UNIVERSITE LIBRE DE BRUXELLES — Lundi 10 juin 2013
Faculté des Sciences Appliquées

INFO-H-200 : Programmation orientée objet
Examen

Remarques préliminaires

- On vous demande de répondre à **chaque question sur une feuille séparée**.
- N'oubliez pas d'inscrire votre nom, prénom et numéro de matricule sur chaque feuille.
- Vous disposez de 3 heures et vous ne pouvez pas utiliser de notes.
- La réponse à la question doit comprendre, si approprié, le code *Python* structuré et conforme aux règles de bonne pratique et conventions ainsi que des commentaires pertinents.
- Vous pouvez ajouter des fonctions si cela vous semble nécessaire.
- Sauf mention contraire, vous ne pouvez utiliser aucune fonction de librairies (pas d'`import`).

Question 1

Pour chaque affirmation suivante, indiquez si elle est vraie ou fausse dans le cadre de la programmation orientée objet.

- Dans un diagramme de classe UML, si une flèche pointe de la classe A vers la classe B (quel que soit le type de flèche), cela signifie que le code dans le fichier de A dépend du code dans le fichier de B. En d'autres termes, si B change, il faudra au minimum vérifier A, et probablement le changer également. Par contre (en supposant qu'il n'y a pas de flèche dans l'autre sens), si A change, il n'est pas nécessaire de vérifier B.
- Dans une architecture orientée-objet, pour respecter la notion de cohérence forte, on souhaite généralement avoir des groupes de classes qui sont très liées entre elles, formant pratiquement un graphe complet bidirectionnel, c'est-à-dire que chaque classe a une flèche vers et depuis chaque autre classe.
- Il vaut toujours mieux dépendre directement d'une classe plutôt que d'inventer une interface intermédiaire, parce que ça économise la création d'un fichier.
- Une bonne architecture est une architecture dans laquelle on peut "tracer des lignes", c'est-à-dire qu'il est possible, sur le diagramme de clase UML, d'isoler des parties du système en traçant une ligne qui n'est jamais traversée que dans un seul sens par les flèches.

Question 2

Le code suivant contient des erreurs. Pour chaque erreur, indiquez quand elle aura lieu : à la compilation ou pendant que le programme tourne. Dans ce dernier cas, indiquez si, avec la fonction `main` fournie, le problème se produira. Pour chaque erreur, indiquez une ou deux manières de le résoudre : la manière qui implique le moins de changements possibles dans le code, et, si elle est différentes, la manière que vous considèreriez correcte d'un point de vue orienté-objet.

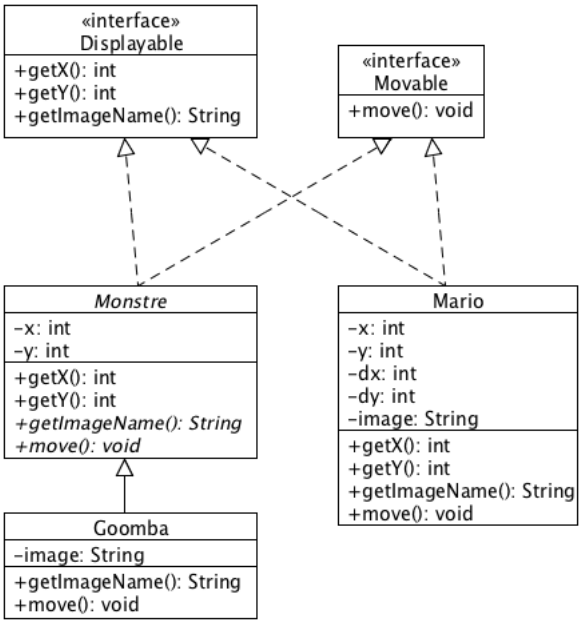
Indiquez également ce que le programme correct aurait dû afficher (et affichera avec vos corrections).

```
public class Main {
    public static void main(String[]) {
        Monster g = new Gobelin();
        Monster h = new Hero(2, "Warrior");
        System.out.println("Le combat commence!");
        System.out.println("\t" + h + " affronte " + g);
        while (g.isAlive()) {
            h.attack(g);
            System.out.println("Le combat n'est pas fini!");
            System.out.println("\t" + h + " affronte " + g);
```

```
        }
        System.out.println(h.toString() + " a vaincu son adversaire !");
    }
}

abstract class Monster {
    private int pv;
    public Monster(int pv) {
        this.pv = pv;
    }
    public abstract void receiveDamage(int a) {
        this.pv -= a;
    }
    public abstract boolean isAlive();
}

class Gobelin extends Monster {
    public Gobelin() {
        super(4);
    }
    public boolean isAlive() {
        return this.pv > 0;
    }
    public toString() {
        return "Un Gobelin sans nom, avec " + this.pv + "point(s) de vie.";
    }
}

class Hero {
    private String name;
    public Hero(int damages, String name) {
        this.damages = damages;
    }
    public void attack(Monster m) {
        m.receiveDamage(damages);
    }
    public String toString() {
        return this.name;
    }
}
```

Question 3

Ecrivez le code java équivalent au diagramme de classe UML suivant :



Question 4

Tracez le diagramme de classe UML représentant le code présenté à la question 2 (votre diagramme ne doit pas reprendre les classes de la librairie standard).

Question 5

Dans le cadre d'un jeu de type *Asteroid*, le programme doit gérer un certain nombre de types d'éléments. A ce stade de la conception, on doit gérer un vaisseau (classe `Ship`), des rochers (classe `Rock`) et des missiles (classe `Missile`). Dans les versions suivantes du programme, on s'attend à devoir ajouter des vaisseaux ennemis et des bonus. L'objet principal du modèle est de la classe Game, et sert essentiellement à contenir des références vers tous les objets du jeu.

Pour détecter les collisions, l'équipe de développement propose les deux solutions suivantes :

– L'objet Game pourrait avoir un attribut `Ship` pour le joueur, un attribut `List<Rock>` pour les rochers et un attribut `List<Missile>` pour les missiles. Il suffit alors d'avoir, dans Game, une méthode surchargée pour les deux types de collisions (le vaisseau est immunisé contre ses propres tirs) avec les signatures suivantes : `void collide(Ship s, Rock r)` et `void collide(Rock r, Missile m)`.
– Créer une interface `Collidable` avec une unique méthode `boolean collides(Collidable c)` ; implémenter cette interface dans chaque classe représentant un type d'élément du jeu, et n'avoir pour les collisions qu'une seule liste de type `List<Collidable>` dans Game.

Notez qu'on parle bien à ce stade de *détecter* les collisions, pas de gérer les conséquences.

Laquelle leur recommanderiez-vous ? Justifiez.