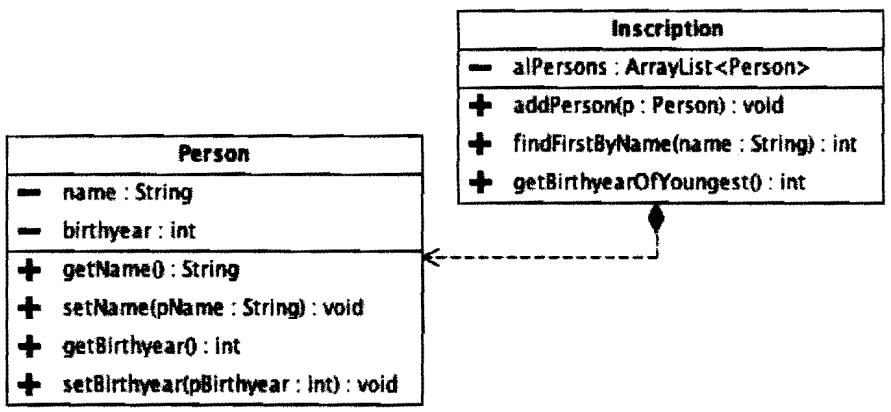
Code branche **INFOR-J**	**Ministère de l'Education nationale et de la Formation professionnelle** EXAMEN DE FIN D'ETUDES SECONDAIRES TECHNIQUES **Régime de la Formation de Technicien - Session 2012/2013**	
Épreuve écrite	*Branche*	*Division / Section*
Durée épreuve **3h**	**Informatique** **JAVA**	**GE**
Date épreuve 3 juin 2013		

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 : **LTMN_GE2_07**). *Tous vos fichiers devront être sauvegardés à l'intérieur de ce sous-dossier, qui sera appelé 'votre dossier' dans la suite !*

Question 1 — *15 points*

Ouvrez le projet « Question1 » qui contient déjà les deux classes `Person` et `Inscription`. Elles représentent des personnes pouvant s'inscrire à une certaine liste :



1. Écrivez le code source de la méthode `findFirstByName` qui recherche dans `alPersons` une personne, dont le nom est passé comme paramètre. La méthode retourne l'indice de la personne trouvée ou elle retourne la valeur -1 si aucune personne avec le nom indiqué ne se trouve dans la liste. (6 points)

2. Écrivez le code de la méthode `getBirthyearOfYoungest` qui doit déterminer et retourner l'année de naissance de la personne la plus jeune de la liste `alPersons`. Si la liste est vide, cette méthode retourne la valeur -1. (9 points)

Question 2 — *45 points*

Dans la suite vous allez écrire l'application dénommée « PathFinder ». Il s'agit d'un petit programme capable de créer des animations simples dans lesquelles une balle doit suivre un chemin que l'utilisateur a défini auparavant à l'aide de la souris.

Vous trouvez une version exécutable du programme (**Pathfinder.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 **PathFinder** 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.

PathPoint (5 points)

- Cette classe représente un point dans l'espace qui est visualisé (méthode `draw`) par un cercle de 3 pixels de diamètre. Les coordonnées `x` et `y` représentent le centre du cercle. Chaque coordonnée possède un accesseur et un manipulateur.
- La représentation textuelle d'un point est de la forme suivante : `<x>,<y>`
- Cette classe possède aussi un deuxième constructeur qui crée un nouveau point à partir d'une instance d'un objet de la classe `Point` (package `java.awt.Point`).

Ball (3 points)

- Cette classe représente une balle rouge avec un rayon de 10 pixels.
- La méthode `draw` dessine la balle à la position indiquée par le paramètre `pPathPoint` du type `PathPoint`.

Path (19 points)

- Cette classe représente le chemin que la balle doit parcourir. Elle possède à cet effet une liste de points `alPathPoints`, une balle ainsi qu'un attribut `ballIndex`. A chaque instant, la balle se trouve sur un point (`PathPoint`) de la liste `alPathPoints`. Sa position actuelle est donnée par `ballIndex` qui représente l'indice de ce point dans la liste `alPathPoints`. Si la balle ne se trouve pas sur le chemin, `ballIndex` a la valeur -1 (valeur par défaut). L'attribut `ballIndex` possède uniquement un accesseur.
- Dépendant des attributs `pathVisible` et `ballVisible`, la méthode `draw` dessine les points du chemin en bleu, respectivement la balle sur le canevas. En outre, la balle ne peut être dessinée si `ballIndex` est -1. Les attributs `pathVisible` et `ballVisible` possèdent aussi un accesseur et un modificateur respectif.
- La méthode `gotoNext` incrémente la valeur de `ballIndex`. Dans le cas où il n'existe pas de point suivant, `ballIndex` pointe vers le premier point de la liste.
- La méthode `gotoIndex` modifie la valeur de `ballIndex`, mais uniquement si l'indice du point passé en tant que paramètre est valide.
- La méthode `clear` supprime tous les points de la liste et remet à la valeur par défaut l'attribut `ballIndex`.
- La méthode `isEmpty` indique si la liste est vide ou non.
- La méthode `add` permet d'ajouter un point au chemin.
- La méthode `toArray` transforme la liste des points en un objet du type `Object[]`.

DrawPanel (2 points)

- Cette classe possède un attribut `path` du type `Path` avec un manipulateur.
- La méthode `paintComponent` vide le dessin, puis dessine le chemin `path`.

MainFrame (16 points)

- Un clic sur le bouton de démarrage *Start* fait en sorte que, la balle apparaisse, que le chemin ne soit plus affiché et que le chronomètre soit démarré. Si le chemin est vide, il ne se passe rien.
- Un clic sur le bouton d'arrêt *Stop* arrête l'animation.
- Un clic sur le bouton *Clear* fait la même chose que le bouton d'arrêt et supprime en plus tous les points du chemin.
- Le chronomètre déplace chaque 10 millisecondes la balle à sa prochaine position et sélectionne dans la liste des points cette nouvelle position de la balle.
- Lorsque l'utilisateur clique sur une entrée de la liste `pointList`, la balle apparaît à la position sélectionnée (peu importe si le chronomètre est actif ou non).
- Lorsque le chronomètre est arrêté et que l'utilisateur bouge la souris avec un bouton enfoncé au dessus du canevas, le point de la souris est ajouté au chemin. N'oubliez pas de mettre à jour le dessin et la liste des points !

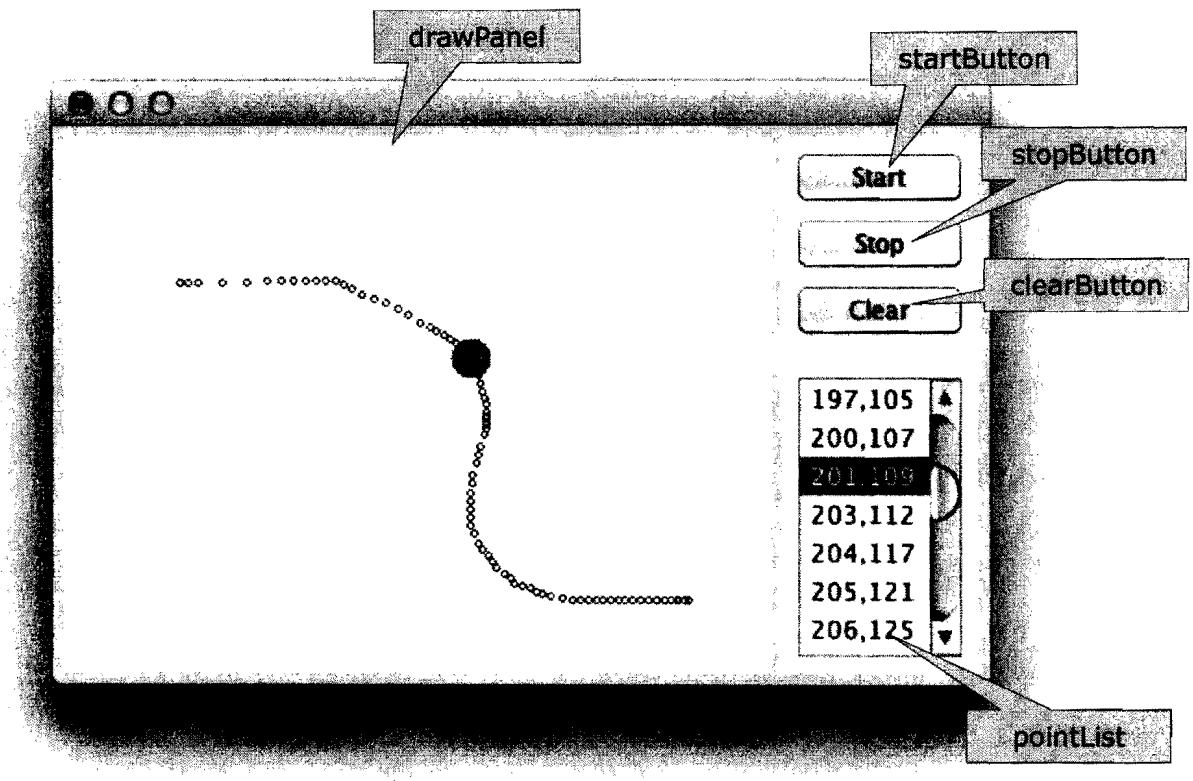
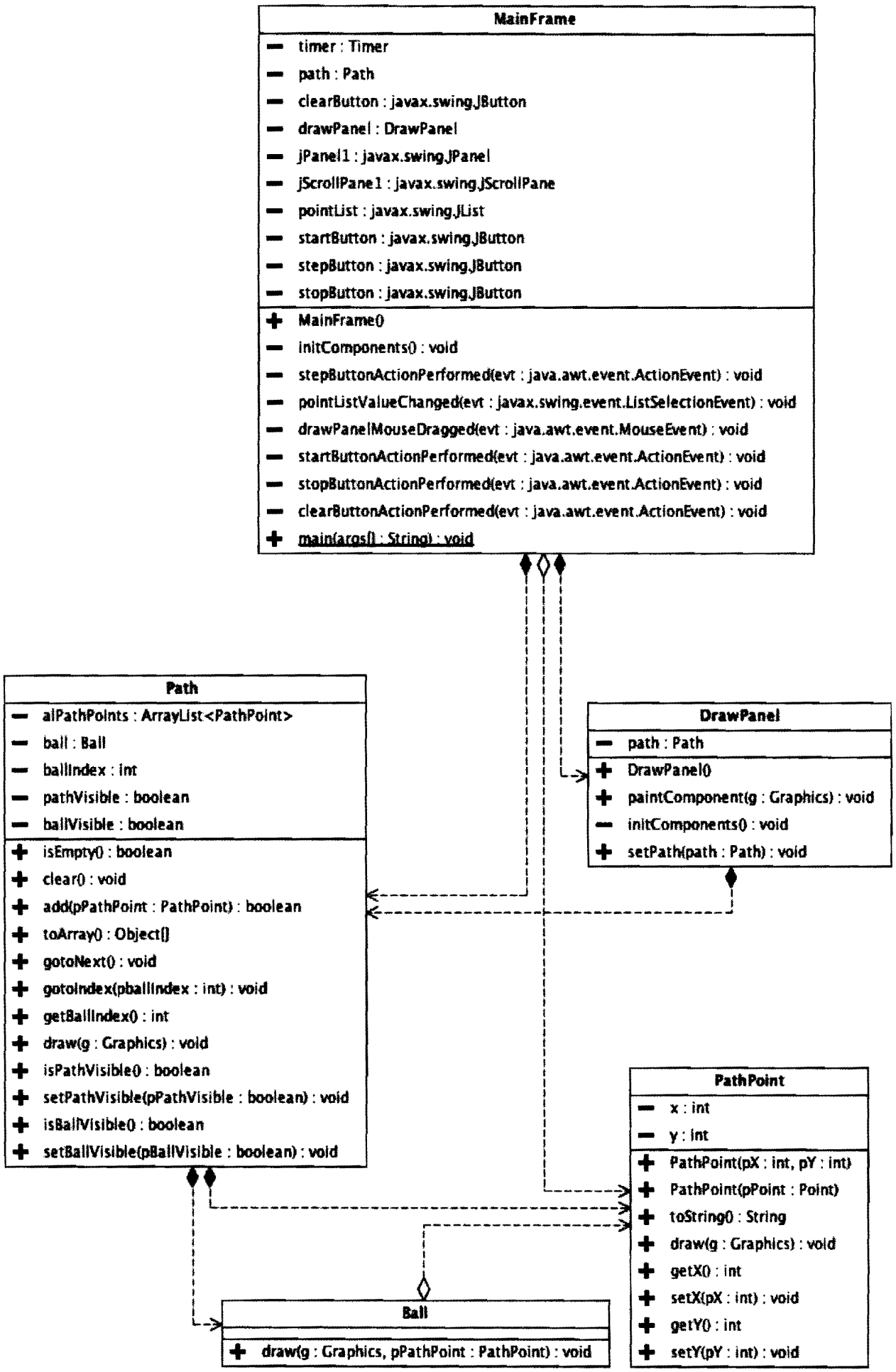


Diagramme de classes



Enseignement secondaire technique
Division technique générale
Examen 13GE

Liste des composants et classes connus

Liste des composants (propriétés, événements et méthodes) et classes à connaître pour l'épreuve en informatique à l'examen de fin d'études secondaires techniques - division technique générale.

Package	Classe	Détails	Commentaires / Particularités
javax.swing	JFrame	Méthodes - setTitle(...) / getTitle() - setLocation(...) / getLocation() NetBeans Object Inspector Property - title	
	JButton JLabel JTextField	Méthodes - setText(...) / getText() - setLocation(...) / getLocation() - getX() / getY() - getWidth() / getHeight() - setVisible(...) - setEnabled(...) Événement - actionPerformed NetBeans Object Inspector Property - icon	- le libellé *JLabel* peut aussi être utilisé pour visualiser des images via la propriété « icon » de l'inspecteur objet de NetBeans (le composant *JTextField* ne possède pas de propriété « icon »).
	JSlider	Méthodes - setMinimum(...) / getMinimum() - setMaximum(...) / getMaximum() - setValue(...) / getValue() Événement - stateChanged	
	JPanel	Méthodes - setVisible(...) - setEnabled(...) - setBackground(...) / getBackground() - getWidth() / getHeight() - getGraphics() - paintComponent(Graphics g) - repaint() Événements - MousePressed / MouseReleased - MouseDragged	- *JPanel* est utilisé pour regrouper d'autres composants visuels et pour réaliser des dessins. - Lors de la réalisation de dessins, la méthode **`public void paintComponent(Graphics g)`** est à surcharger.
javax.swing	JList	Méthodes - setListData(...) - getSelectedIndex(...) / setSelectedIndex() Événement - valueChanged NetBeans Object Inspector Properties - model - selectionMode	- *JList* est utilisé surtout pour afficher le contenu d'une liste *ArrayList*.
java.awt.event	ActionEvent	- Ce type d'objet est uniquement utilisé dans les méthodes de réaction ajoutées de manière automatique à l'aide de NetBeans.	
	MouseEvent	Méthodes - getX() / getY() - getLocation() - getButton() BUTTON3	Constantes - BUTTON1 - BUTTON2 - BUTTON3

Package	Classe	Détails	Remarques
javax.swing	Timer	Constructeur - Timer(int,ActionListener) Méthodes - start() - stop() - setDelay(...) - isRunning()	
		- Comme *ActionListener* il faut utiliser celui d'un bouton. Exemple : `Timer tm = new Timer(1000,stepButton.getActionListeners()[0]);`	
java.awt	Graphics	Méthodes - drawLine(...) - drawOval(...) / fillOval(...) - drawRect(...) / fillRect(...) - drawString(...) - setColor(...) / getColor()	
	Color	Constructeurs - Color(...)	
	Point	Constructeurs - Point(...) Méthodes - setLocation(...) / getLocation() - getX() / getY()	
java.util	ArrayList	Méthodes - add(...) - clear() - contains(...) - get(...) - indexOf(...) - remove(...) - set(...) - size() - toArray()	- `Object[] toArray()` est employé uniquement pour passer les contenus d'une liste à la méthode *setListData(...)* d'une *JList*.
java.lang	String	Méthodes - equals(...) / compareTo(...) - indexOf(...) - valueOf(...)	
	Integer Double	Méthodes - equals(...) / compareTo(...) - valueOf(...)	
	Math	Méthodes - abs(...) - round(...) - random() - sqrt(...) - pow(...) - sin(...), cos(...), tan(...)	Constante: - PI
	System	Méthode - out.print() - out.println()	