Programmation orientée objet
Examen individuel

Romain Marie & Hervé Gaudin

Décembre 2019

3iL INGENIEURS
</les sciences informatiques>

Pour éviter toute mauvaise surprise, prenez le temps de bien lire l'ensemble du sujet avant de commencer

Points importants avant de commencer :

- Le nom des classes, méthodes, et propriétés vous est imposé. Vous serez donc pénalisés si vous ne respectez pas rigoureusement le diagramme de classes fourni avec l'énoncé.
- Le diagramme de classes qui vous est fourni n'est pas complet ! Vous devrez donc ajouter des propriétés et méthodes aux classes.
- Sauf indication contraire, les propriétés d'une classe ne doivent jamais être public
- Si une fonctionnalité vous manque, vous avez le droit de l'ajouter. Non seulement vous ne serez pas pénalisés, mais un point bonus pourrait même être attribué si l'ajout se justifie.
- Pour éviter toute erreur de manipulation, chaque fichier que vous créez devra commencer par un commentaire indiquant vos noms et prénoms
- N'oubliez pas de correctement mettre en forme chacun de vos fichiers (indentation = ctrl+maj+f)
- Votre programme devra compiler et s'exécuter à la fin de l'examen. Vous serez pénalisés si ce n'est pas le cas.
- Vous allez programmer dans un workspace et un projet existant. Pour éviter toute erreur de manipulation, ne supprimez rien de ce qui vous est fourni.

1 Introduction

Dans cet examen, vous allez prouver votre connaissance du langage Java et de Slick2D à travers une petite application assez proche de ce qui a été vu en TD et en TP. L'idée va être de générer et d'afficher deux types de formes différents : des Disques centrés sur la fenêtre et qui grossiront au fil du temps, et des Croix, créées au centre de la fenêtre, qui se déplaceront dans une direction aléatoirement choisie.

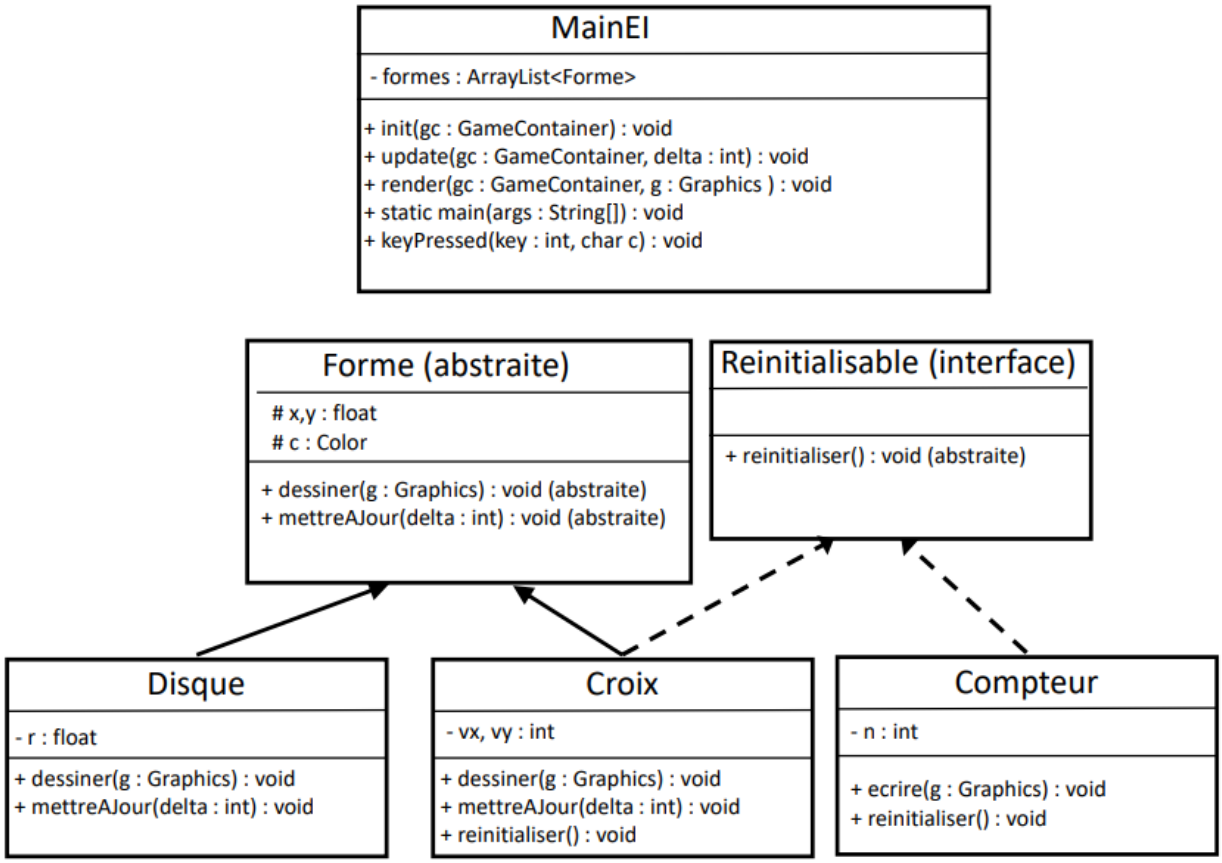
Une nouvelle Croix apparaitra toutes les 200ms, tandis qu'un Disque devra être créé chaque fois que l'utilisateur appuie sur la touche espace. Lorsque la touche r est pressée, les Croix devront toutes être replacées au centre de la fenêtre.

Pour finir, l'application devra également afficher un compteur indiquant le nombre de formes créées, et qu'il sera possible de remettre à zéro en appuyant sur la touche z.

2 Diagramme de classes

Pour parvenir à ce résultat, vous allez progressivement mettre en œuvre un ensemble de classes. Le diagramme ci-dessous vous impose une partie de leur contenu.

ATTENTION: Ce diagramme de classes n'est pas complet. Vous devrez probablement ajouter des propriétés / méthodes à certaines des classes qui vous sont proposées. Vous devez par contre impérativement scrupuleusement respecter tout ce qui vous est précisé.



3 Mise en place

1. Commencez par lancer Eclipse, en choisissant le workspace qui se trouve sur le bureau.
2. Vous trouverez dedans un projet existant et déja configuré pour utiliser Slick. Exécutez le. Si la fenêtre Slick ne s'affiche pas, appelez rapidement votre enseignant.
3. En haut du fichier MainEI.java, écrivez en commentaire votre nom et prénom

4 Génération et affichage des formes

1. Ajoutez à votre projet une classe abstraite **Forme**, prenant en propriété les coordonnées réelles **x** et **y** de son centre, ainsi qu'une couleur **c** (de type **Color**). Vous ferez bien attention à importer la classe **Color** de Slick.
2. Ajoutez un constructeur permettant de créer une **Forme** avec une couleur aléatoire, et des coordonnées passées en paramètres.

3. Ajoutez à votre classe les méthodes *dessiner()* et *mettreAJour()* telles que décrites dans le diagramme de classes. Vous respecterez bien sûr le fait qu'elles sont abstraites.

4. Ajoutez à votre projet les classes **Disque** et **Croix**, sans oublier qu'elles héritent de la classe **Forme**, et ajoutez-y les propriétés et méthodes qui vous semblent pertinentes. A leur création :

   - les **Disque** auront un rayon de 30 pixels et seront centrés aux coordonnées **x** = 250, et **y** = 250.
   - les **Croix** seront également centrées aux coordonnées **x** = 250, **y** = 250, et possèderont une vitesse suivant x et une vitesse suivant y aléatoires, chacune comprise dans l'intervalle $[-100, 100]$ pixels par seconde.

5. Pour chacune de ces deux classes, écrivez le contenu de la méthode *dessiner(Graphics g)*. Pour dessiner une Croix, il vous suffit de dessiner 2 segments (*g.drawLine()*) perpendiculaires de longueur 20 pixels, s'intersectant en leur milieu.

6. Modifiez votre classe MainEI pour qu'elle possède une ArrayList de **Forme** en propriété que vous nommerez **formes**. Vous pouvez remplacer l'ArrayList par un tableau de 1000 cases si vous le souhaitez, mais attention aux implications dans la suite du sujet.

7. Modifiez la méthode *render()* de votre classe MainEI pour que chaque **Forme** existante soit dessinée à chaque itération.

8. Modifiez la méthode *update()* pour qu'une nouvelle **Croix** soit ajoutée à l'ArrayList toutes les 200ms. N'hésitez pas à ajouter une propriété à la classe MainEI si vous estimez en avoir besoin.

9. Modifiez la méthode *keyPressed()* pour qu'un nouveau **Disque** soit ajouté à la fin de l'ArrayList à chaque fois que la touche espace est pressée. Attention : pour ceux ayant choisi d'utiliser un tableau à la place d'une Arraylist, il faudra veiller à ce que les **Forme** soient stockées dans le tableau dans l'ordre chronologique de leur création.

5 Mise à jour des formes

1. Pour chacune des classes **Disque** et **Croix**, écrivez le contenu de la méthode *mettreAJour(int delta)*. Pour un **Disque**, il s'agira d'augmenter son rayon à une vitesse de 50pixels/seconde. Pour une **Croix**, il s'agira de la déplacer en fonction de ses vitesses suivant x et y (déja choisies aléatoirement lors de la construction). N'oubliez pas que ces modifications devront être fonction du paramètre delta.

2. Modifiez la méthode *update()* de votre classe MainEI pour que chaque Forme existante se mette à jour à chaque itération.

3. Si ce n'est pas déja fait, ajoutez l'interface **Reinitialisable** à votre projet, en respectant le diagramme de classe présenté plus tôt.

4. Faites en sorte que la classe **Croix** implémente l'interface **Reinitialisable**. Pour une **Croix**, la méthode *reinitialiser()* aura pour seule action de la replacer aux coordonnées **x**=250, **y**=250.

5. Modifiez la classe MainEI, pour que chaque **Croix** soit réinitialisée lorsque la touche r est pressée.

6 Compteur

1. Ajoutez une classe **Compteur** telle que décrite dans le diagramme de classes. A la construction, sa propriété **n** sera initialisée à 0.

2. Ecrivez la méthode *ecrire()* de cette classe. Elle devra afficher dans le coin supérieur gauche de la fenêtre le message "Compteur : n" où n correspond à la valeur stockée dans la propriété n.

3. Mettez à jour la classe MainEI pour que :

- Un **Compteur** soit affiché à chaque passage dans la méthode *render()*,
- Ce même **Compteur** soit incrémenté à chaque fois qu'une **Forme** est créée (**Croix** ou **Disque**),

4. Si ce n'est pas déja fait, faites en sorte que la classe **Compteur** implémente l'interface **Reinitialisable**. La méthode *reinitialiser()* aura pour seule action de remettre à 0 le compteur (sa propriété n).

5. Mettez à jour la classe MainEI pour que, lorsque l'utilisateur appuie sur la touche z :

   - l'ArrayList formes soit entièrement vidée,
   - le **Compteur** soit remis à zéro