez des grands principes d'organisation du code et styles de programmation classiques avec des objets ou en fonctionnel.