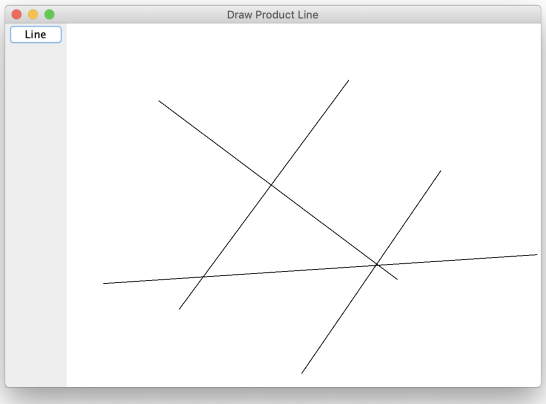
Toutes les réponses sont à rendre dans un document PDF, accompagné de votre code source. Cela comprend aussi les graphes, matrices, Treillis, etc.

Préambule

Récupérez sur Moodle l'archive DrawLPL. Cette archive comprend 5 projets Java Eclipse que vous pouvez importer dans Eclipse. Ces 5 variantes de produits font partie de famille de logiciels qui permet de dessiner des formes à l'aide d'une interface graphique. Il est possible de tracer des formes, avec des couleurs et de les effacer. Testez les projets en les exécutant dans Eclipse afin de vous faire une idée des produits que vous manipulerez dans ce TP (e.g. figure 1).

Figure 1: Capture d'écran de l'un des produits de DrawLPL



Partie 1 : Extraire les artefacts

L'objectif du TP est de réaliser l'extraction de la famille de produits vers une ligne de produits logiciel. Pour cela, vous allez appliquer dans les grandes lignes les principes de l'extraction décrit dans le cours. Les produits ont tous 4 classes : Line, Rectangle, Canvas et Main.

Exercice 1 : Extraire les OBEs

Dans cet exercice vous réaliserez l'extraction des artefacts de la LPL. A partir du code source des produits, vous devez identifier les différents artefacts en vous appuyant le méta-modèle figure 2.

1. Pour le produit P1, à la main, identifiez les Object-oriented Building Elements (OBEs) de la classe *Line.java* en suivant le méta-modèle figure 2 et réalisez les diagrammes d'instance des OBEs de ce fichier.

2. Réalisez un programme Java qui permet d'identifier et l'instancier le méta-modèle de la figure 2. Pour cela, utilisez SPOON. Commencez par modéliser les classes du méta-modèle dans votre programme. Puis, avec SPOON, parcourez l'AST afin d'instancier vos types d'*OBE*. Rajoutez aussi dans la classe *OBEs* un attribut "String name" qui vous permettra d'identifier les différents OBEs en les nommant à leur création. Par exemple :

   - $Interface$ :: *name* contiendra le nom de l'interface.
   - $Class$ :: *name* contiendra le nom de la classe.
   - *Attribute* :: *name* contiendra le nom de l'attribut déclaré.
   - $Method$ :: *name* contiendra le nom de la méthode.
   - etc.

   Ces informations peuvent facilement être récupérées avec SPOON.

Indications sur comment faire l'instanciation : Pour chaque produit, récupérez d'abord les différentes types d'OBE (package, class, methode, etc). Pour instancier les relations Access, parcourez la liste des OBE et regardez les *CtReference* de chaque éléments. Récupérez uniquement les CtReference qui pointent vers un autre OBE déjà instancié.

Exercice 2 : Définir les artefacts

Maintenant que le méta-modèle de la LPL peut être instancié par votre programma Java, vous allez récupérer les artefacts à proprement parler.

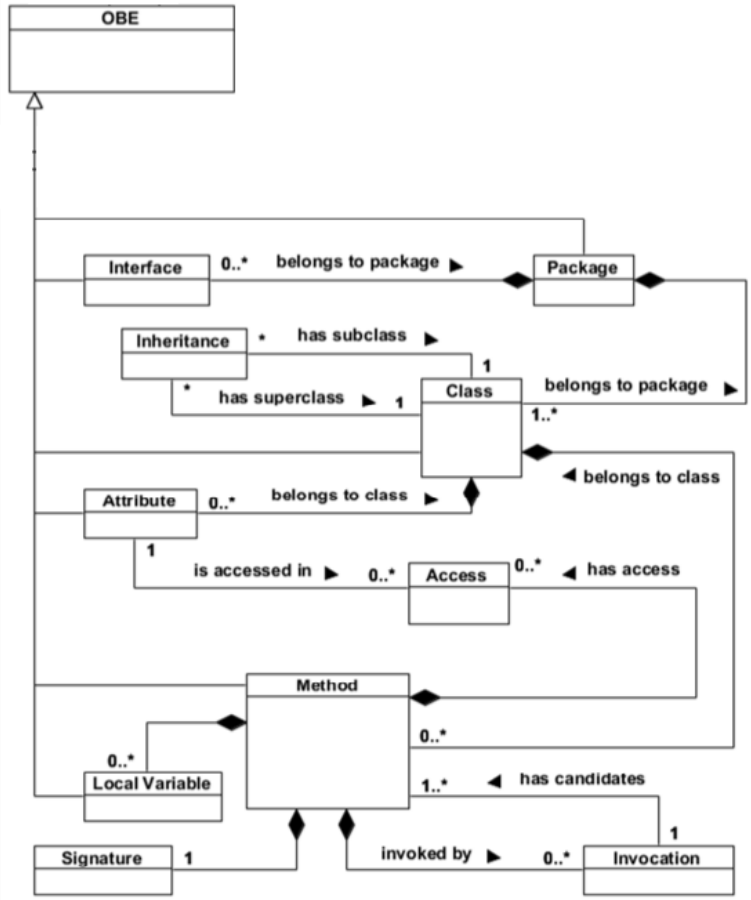
1. Nous allons définir les artefacts. Les artefacts sont un sous-ensemble des OBEs. Ceux sont autour de ces artefacts que vont se définir les points de variations de la LPL. Nous considérons les OBEs suivant comme étant des artefacts :

   - Class
   - Attribute
   - Method

   Rajoutez une interface Java à votre programme, nommez cette interface $IArtefact$ et implémentez là dans les classes *Class*, *Attribute* et *Method* de vos OBEs.

2. Définissez dans cette interface les méthodes suivantes :

Figure 2: Méta-Modèle des **O**bject-oriented **B**uilding **E**lements (OBEs).



- String getName() : renvoi le nom l'artefact (celui que vous avez extrait dans l'exercice 1)
- int getIdentifiant() : renvoi le Hash du nom de l'artefact (return getName().hashcode();)

3. Complétez le corps des méthodes de $IArtefact$ pour les classes Class, Attribute et Method.

4. Ajoutez à votre programme une fonction qui renvoie la liste des artefacts qui composent un produit. Astuce : le type $IArtefact$ vous permet de facilement identifier les artefacts parmi l'ensemble des OBEs.

5. Utilisez maintenant la liste des artefacts de chaque produit pour définir la configuration du produit. Formez à partir de chaque configuration une matrice des configurations. Cette matrice décrit les artefacts par produits. Nous nommerons cette matrice le PCM (Product Configuration Matrice). Figure 4 présente un exemple.

6. Téléchargez **rca-explorer** (disponible sur moodle). Suivez les instructions des vidéos pour apprendre à créer un concept formel à partir du PCM (tuto 1). (**Optionnelle**) Vous pouvez sinon exportez votre matrice au format .rfc, afin de la charger dans rca-explorer sans avoir à l'écrire à la main. Il s'agit d'un format proche de CSV, à la différence que les séparateurs sont des pipes —ët qu'il faut un pipe en début et fin de chaque ligne (voir figure 3)

7. Exécutez l'algorithme FCA de rca-explorer sur le PCM pour obtenir le treillis de concept (tuto 2). A partir du Treillis, donnez la liste des artefacts communs et optionnels de la LPL.

Figure 3: Exemple de concept formel au format rfc pour rca-explorer

```
|           | flyingSquirrel | bat | ostrich | flamingo | chicken |
| John      |                |     |         | x        | x       |
| George    |x               |     |         | x        | x       |
| Paul      |                | x   |         |   x      | x       |
| Ringo     | x              | x   |         |          |         |
```

Figure 4: Exemple de Product Configuration Matrice (PCM)

	Class (PaintJPanel_Drawing.Shapes.Core)	Class (DrawingShapes_Drawing.Shapes.Core)	Class (MyShape_Drawing.Shapes.Core)	Class (LineSettings_Drawing.Shapes.Line)	Class (LinePosition_Drawing.Shapes.Line)	Class (MyLine_Drawing.Shapes.Line)	Class (ImagePath_Drawing.Shapes.Image)	Class (MyImage_Drawing.Shapes.Image)	Class (ImagePostion_Drawing.Shapes.Image)	Class (ArcSettings_Drawing.Shapes.Arc)	Class (Arc_Drawing.Shapes.Arc)	Class (ArcAngle_Drawing.Shapes.Arc)	Class (MyTextShape_Drawing.Shapes.Text)	Class (Text_Drawing.Shapes.Text)	Class (TextInfo_Drawing.Shapes.Text)
Software_1	✗	✗	✗	✗	✗	✗	✗	✗	✗						
Software_2	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Software_3	✗	✗	✗	✗	✗	✗	✗	✗	✗						
Software_4	✗	✗	✗	✗	✗	✗	✗	✗	✗						

Exercice 3 : Extraire la variabilité et les caractéristiques

1. Toujours à partir du treillis obtenu à l'exercice 2, identifiez les artefacts co-occurrents. Pour chaque groupes d'artefacts co-occurrents, listez l'ensemble de ses artefacts en donnant le type et son nom. (ex. Class:Line, Attribut:Color, etc).

2. L'extraction de vos artefacts faite, vous obtenez une information capitale sur les produits que vous venez d'étudier : La liste de ses caractéristiques: [LINE, RECTANGLE, COLOR, WIPE]. Explorez le code source associé à vos groupes: si nous devions associer

chaque groupe[1] d'artefacts co-occurrents à une caractéristique, est-ce que cela serait pertinent? Donnez un groupe pour lequel l'association 1-1 avec une caractéristique est pertinente. Et donnez un groupe pour lequel cette association ne vous semble pas *normale*. Argumentez votre décision.

Partie 2 : L'identification des caractéristiques

Exercice 4 : Couplage Structurelle

Comme nous l'avons vu en cours, les groupes d'artefacts co-occurrents peuvent cacher l'implémentation de plusieurs caractéristiques. A partir des résultats de l'exercice précédent, vous allez implémenter l'approche vu en cours pour identifier les groupes minimaux d'artefacts. Pour cette exercice, vous allez travailler sur le groupe d'artefacts co-occurrrents identifié à l'exercice 3.2 comme pouvant contenir les artefacts de plusieurs caractéristiques.

1. Analysez les OBEs de votre LPL pour obtenir les relations de dépendances structurelle entre les artefacts appartenant à un même groupe d'artefacts co-occurrents. Produisez le graphe des dépendances pour ce groupe d'artefacts. Utilisez pour cela les classes et leurs relations que vous avez instancié dans l'exercice 1.

2. A partir du graphe, construisez la matrice de couplage des dépendances structurelles.

Exercice 5 : Les groupes minimaux d'artefacts

Nous avons pendant le cours que prendre aussi en compte le couplage lexical pouvant contribuer à améliorer les résultats. Toutefois, nous allons uniquement considérer le couplage structurel dans ce TP pour découvrir les groupes minimaux d'artefacts.

1. Avec rca-explorer, créez un concept formel pour la matrices de couplage structurel.

2. Pour ce concept formel, générer avec le treillis de concept à l'aide de rca-explorer, en utilisant ACPOSET comme algorithme à la place de FCA (voir la figure 5.

3. Observez-vous un treillis qui apparaît déconnecté ? Les treillis déconnectés indiquent que vous pouvez extraire l'implémentation de plusieurs caractéristiques pour un même groupe. Ces sous-groupes vous semblent-t-ils mieux capturer l'implémentation des caractéristiques spécifiques ? Sinon, essayez d'expliquer (selon vous) pourquoi malgré l'étude du couplage structurel, nous ne parvenons pas à isoler l'implémentation des caractéristiques d'un groupe d'artefacts co-occurrents.

[1] Notez qu'un groupe ne contenant qu'un seul artefact va aussi être associé à une caractéristique

Figure 5: Comment changer l'algorithme de construction de rca-explorer vers ACPOSET