Tutoriel de Compléments en Programmation Orientée Objet : Prise en main d'IntelliJ IDEA

Ce TP est à voir comme une série d'exercices simples permettant de faire le tour de quelques fonctionnalités très utiles de l'IDE [1] IntelliJ IDEA. Il est possible que tout ne soit pas toujours expliqué en détails, auquel cas, utilisez votre moteur de recherche (notez qu'il y a un moteur intégré dans la barre d'outils d'IntelliJ IDEA !) ou bien n'hésitez pas à me demander de l'aide.

Au fur et à mesure que vous explorez les menus, notez les racourcis clavier qui vous semblent utiles. Vous gagnerez du temps pour la suite.

Exercice 1 : Avant de commencer

Ouvrir IntelliJ IDEA (commande `idea`).

Pas d'autres étape préliminaire, comme créer un espace de travail (au contraire de l'IDE Eclipse).

Notez qu'il n'y a pas besoin de demander d'enregistrer les fichiers avant de compiler : IntelliJ IDEA utilise l'enregistrement automatique.

Exercice 2 : Créer le projet et ses composants

1. Créez un projet Java : ouvrir le menu « File » puis le sous-menu « New » et choisir « Project... ». Dans le dialogue qui s'ouvre, choisir « Java », puis cliquer plusieurs fois « Next » en laissant les options par défaut. Donner un nom au projet quand la question est posée.
2. Créez quelques paquetages dans le projet : dans l'arborescence sur la partie gauche de l'écran (onglet « Project »), faire un clic droit sur le dossier `src`, puis « New », « Package ».

   Rappel : les paquetages ont par convention des noms tout en lettres minuscules et chiffres. [2]. Supposons pour la suite que vous avez créé `package1` et `package2`.

   Remarquez qu'on peut créer un package à l'intérieur d'un autre package.
3. Créez des classes (toujours en passant par les menus contextuels : « New », puis « Java Class ») : en dehors des paquetages (c.-à-d. directement dans `src`) et dans chacun des paquetages. Supposons que vous avez créé `MaClasse` dans `src`, puis `MaClassei` dans `packagei` (i=1,2).
4. Regardez comment IntelliJ IDEA pré-remplit chaque fichier créé. À quoi sert la ligne « `package blabla;` » au début du fichier ?

Exercice 3 : Remplir les classes

1. Ouvrez la classe `MaClasse`.
2. Dans les accolades de la classe, écrivez « `psvm` » [3], puis faites Tabulation. IntelliJ IDEA génère alors la méthode `main()`.
3. Si votre code est mal indenté, faites ctrl+alt+L. IntelliJ IDEA réarrange le code.

1. Integrated Development Environment
2. Pour les conventions de nommage en Java, vous pouvez consulter le document suivant : `http://www.loribel.com/java/normes/nommage.html`.
3. abbréviation de « `public static void main` »

4. Ajoutez un affichage (`println()`) à l'intérieur du main, puis exécutez (shift+F10, ou alt+shift+F10 pour avoir le choix de la cible à exécuter, ou bien menu « Run », ou bien l'icône en forme de triangle vert).
5. Avez-vous remarqué qu'IntelliJ IDEA propose toute une liste de choix dans un menu surgissant, au fur et à mesure que vous écrivez `System.out.println()` ? (Pratique pour retrouver les noms des méthodes et gagner du temps pour les écrire.). À tout moment, en cours de saisie, vous pouvez valider le premier choix d'"autocomplétion avec Entrée.
6. Déclarez (à la main) des attributs dans les différentes classes.
7. Générez leurs accesseurs (menu « Code », « Generate... », ou bien taper alt+inser).
8. Dans `MaClasse`, déclarez un attribut statique `champ1` de type `MaClasse1`.
9. Remarquez qu'IntelliJ IDEA écrit `MaClasse1` en rouge. Remarquez qu'une suggestion s'affiche (tant que le curseur est sur le texte rouge). Faire alt+entrée pour appliquer la réparation suggérée. IntelliJ IDEA ajoutera alors « `import package1.MaClass1;` » au début du fichier.

   Si aucune suggestion n'était automatiquement affichée, alors la combinaison alt+entrée fait surgir une liste de choix pour réparer le problème.
10. Dans le `main()` (de `MaClasse`), initialisez l'attribut `champ1` en instanciant un objet (`new MaClasse1()`).
11. Ajoutez l'affichage `System.out.println(champ1);`. Exécutez. Que voyez-vous ?
12. Allez dans `MaClasse1`. Ajoutez quelques attributs non-statiques.
13. Allez dans « Code », « Generate... », choisissez « Constructor » et sélectionnez tous les attributs dans la fenêtre qui surgit, validez. Remarquez le nouveau constructeur ajouté.
14. Retournez dans `MaClasse`, remarquez le problème, analysez et corrigez-le à l'aide des suggestions d'IntellIJ IDEA. Exécutez à nouveau, rien ne devrait vraiment avoir changé.
15. Retournez dans `MaClasse1`, faites « Code », « Generate... », choissez « toString() » avec les bonnes options. Validez, exécutez à nouveau ? Est-ce mieux ?
16. Maintenant, on veut renommer « `MaClasse1` ». Placez votre curseur sur le mot `MaClasse1`, allez dans le menu « Refactor », faites « Rename... » (ou bien faites un clic-droit sur le mot, puis idem, ou bien faites shift+F6), choisissez un autre nom. Observez que toutes les occurrences de ce nom ont été changées en conséquence (y compris le nom du fichier `.java`).

Exercice 4 : Déboguer

1. Dans votre main, ajoutez l'instructions suivante :

```
1    for (int i=10; i >= 10; i+=3) System.out.println(i);
```

   Exécutez. Que se passe-t-il ? Arrêtez l'exécution en appuyant sur le carré rouge au bord du cadre où s'exécute le programme ou bien dans la barre d'outils principale (sinon « Run », « Stop »).
2. Faites un clic gauche dans la marge (à droite du numéro de ligne) sur la ligne de l'instruction `for`, afin d'y faire apparaître un rond rouge (un point d'arrêt/*breakpoint*).

3. Maintenant, exécutez le programme en mode débug : c'est l'icône de la barre d'outils à droite de celle qui exécute normalement (sinon, menu « Run », « Debug » ou shift+F9). Ceci va faire apparaître un onglet « Debugger » dans la même zone que le cadre d'exécution (désormais l'onglet « Console »).
4. Le programme est maintenant en pause sur l'instruction qui a le breakpoint. Pour avancer pas-à-pas dans l'exécution, on peut utiliser les touches F7 (qui fait les pas en entrant dans les méthodes) et F8 (qui saute jusqu'au retour de méthode, si jamais l'instruction courante en appelle une). Appuyez plusieurs fois sur F8 pour voir le programme avancer.
5. Dans l'onglet « Debugger », regardez le cadre « Variables », et remarquez qu'on peut observer la valeur de chaque variable dans la portée courante, afin d'aider à déceler l'origine du bug.