INFO-H-301 : Préparation à l'examen

Sources

- http://oraux.pnzone.net/list-30-3-s6-l4-0000.html
- L'orienté objet, Hugues Bersini.

Question 1 : Héritage / Compatibilité de types.

Une classe A contient une méthode faireA(), une classe B contient une méthode faireA() et faireB(). B hérite de A.

```
A a = new B();
B b = new A();
a.faireA();
a.faireB();
b.faireA();
b.faireB();
```

Expliquer ce que fait ce code. S'il y a des erreurs, détailler.

Mots clés :

- Type variable / type objet pointé
- Compatibilité des types avec l'héritage
- Héritage et encapsulation (modifiers)
- Héritage et exceptions
- La sous classe doit remplir le contrat de la superclasse. Si on redéfinit une méthode publique, elle ne peut pas être private. Si on redéfinit une méthode privée, elle peut être publique.
- Lors de la redéfinition d'une méthode qui déclenche des exceptions, on ne peut pas envoyer plus d'exception que la méthode redéfinie.
- Voir exercice du TP plus complet (pour le casting).

Question 2 : Exceptions

Expliquer comment implémenter un mécanisme d'exceptions.

Mots clés :

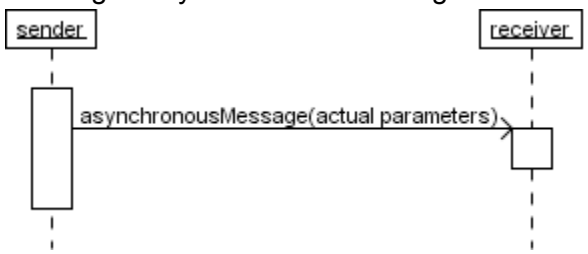
- throw
- throws
- try/catch
- throwable ← exception
- exceptions et redéfinition de méthodes

Question 3 : Threads

Expliquer comment faire tourner du code dans un Thread (deux méthodes possibles). Comment représenter des threads à l'aide d'un diagramme UML ?

Mots clés :

- Première méthode : sous-classer Thread
- Seconde méthode (préférée) : implémenter un Runnable et le passer à un Thread
- Blocs synchronized
- Interblocages
- Messages asynchrones dans diagramme de séquences :



- Synchrone = flèche pleine.

Question 4 : Mémoire Java et Garbage Collector

Expliquer comment Java gère la mémoire. Illustrer à l'aide d'un schéma de mémoire (référents, objets, …).
Quelles sont les avantages et inconvénients de la gestion mémoire par pile par rapport au heap?
Comment invoquer le Garbage Collector ? Est ce que le fait de mettre un référent à *null* a pour effet de supprimer l'objet pointé de la mémoire ?

Mots clés :

- Référents / objets
- Pile = allocation/désallocation mémoire prévue à la compilation. Plus efficace.
- Heap = allocation/désallocation dynamique. Moins efficace.
- Aller voir dans le livre pour les détails (Chapitre 9 "question de mémoire").
- http://fr.wikipedia.org/wiki/Allocation_de_m%C3%A9moire
- Garbage Collector
    - Efface régulièrement les objets qui ne sont plus pointés.
    - On peut l'invoquer avec System.gc()
    - Mettre un référent à *null* n'a pas nécessairement pour effet de supprimer l'objet (il peut être pointé par un autre référent et le Garbage Collector ne passe pas tout le temps).

Question 5 : Tableaux et référents

En java, quand on crée les tableaux suivants, combien de référents sont créés en réalité?

- `A[][] matrice = new A[3][2] ;`
- `int[][] matrice2 = new int[3][2] ;`
- `int[] vect = matrice2[0];`

Question 6 : Comparaison d'objets

```
A a1 = new A(12) ;
A a2 = new A(12) ;
```

Que signifie (`a2 == a1`) ?

Expliquer comment comparer des objets.

Mots clés :

- == compare les référents.
- equals(obj) par défaut compare les référents.
- on peut redéfinir equals(obj)
- comparaison en surface / profonde.
- Equals objets must have equals hashCodes.
- http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html#equals%28java.lang.Object%29
- http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html#hashCode%28%29

Question 7 : Diagramme de séquence et destruction d'objets.

Dessiner un diagramme de séquences avec création et destruction d'objets.
Expliquer comment cela se traduit en Java et en C++ ?

Question 8 : Types primitifs et types non primitifs.

Soit le bout de code suivant :

```
public class A
{
  private int a;
  public A(int a)
  {
    this.a=a;
  }
  public void IncA()
  {
```

```
    a++;
  }
}

public class Principale
{
  public static void Inc(int b)
  {
    b++;
  }
  public static void main(String []args)
  {
    int b = 5;
    A a = new A(b);
    a.IncA();
    Inc(b);
  }
}
```

Que vaut l'entier b à la fin ?

Mots clés :

- Types primitifs / non primitifs (copie / référence).

Question 9 : Copie / Clonage d'objets

Soit 2 classes A et B, A est associé à un B. On instancie un objet de type A. Comment faire une copie de cet objet ? Quels problèmes peuvent survenir ? Expliquer ce qu'il se passe en mémoire lors de cette copie.

Mots clés :

- clone()
- comment redéfinir clone ?
- en surface / en profondeur
- pourquoi clone() est protected dans Object ?
- interface Cloneable
- Lire chapitre 14 du livre et http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html#clone%28%29

Question 10 : Erreurs et gestion mémoire

Une classe B hérite d'une classe A.
Soit le code suivant :

```
void static main()
{
  A a1 = new A();
  A a2 = new A();
  B b1 = new B();
  B b2 = new B();

  a1 = a2; //(1)
  b1 = a2; //(2)
  a1 = b1; //(3)
  b1 = b2; //(4)
}
```

Que se passe-t-il à la compilation et à l'exécution ?

Mots clés :

- erreurs compilation / exécution
- gestion de la mémoire

Question 11 : UML <-> Java

La classe A est composée d'objets B (1-infini) et les objets de la classe B envoient des messages aux objets A. Ecrire le diagramme de classes et un diagramme de sequences illustrant cela. Ecrire le code Java de ces deux classes.

Les classes C1 et C2 héritent de la classe B. Les objets B, C1 et C2 répondent de la même façon au message faire(). Modifier votre code Java pour implémenter ce comportement.
Que se passe-t-il si un objet de la classe C2 s'envoie un message faire() à lui même ? Illustrer à l'aide d'un diagramme de séquences.