

# Application Pile d'Exécution

## Manuel Utilisateur

### Introduction

Comme son nom l'indique, la présente application permet de visualiser dynamiquement une pile d'exécution ainsi que la création de variables dans la mémoire au fur et à mesure que des méthodes et constructeurs seront appelés.

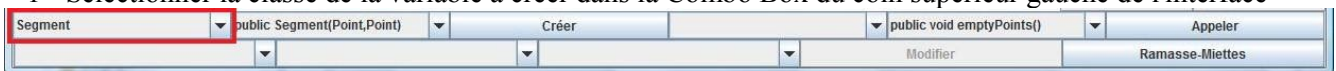
Comme l'application contient de nombreuses fonctionnalités, l'utilisation de chacune d'entre elles est expliquée dans cette documentation.

### Table des matières

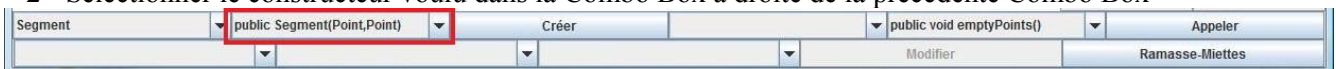
- I – Appel de constructeurs
- II – Appel de méthodes
- III – Choix Arguments Constructeur ou Méthode
- IV – Retour de la méthode appelée
- V – Modification d'attribut
- VI – Modification de poignée
- VII – Ramasse-miette
- VIII – Contrôle du temps
- IX – Attributs Déclarés en Static
- X – Fichier XML
- XI – Classes Affichées

### I – Appel de constructeurs

1 - Sélectionner la classe de la variable à créer dans la Combo Box du coin supérieur gauche de l'interface



2 - Sélectionner le constructeur voulu dans la Combo Box à droite de la précédente Combo Box



3 - Appeler le Constructeur en cliquant sur le bouton créer

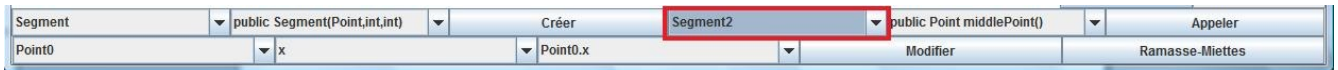


## II – Appel de méthodes

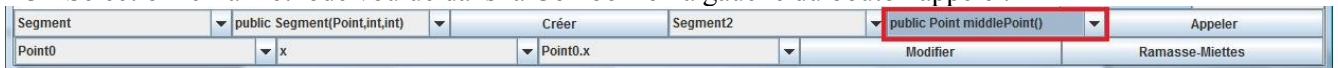
1 - Sélectionner la classe contenant la méthode voulue dans la Combo Box à droite du bouton créer.



2 - Sélectionner la variable sur laquelle la méthode sera appelée dans la deuxième Combo Box à droite du bouton créer.



3 - Sélectionner la méthode voulue dans la Combo Box à gauche du bouton appeler.



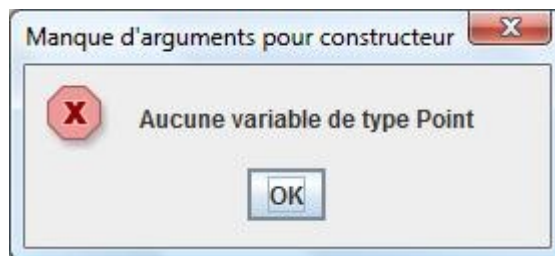
4 - Appeler la méthode choisie en cliquant sur le bouton appeler.



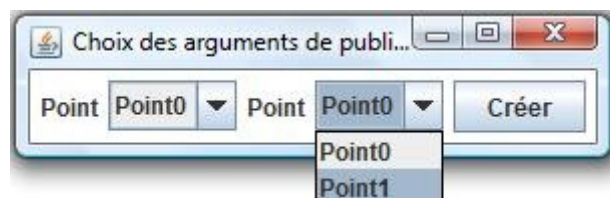
## III – Choix Arguments Constructeur ou Méthode

Si le constructeur ou la méthode ne requiert aucun argument, l'appel sera immédiat.

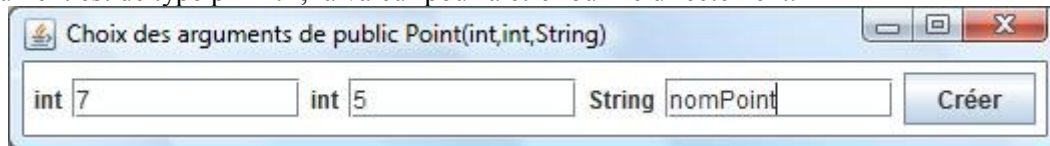
Si le constructeur ou la méthode requiert un/des arguments et que l'environnement ne possède aucune variable de type compatible avec un/des arguments, un message d'erreur s'affichera.



Si le constructeur ou la méthode requiert un/des arguments et que l'environnement possède au moins une variable de type compatible avec chaque argument, une nouvelle fenêtre apparaît dans laquelle le choix des arguments sera fait.



Si un argument est de type primitif, la valeur pourra être fournie directement.

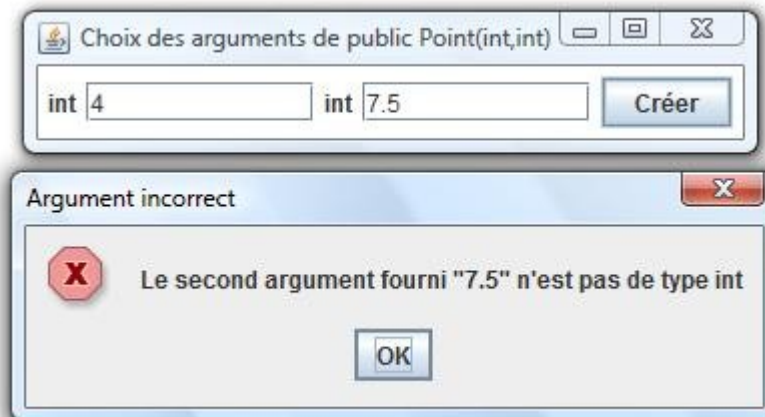


Choix des arguments de public Point(int,int,String)

int 7 int 5 String nomPoint

Créer

Cependant, si la valeur fournie n'est pas conforme au type indiqué, un message d'erreur s'affichera.



Choix des arguments de public Point(int,int)

int 4 int 7.5

Créer

Argument incorrect

Le second argument fourni "7.5" n'est pas de type int

OK

#### IV - Retour de la méthode appelée

Si la méthode retourne une valeur autre que void, une nouvelle fenêtre apparaîtra pour choisir comment affecter ce retour.



Retour p2.middlePoint()

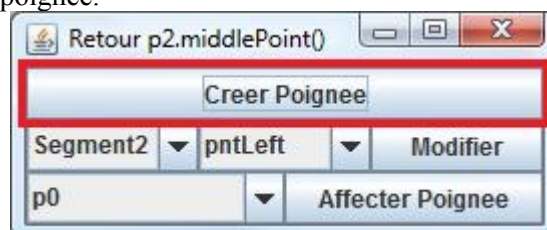
Créer Poignee

Segment2 pntLeft Modifier

p0 Affecter Poignee

**Pour l'affecter à une nouvelle poignée :**

cliquer sur le bouton nouvelle poignée.



Retour p2.middlePoint()

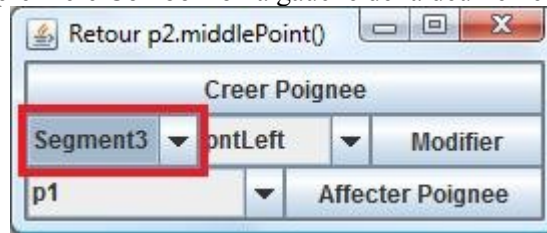
Créer Poignee

Segment2 pntLeft Modifier

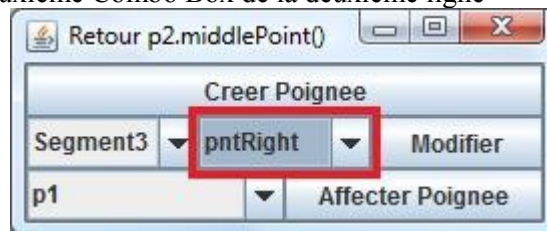
p0 Affecter Poignee

**Pour l'affecter à un attribut d'une variable :**

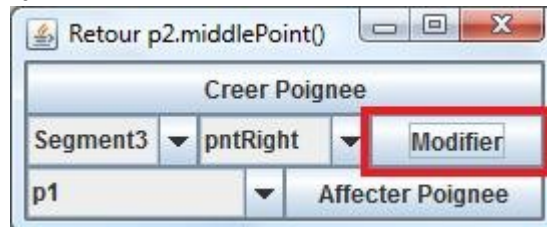
1 - Choisir la variable dans la première Combo Box à gauche de la deuxième ligne



2 - Choisir l'attribut dans la deuxième Combo Box de la deuxième ligne

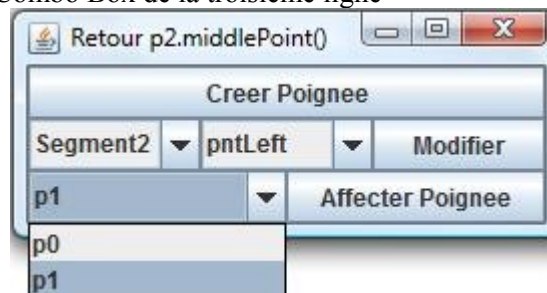


3 - Cliquer sur le bouton Modifier

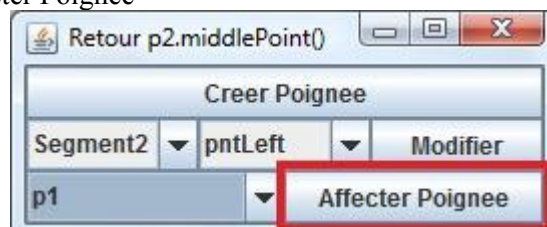


**Pour l'affecter à une poignee préexistante :**

1 - Choisir la poignée dans la Combo Box de la troisième ligne



2 - Cliquer sur le bouton Affecter Poignée



## V – Modification d'attribut

1 – Sélectionner la variable possédant l'attribut à modifier dans la Combo Box du coin inférieur gauche de l'interface

Segment	public Segment(int,int,int)	Créer	Segment2	public void emptyPoints()	Appeler
Segment3	pntLeft	Point0		Modifier	Ramasse-Miettes

2 – Sélectionner l'attribut à modifier dans la Combo Box à droite de la précédente Combo Box

Segment	public Segment(int,int,int)	Créer	Segment2	public void emptyPoints()	Appeler
Segment3	pntRight	Point0		Modifier	Ramasse-Miettes

3 – Sélectionner la nouvelle valeur de l'attribut choisi dans la Combo Box à gauche du bouton Modifier. Cette nouvelle valeur pourra être soit une variable de type compatible avec celui de l'attribut soit la valeur null.

Segment	public Segment(int,int,int)	Créer	Segment2	public void emptyPoints()	Appeler
Segment3	pntRight	Point1		Modifier	Ramasse-Miettes

4 – Modifier l'attribut choisi en cliquant sur le bouton Modifier

Segment	public Segment(int,int,int)	Créer	Segment2	public void emptyPoints()	Appeler
Segment3	pntRight	Point1		Modifier	Ramasse-Miettes

## VI – Modification de poignée

1 – Sélectionner la poignée à modifier dans la Combo Box du coin inférieur gauche de l'interface

Segment	public Segment(int,int,int)	Créer	Segment2	public void emptyPoints()	Appeler
p3		p2		Modifier	Ramasse-Miettes

2 – Sélectionner la nouvelle valeur de la poignée dans la Combo Box à gauche du bouton Modifier. Cette nouvelle valeur pourra être soit une variable de type compatible, soit une poignée de type compatible, soit la valeur null. La Combo Box à droite de celle contenant la poignée choisie restera vide.

Segment	public Segment(int,int,int)	Créer	Segment2	public void emptyPoints()	Appeler
p3		Segment3		Modifier	Ramasse-Miettes

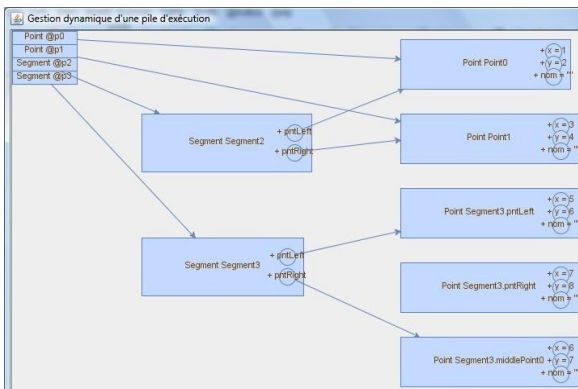
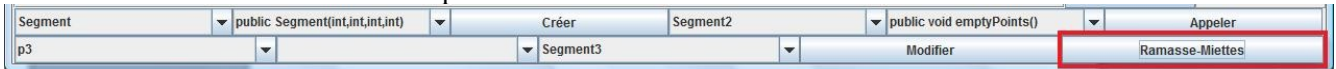
3 – Modifier la poignée en cliquant sur le bouton Modifier

Segment	public Segment(int,int,int)	Créer	Segment2	public void emptyPoints()	Appeler
p3		Segment3		Modifier	Ramasse-Miettes

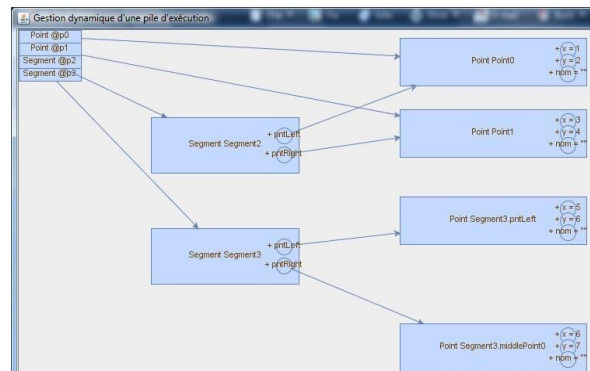
## VII – Ramasse-miette

Lorsqu'une variable n'est liée à aucune poignée et à aucun attribut d'une autre variable, elle devient complètement inaccessible. Par conséquent, elle ne pourra plus être utilisée comme argument pour une méthode, aucune méthode ne pourra être appelée dessus, et l'attribut d'une autre variable ne pourra être affecté à cette variable inaccessible. Le ramasse-miettes se charge de supprimer de l'environnement de telles variables inaccessibles et inexploitable.

Le ramasse-miettes est lancé en cliquant sur le bouton Ramasse-Miettes.



Avant appel du ramasse-miettes

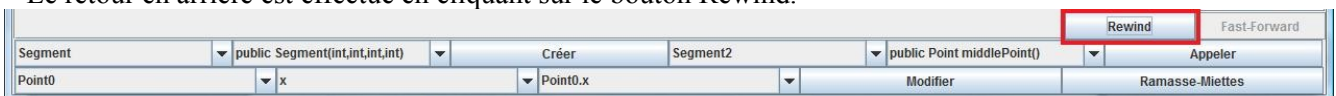


Après appel du ramasse-miettes

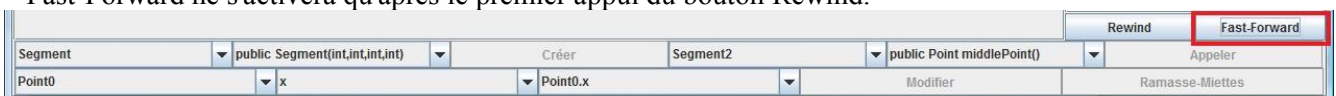
## VIII – Contrôle du temps

La présente application offre la possibilité de retourner à tout état précédent de l'environnement d'exécution et d'ensuite retourner à l'état courant de l'environnement afin de poursuivre les opérations sur l'environnement courant. Lorsque l'environnement affiché n'est pas l'environnement courant, tous les boutons seront désactivés hormis ceux permettant de contrôler le temps.

Le retour en arrière est effectué en cliquant sur le bouton Rewind.

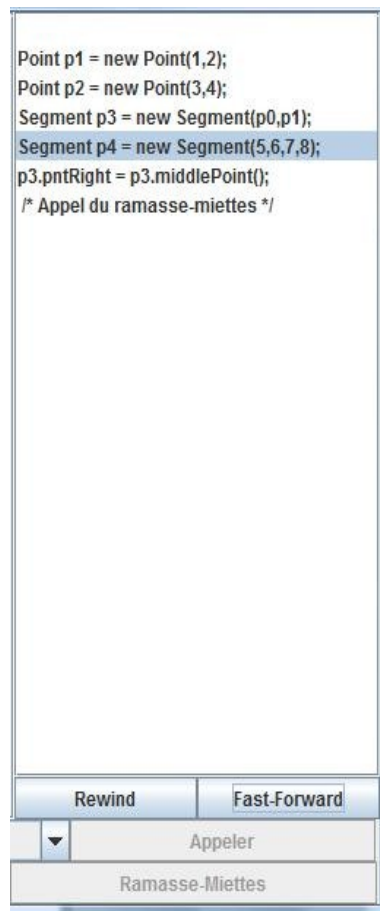


Le bouton Fast-Forward permet d'avancer vers le prochain état chronologique de l'environnement. Le bouton Fast-Forward ne s'activera qu'après le premier appui du bouton Rewind.





À tout moment, la dernière commande Java exécutée dans l'environnement affiché sera sélectionnée parmi toutes les commandes Java affichées.



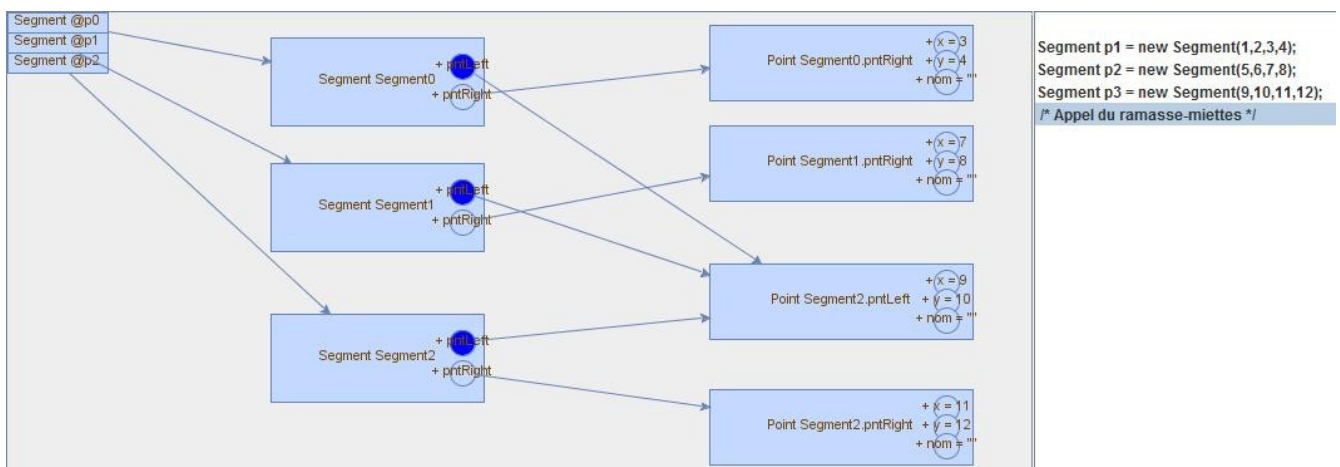
## IX – Attributs Déclarés en Static

L'application manipule et affiche correctement les attributs déclarés en static. Un attribut static sera affiché en bleu foncé dans tout noeud représentant une instance d'une classe possédant un attribut static.

Par conséquent, à tout moment, les différentes instances d'une classe possédant un attribut static pointeront toutes vers la même instance de l'attribut.

De plus, la modification de l'attribut static d'une instance d'une classe entraînera la même modification chez toutes les autres instances :

- de cette classe
- des superclasses desquelles l'attribut static pourrait être hérité
- et des sous-classes héritant de cet attribut static.



## X – Fichier XML

Avant que l'application ne soit lancée, il faut d'abord sélectionner le fichier XML de configuration.

Dans ce fichier XML de configuration seront spécifiées :

- Les classes qui seront affichées. Leur nom complet devra être écrit entre les balises `<Dessinable>` et `</Dessinable>`
- Les classes considérées comme étant primitives. Leur nom complet devra être écrit entre les balises `<Primitive>` et `</Primitive>`

Voici le fichier XML Config.xml fourni avec l'application :

```
<?xml version="1.0" encoding="UTF-8"?>
<Environnement>

    <Dessinable>pilegraph.Segment</Dessinable>
    <Dessinable>pilegraph.Point</Dessinable>
    <Dessinable>pilegraph.Point2</Dessinable>

    <Primitive>java.lang.Integer</Primitive>
    <Primitive>int</Primitive>
    <Primitive>java.lang.String</Primitive>

</Environnement>
```

## XI – Classes Affichées

Les fichiers .java ou .class des classes à afficher dont les noms sont spécifiés dans le fichier de configuration XML devront être placés dans le même package que celui contenant l'application. Comme le package se nomme `pilegraph`, le nom complet de chaque classe sera donc précédé par le préfixe `pilegraph.` afin que la classe soit localisée et reconnue par l'application.