Master Miage
Programmation Avancée

Décembre 2016, durée: 2H00, Feuille manuscrite A4 autorisée. Les exercices et les questions sont de difficulté variable, le barème n'est qu'indicatif. Les questions d'un même exercice doivent être traitées séquentiellement et justifiées.

1 Questions en vrac (4 points)

Question A:
En Java, le passage de paramètre est-il par référence ou par copie ?
Question B:
Quels sont les intérêts d'utiliser une JVM ?
Question C:
Quels sont les inconvénients d'utiliser une JVM ?
Question D:
Qu'est-ce que le JIT ?

2 Mécanisme de persistance (16 points)

Le but de cet exercice est de développer un mécanisme de persistance (i.e. permettant de sauvegarder l'état d'un objet) basé sur les annotation.
Question A:
Sur quels éléments peut porter une annotation?
Question B:
Donnez deux exemples d'annotations standards
Question C:
À quoi sert l'annotation @Retention ?

Le principe de notre mécanisme d'annotation est le suivant :

- Les classes autorisant la sauvegarde seront annotées par *@SaveObject*
- Les attributs pouvant être sauvegardés seront annotés par *@SaveField*

Un exemple de classe utilisant ce mécanise est donné ci-après.

```
@SaveObject
public class MaClasseAMoi {
   @SaveField
```

```
    public int toto;

   @SaveField
   private double titi;

   protected long nombre;

}
```

Question D:
Dans la classe *MaClasseAMoi*, quels sont les attributs qui seront effectivement sauvegardés ?

Question E:
Écrivez le code des annotations *@SaveObject* et *@SaveField*.

On souhaite maintenant développer deux classes permettant de sauvegarder (*Save*) et charger (*Save*) un objet. Un exemple d'utilisation de la classe *Save* est donné ci-dessous :

```
....
MaClasseAMoi ma = new MaClasseAMoi(....);
...
//on sauvegarde vers un fichier sur le disque
Save s = new Save(new FileOutputStream(....));
s.writeObject(ma);
```

Question E:
De quelle classe devra hériter *Save* pour rester dans la philosophie des I/O Java.

Question F:
Écrire un squelette de classe (attributs, constructeurs, méthodes) sans l'implémentation de la méthode *writeObject*.

Nous allons maintenant nous intéresser au fonctionnement de la méthode *writeObject*.

Question G:
Quel mécanisme permet en Java d'aller étudier la structure d'un objet ou d'une classe ?

Question H:
Ecrire le code de la méthode *writeObject* en supposant qu'on ne sauvegarde que des attributs de type primitif.

Question I:
Comment étendre ce mécanisme pour sauvegarder un attribut de n'importe quel

type ?

Question J:
On veut pouvoir remplacer la valeur d'un attribut par une constante lors de la sauvegarde. Par exemple quelle que soit la valeur de *toto*, la remplacer par 5 lors de la sauvegarde. Comment feriez vous ?

3 ClassLoader (4 points)

Question A:
Quel est le rôle du *ClassLoader* en Java ?

Question B:
Comment construit-on son propre ClassLoader en Java2 ?

Question C:
Donnez une architecture utilisant votre propre ClassLoader permettant de charger plusieurs fois la même classe dans la JVM.

Question D:
Comment avoir son propre *ClassLoader* permet-il de décharger des classes de la JVM ?