Examen – AO (Approche Objet)

Responsable : Pr. Xavier Blanc – Université de Bordeaux 1 (Xavier.Blanc@labri.fr)
Le 17 décembre 2013, 14h – 17h
3 pages - Tous documents autorisés

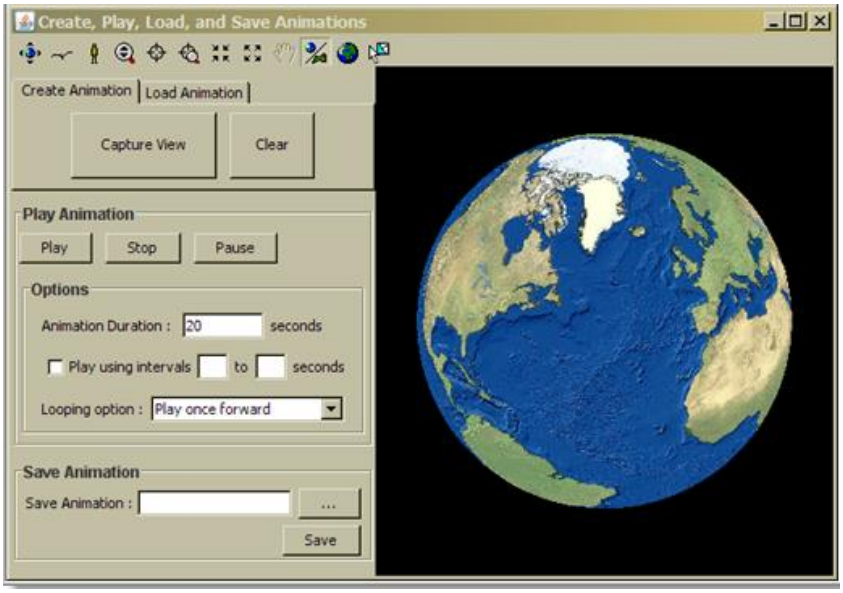
Questions de cours (2 points) :

Q1 Les relations d'héritage doivent-elles être comptées pour mesurer le couplage ? Expliquez pourquoi.

Q2 En java, expliquez l'impact des packages pour le couplage (deux classes dans un même package sont-elles couplées ? Peut-on mesurer le couplage entre packages ? etc.).

Questions Interface Graphique (3 points)

L'application suivante permet de créer des animations.



Q1 Décrivez (à l'aide d'un arbre) la structure de cette interface graphique (objets graphiques + layout)

Q2 Lorsqu'on clique sur le bouton « Save » l'animation est sauvée. On considère que la classe Animation est fournie et qu'elle possède une méthode save(File f) qui permet sa sauvegarde. Donner le pseudo-code permettant la gestion du clic sur le bouton Save (introduisez toutes les classes que vous jugez nécessaires).

Questions conception objet (7 points) :

Une jeune équipe de développeurs Java a développé le noyau d'un site de gestion des notes des étudiants. StudBX est un noyau relativement classique qui permet la gestion simple des UE et des étudiants (par des clés) ainsi que la gestion des examens. La figure suivante montre la première version de StudBX proposée par les jeunes développeurs. Cette version est réalisée en une seule classe.

```
public class StudBX {
    HashMap<String , Integer > ues; //String pour keyUE, Integer pour le coef de l'UE
    List<String> etus; //liste des keyEtu

    //String pour keyExam
    // HashMap<String , Integer> pour la note de l'Etu (String keyEtu, Integer note)
    HashMap<String , HashMap<String , Integer>> examEtuNote;

    //Une UE peut organiser plusieurs exams
    //String pour keyEU, List<String> pour la liste des keyExam
    HashMap<String , List<String>> euExams;

    public void modifierUE(String key , Integer coef) {
        if (ues.containsKey(key)) {
        retirerUE(key);
        ues.put(key, new Integer(coef));
        }
    }

    public void retirerUE(String key) {
      ues.remove(key);
    }

    public String creationExam(String ue) {
      List<String> exams = euExams.get(ue);
      if (exams == null) {
        exams = new LinkedList<String>();
        euExams.put(ue, exams);
        }
        int i = exams.size()+1;
        String examKey ="exam_"+i;
        exams.add(examKey);
        examEtuNote.put(examKey, new HashMap<String,Integer>());
        return examKey;
    }

    public void saisirExamEtuNote(String examKey , String etuKey, int note) {
        //TODO Q1
    }

    public int calculerMoyenneExam(String examKey){
        //TODO Q2
        return 0;
    }
}
```

Q1 Ecrivez le code qui consiste à saisir la note obtenue à un examen par un étudiant //TODO Q1

Q2 Ecrivez le code qui consiste à calculer la moyenne générale obtenue à un examen //TODO Q2 (retirer le return 0).

Q3 Expliquez pourquoi la cohérence de cette classe n'est pas bonne.

Q4 On veut maintenant faire apparaitre les classes UE et Etudiant afin d'améliorer la cohérence de cette application. Proposez votre nouvelle conception avec ces nouvelles classes (vous pouvez proposer d'autres classes).

Q5 On veut pouvoir sauvegarder sur disque dur les UEs, les étudiants et les notes pour pouvoir tout recharger facilement. Mettez en place cette fonctionnalité.

Q6 Un étudiant a une moyenne générale qui correspond à la moyenne des notes obtenues à tous les examens pondérées par les coefficients des UEs. On considère que cette fonction « int moyenne() » est déjà codée et qu'elle a été rajouté à la classe Etudiant (cf. Q4). Les enseignants utilisateur de cette application veulent savoir si la moyenne d'un étudiant est « Inconnue », « entre 0 et 8 », « strictement inférieure à 10 mais au-dessus de 8 », « entre 10 et 16 », « supérieure à 10 ». « au-dessus de 14 ». Supportez cette demande en vous aidant d'un pattern vu en cours (vous pouvez ajouter tout ce qui vous semble nécessaire).

Tests (4 points)

Rappelez-vous que l'objectif d'un testeur est de trouver des fautes !

Q1 On veut tester la fonction « saisirExamEtuNote(String examKey , String etuKey, int note) » de l'exercice de conception objet. Proposez vos tests fonctionnels (donnée en entrée et résultat attendu) en expliquant vos classes d'équivalence.

Q2 Le code de la figure suivante est utilisé dans un jeu relativement simple. Réalisez les tests boîte blanche de ce code (n'oubliez pas les exceptions !).

```
public boolean checkGame(String source , int[] seq , int i) {
    boolean good = true;

    if (source == "r") {
        if (seq[i]!=0) good = false;
    } else if (source == "b") {
        if (seq[i]!=1) good = false;
    } else if (source == "y") {
        if (seq[i]!=2) good = false;
    } else if (source == "g") {
        if (seq[i]!=3) good = false;
    }

    return good;
}
```

Software analysis (4 points)

Q1 Coder la fonction suivante qui permet de lister parmi les attributs d'une classe ceux qui ont des méthodes Getter et Setter « *List<String> listAttributesWithSetterAndGetter(Class clazz)* ».

Q2 On suppose que l'on dispose de la liste de tous les objets actuellement en mémoire « *List<Object> allObjectInMemory* ». Cette liste peut changer à tout moment en fonction des objets présents en mémoire (un objet nouvellement créé est ajouté dans la liste).

On souhaite analyser cette liste afin de savoir quelles sont les classes de ces objets qui possèdent des attributs avec getter et setter. L'idéal serait d'obtenir une interface graphique qui soit mis à jours en temps réel. Proposez une conception multi-threadée d'une telle analyse. Vous ferez bien attention à ne pas analyser deux fois la même classe (si par exemple deux objets d'une même classe ont été instanciés).