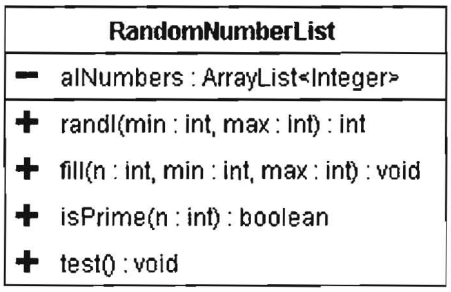
Code branche INFOR	Ministère de l'Éducation nationale, de l'Enfance et de la Jeunesse **EXAMEN DE FIN D'ÉTUDES SECONDAIRES TECHNIQUES** Régime technique – Session 2015	
Épreuve écrite	Branche	Division / Section
Durée de l'épreuve 3h	Informatique Java	GE
Date de l'épreuve 1 Juin 2015		

Dans votre répertoire de travail (à définir par chaque Lycée), vous trouverez un sous-dossier nommé **EXAMEN_GE**. Renommez ce dossier en remplaçant le nom par votre code de l'examen (Exemple de notation : **LBV_GE_01**). *Tous vos fichiers devront être sauvegardés à l'intérieur de ce sous-dossier, qui sera appelé 'votre dossier' dans la suite !*

Question 1 — 1 + 3 + 6 = 10 pts

Ouvrez le projet **RandomNumberList** de votre dossier. L'environnement de programmation est à votre choix (**Unimozer** ou **NetBeans**).

La classe `RandomNumberList` possède une liste `alNumbers` qui sera remplie avec des nombres aléatoires.



Travail à faire :

- Programmez la méthode **`randI`** qui retourne un nombre aléatoire entier dans l'intervalle **[`min`,`max`]**.
- Programmez la méthode **`fill`** afin qu'elle ajoute à la liste **`alNumbers`** **`n`** nombres aléatoires dans l'intervalle **[`min`,`max`]**.
- Complétez la méthode **`isPrime`** qui détermine si un nombre **`n`** donné est premier ou non.
- Vous pouvez tester vos développements en exécutant la méthode **`test`**.

Question 2 — 50 pts

Dans la suite, vous allez écrire l'application dénommée « Pang ». Il s'agit d'un petit jeu qui a pour but d'éliminer une balle en cliquant dessus. À chaque fois qu'on clique sur une balle, celle-ci se divise en deux. Le jeu continue jusqu'à ce les balles soient devenues trop petites et disparaissent.

Vous trouvez une version exécutable du programme (**Pang.jar**) dans le dossier **dist** de votre dossier. (Avant de continuer, il est recommandé de lancer et de tester ce programme !)

Créez avec *NetBeans* un nouveau projet nommé **Pang** dans votre dossier.

Réalisez ce programme en vous basant sur la version exécutable fournie ainsi que sur le diagramme UML de la dernière page et tout en respectant les instructions et précisions données dans la suite.

Ball (1 + 1.5 + 2.5 + 2.5 + 3 + 1.5 = 12 pts)

- La classe `Ball` représente une balle de rayon `radius`. Le centre de la balle a pour coordonnées `x` et `y`. Les attributs `xStep` et `yStep` déterminent le sens de mouvement de la balle (valeur positive ➔ vers la droite/vers le bas, valeur négative ➔ vers la gauche/vers le haut). L'attribut `color` représente la couleur de la balle.

- La classe dispose d'un constructeur ainsi que d'accesseurs et de manipulateurs (voir diagramme UML). Dans le constructeur, on attribue à l'attribut `color` la couleur bleue.

- La méthode `isInside` vérifie si le point passé en paramètre se trouve à l'intérieur de la balle (astuce : calculez la distance du point par rapport au centre de la balle et vérifiez qu'elle est inférieure ou égale au rayon).

- La méthode `isTouching` vérifie si la boule actuelle et la boule passée en paramètre se touchent, c'est-à-dire si la distance de leurs centres est inférieure à la somme de leurs rayons.

- La méthode `doStep` vérifie à l'aide de 4 conditions différentes si la boule a atteint les limites (déterminées par les paramètres `width` et `height`) et modifie le cas échéant les attributs `xStep` et `yStep`. Elle ajoute ensuite les valeurs de `xStep` (respectivement de `yStep`) aux attributs `x` respectivement `y`.

- La méthode `draw` dessine la boule en utilisant la couleur `color`.

UnstableBall (2 + 3 = 5 pts)

- La classe `UnstableBall` est une classe fille de la classe `Ball`. Les attributs `minRadius` et `maxRadius` représentent les rayons minimal et maximal de la balle, initialisés respectivement aux valeurs 10 et 40. Le constructeur fait appel au constructeur hérité pour initialiser les attributs hérités et modifie la couleur de la balle en rouge.

- La méthode `doStep` génère un nombre aléatoire compris dans l'intervalle [-2 , +2]. Ce nombre est ajouté au rayon si le rayon résultant se situe entre les limites `minRadius` et `maxRadius`. Elle appelle ensuite la méthode `doStep` de la classe mère.

Balls (1 + 4 + 3 + 6 + 1,5 + 4+1,5 = 21 pts)

- Cette classe gère la liste `alBalls` de balles. L'attribut `minRadius`, initialisé à 6, représente le rayon en-dessous duquel une balle n'est plus divisée en deux. La classe dispose des méthodes `clear` qui vide la liste et `get` qui la retourne balle indiquée en paramètre.
- La méthode `addRandomBall` ajoute au hasard une balle normale ou une balle instable à la liste `alBalls`. Les valeurs pour `xStep` respectivement `yStep` sont définies par un nombre aléatoire décimal compris dans l'intervalle [-5 , +5].
- La méthode `split` vérifie tout d'abord si en divisant par 2 le rayon de la balle référencée par le paramètre `i`, celui-ci est supérieur ou égal au rayon minimal. Dans ce cas, la méthode ajoute à la liste deux nouvelles balles ayant pour rayon la moitié du rayon actuel. Faites attention à ce que ces deux nouvelles balles ne se touchent pas !

La méthode efface ensuite la balle actuelle de la liste, peu importe si de nouvelles balles ont été ajoutées ou non.

- La méthode `getClickedBallIndex` recherche dans la liste `alBalls` l'index de la balle pour laquelle le point `p` passé en paramètre se trouve à l'intérieur de la balle. Dans le cas où aucune balle vérifiant la condition n'est trouvée, la méthode retourne -1.
- La méthode `doStep` appelle pour chacune des balles de la liste `alBalls` la méthode correspondante de la classe `Ball`.
- La méthode `checkCollisions` vérifie pour chaque balle de la liste si elle touche une balle du reste de la liste. Dans ce cas, le mouvement horizontal des deux balles est inversé.
- La méthode `draw` dessine chacune des balles de la liste `alBalls`.

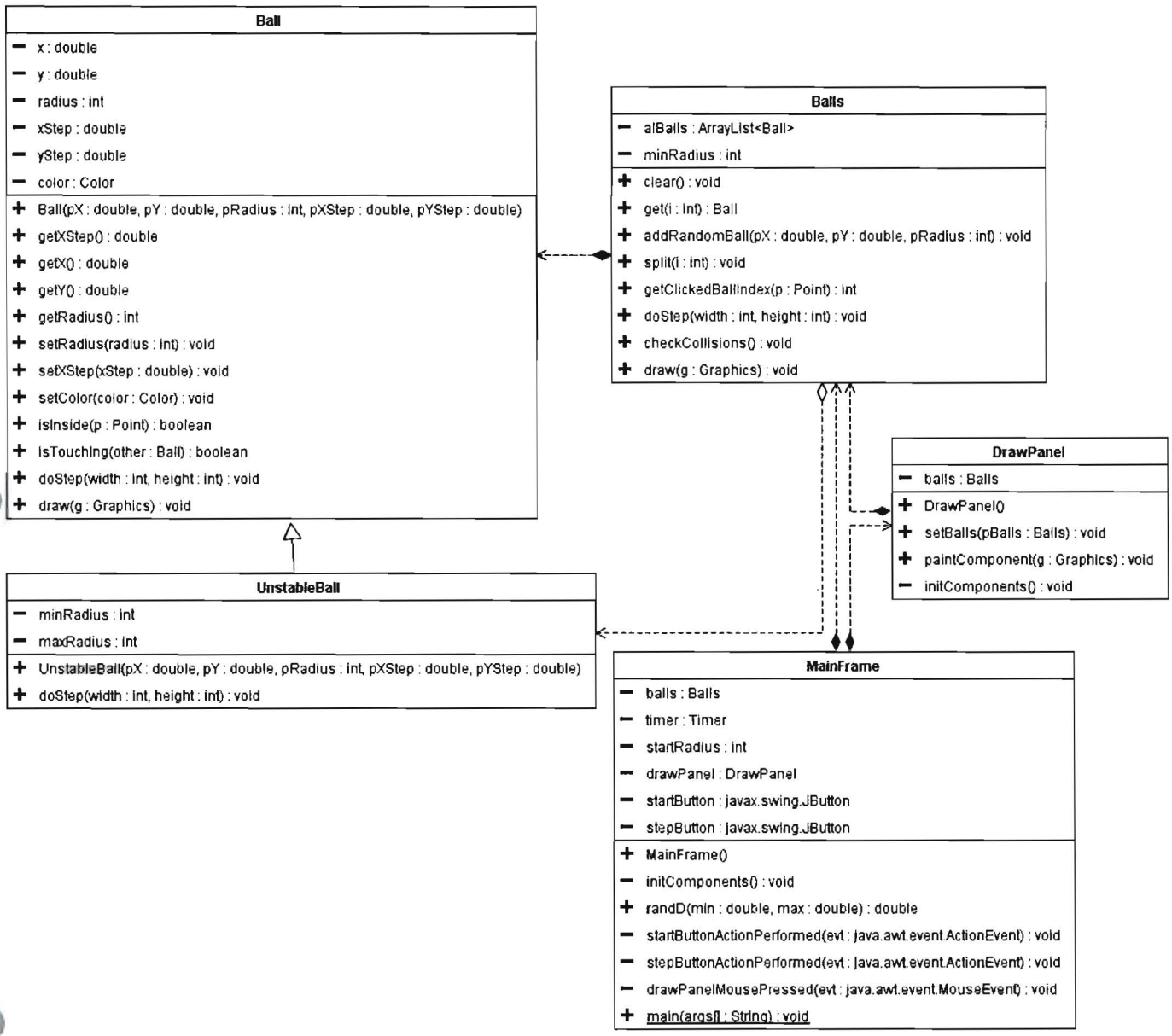
DrawPanel (1 + 1 = 2 pts)

- La méthode `setBalls` initialise la liste avec la liste passée en paramètre.
- La méthode `paintComponent` dessine un rectangle blanc sur toute la surface du `drawPanel` et dessine toutes les balles de la liste si la liste n'est pas vide.

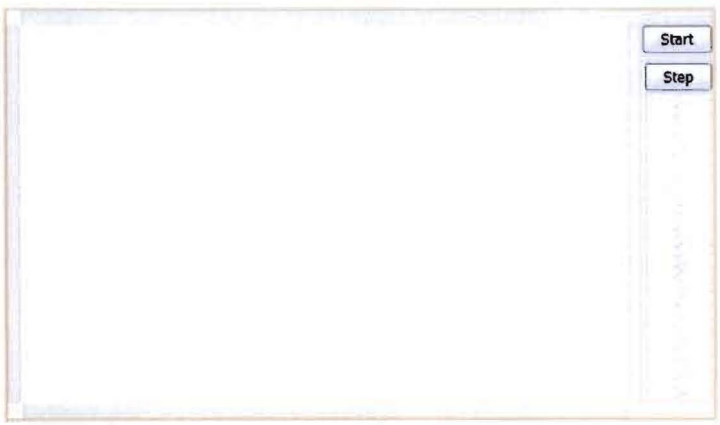
MainFrame (1 + 1.5 + 0.5 + 3 + 2 + 2 = 10 pts)

- Réalisez une interface graphique contenant un `DrawPanel` nommé `drawPanel` et deux `JButton` nommés `startButton` et `stepButton` (voir copie d'écran).
- Le rayon initial `startRadius` est fixé à 60 pixels. Le constructeur `MainFrame()` passe la liste `balls` au `drawPanel`, crée un nouveau chronomètre `timer` de périodicité de 40 ms et rend invisible le bouton `stepButton`.
- La méthode `randD` génère un nombre aléatoire décimal situé dans l'intervalle [`min`,`max`[.
- La méthode `startButtonActionPerformed` vide la liste `balls`, ajoute une balle de rayon `startRadius` à la liste `balls` et démarre le chronomètre,. La position de la balle est choisie aléatoirement, mais la balle doit se situer entièrement dans le `drawPanel`.
- La méthode `stepButtonActionPerformed` vérifie des collisions éventuelles entre les balles de la liste, ensuite toutes les balles effectuent un pas (*step*) de mouvement.
- La méthode `drawPanelMousePressed` divise la balle sur laquelle on vient de cliquer. Si aucune balle n'est cliquée, rien ne se passe.

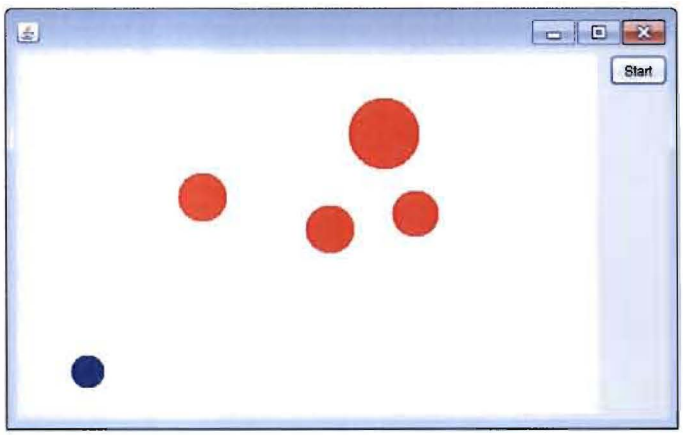
UML



Interface



Interface



Jeu en cours d'exécution