# INF7845 — Principes avancés des langages à objets
# Examen Intra
## Mercredi 24 février 2010 — Durée 3 heures

## 1 Exercice 1 (20 points)

Soit le programme pseudo-Java suivant

```
abstract class A {
  int x;
  void foo() { x++; }
}

interface I {
  int bar();
}

class B extends A implements I {
  int y;
  @override void foo() { y++; }
  B() { x = y = 0; }
  int bar() { return y - x; }
}

class Main {
  static void main() {
    B b = new B();
    System.out.println(b.bar());
    b.foo();
    System.out.println(b.bar());
    A a = b;
    a.foo();
    System.out.println(b.bar());
  }
}
```

1. Identifiez les différentes propriétés globales et locales de ce programme.
2. Précisez pour chacune d'elles les relations avec les classes et les autres propriétés.
3. Quel est la nature de la fonction `main` de la classe `Main` ? Précisez ces relations avec les classes et les propriétés.
4. Qu'affiche la fonction `main` de la classe `Main` ? Justifiez votre réponse.

## 2 Exercice 2 (40 points)

Soit un jeu vidéo médiéval fantastique. On a demandé à un développeur de développer la gestion des personnages dans une bibliothèque autonome.

Les personnages du jeu vidéo ont un nom et des points de vie et un métier. Une fois une personnage crée, il ne peut plus changer de nom ou de métier. Le guerrier peut s'équiper d'une arme des toutes les catégorie (contondantes, tranchantes et perforantes). Le prêtre ne peut s'équiper que d'une arme contondante. Il possède des points de mana et peut guérir des personnages (1 point de mana guérit 1 point de vie). Le paladin peut faire tout ce que les guerriers et les prêtres peuvent faire.

Proposez une modélisation qui permette à un client de la bibliothèque de :

- Créer un nouveau personnage avec un nom, un métier, des points de vie et, si applicable, des points de mana.
- Permettre à un personnage de se présenter (on récupère une chaîne du type `"Je suis %s le %s"`, par exemple `"Je suis Grunt le guerrier."`)
- Obtenir et modifier le nombre de points de vie d'un personnage.
- Obtenir l'arme du personnage et permettre à un personnage de s'équiper d'une arme ou de s'en déséquiper.
- Permettre au prêtre et au paladin de soigner un autre personnage.

1. Justifiez votre modélisation.
2. Illustrez-la avec un diagramme de classe.
3. Implémentez-la en pseudo-Java. Vous pouvez utiliser des caractéristiques non présentes en Java comme l'héritage multiple si vous le précisez.

# 3 Exercice 3 (20 points)

Un programeur fatigué à construit la modélisation suivante mais n'a pas compilé le code source.

```
public class Moteur {
  public void mettreDuCarburant(Carburant c);
  public void allumer() { ... }
  public void eteindre() { ... }
  ...
}
public class Voiture extends Moteur {
  @override public void mettreDuCarburant(Essence e) { ... }
  public void demarrer(Cle cle) { if (serrure.accepte(cle) {allumer();} }
  private Serrure serrure;
  ...
}
public interface Carburant { ... }
public class Essence implements Carburant { ... }
...
```

Identifiez les différentes erreurs de modélisation. Pour chacune des erreurs :
- expliquez-la en détail ;
- écrivez un morceau de code client illustre/exploite l'erreur ;
- proposez une solution valide qui répare l'erreur.

# 4 Exercice 4 (20 points)

Soit le programme pseudo-Java suivant

```
class Queueleuleu<O extends Oiseau> {
   ArrayList<O> list = new ArrayList<O>();
   O get(int i) { return list.get(i); }
   int size() { return list.size(); }
   void nouveau() { O n = new O(); list.add(n); }
}
interface Oiseau { String cri() {...} ... }
class Canard implements Oiseau { String cri() { return "Coin!"; } ... }
class Oie implements Oiseau { ... }

class Main {
  static void main() {
    Queueleuleu<Canard> queue = new Queueleuleu<Canard>();
    queue.nouveau();
    System.out.println(queue.size()); // Outputs 1
    System.out.println(queue.get(0).cri()); // Outputs "Coin!"
  }
}
```

1. L'instruction

   ```
   Queueleuleu<Oiseau> queue = new Queueleuleu<Canard>();
   ```

   est-elle statiquement valide ? Justifiez votre réponse.
2. Expliquez pourquoi dans la plupart des langages à objets, il n'est pas possible de faire

   ```
   new O();
   ```

   lorsque O est un type formel générique.
   Justifiez et illustrez votre réponse.
3. Trouvez les différents moyens de récrire le programme pour que ça marche sachant que **vous devez respecter l'API** : vous n'avez pas le droit de changer le corps du `main`. Étayez votre raisonnement.