1 Prise en main

Décompressez l'archive `sources.zip` dans votre répertoire `../data/java/coo`.

Dans BlueJ : Project → Open Project... puis ouvir le projet Tv du répertoire `coo/tp1`.

Dans la fenêtre de projet apparaissent différentes icônes :

- en haut à gauche, une icône désignant le fichier `README.TXT` permettant de donner une description du projet. Un double-click permet d'afficher ce fichier et de le modifier si besoin.
- au centre, une icône saumon de titre Tv. Elle représente la classe de nom `Tv`. Un double-click permet d'afficher le source de la classe.

Un rapide coup d'œil sur le source permet d'avoir un premier aperçu d'un code Java. Ne rentrons pas dans le détail de ce code maintenant, les éventuels points obcurs s'éclairciront au cours des prochaines séances.

Intéressons nous plutôt au menu déroulant en haut à droite de la fenêtre, le choix actuel est Implementation. Activez l'autre choix : Interface.

Le contenu de la fenêtre change alors et vous propose la documentation sur la classe `Tv`. La structure de cette documentation est commune à tous les projets Java. Elle est générée à partir de commentaires insérés dans le source (si vous y rejettez un œil, vous devriez pouvoir deviner lesquels) et de l'outil JavaDoc que nous découvrirons prochainement.

Cette documentation est un document hypertexte qui présente les différents points suivants :

1. le nom de la classe
2. une description de la classe et des informations "annexes" (auteur, version, etc.)
3. le(s) constructeur(s)
4. la signature des méthodes (avec hyperlien vers les détails)
5. les détails des méthodes. Par exemple pour la méthodes `changeChaine` vous avez un descriptif du paramètre (rubrique `Parameters`) et pour la méthode `chaineCourante` un descriptif du résultat (rubrique `Returns`).

Il faudra s'habituer à la lecture de documentation de ce style.

2 Première classe, premières manipulations

Au fur et à mesure des manipulations, veillez à consulter le contenu de la fenêtre "Terminal Window" afin de visionner la syntaxe des invocations Java (si vous avez bien activer l'option "Record Method Calls" comme demandé précédemment) et leurs résultats. Vous pouvez réinitialiser le contenu de cette fenêtre par `Option → Clear`.

Revenons à la fenêtre projet, contenant la classe `Tv`.

compilation un click droit sur l'icône de la classe ouvre un menu contextuel : choisissez Compile pour compiler la classe (le dessin de l'icône change). L'icône de classe change de couleur pendant sa compilation.

création d'instance pour l'exploitation de la classe, il est nécessaire de créer une instance : choisir `new Tv()` dans le menu contextuel et choisissez un identificateur/référence pour votre instance. Une représentation de l'objet apparaît dans la zone grisée située en partie inférieurxse de la fenêtre. La syntaxe de la création de l'instance apparaît dans la fenêtre du terminal.

visualisation de l'état click droit sur l'instance puis choisissez *Inspect* ; un double-click produit le même résultat. Intéressons nous uniquement à la zone Object fields où l'on visualise les différents attributs qui définissent l'état de notre objet ainsi que leur valeur.

envoi de message/invocation de méthode activez le menu contextuel sur l'icône de l'objet et choisissez `chaineCourante()`. Une fenêtre s'affiche vous donnant le résultat retournée par la méthode dont vous pouvez vérifier la signature et le descriptif dans la documentation.

encore Invoquez la méthode `allume()`, vous pouvez visualiser dans la fenêtre d'inspection le changement de d'état de l'objet. La valeur de son attribut `allume` est passée de `false` à `true`.

et encore Invoquez la méthode `affricheEtat()`, cette fenêtre provoque un effet de bord en affichage et un terminal s'ouvre donc pour en permettre la visualisation.

invocations et Java Dans le terminal, choisissez Options→Record method calls. Refaites alors un appel à la méthode `chaineCourante()`, vous voyez alors apparaître dans le terminal la trace de la syntaxe Java de l'invocation de la méthode sur l'instance. Notez la notation ".". Le résultat est également affiché.

libération de référence Dans le menu contextuel de l'instance, choisissez *Remove* : vous avez ici supprimé toute référence à l'objet. Il n'y a alors plus aucun moyen de le référencer et la place qu'il occupait en mémoire peut donc si besoin être récupéré par le ramasse-miettes (*Garbage Collector*).

3 Première classe, autres manipulations

Gardez toujours la fenêtre du terminal active (avec l'option permettant l'affichage des appels Java, vous pouvez faire un Options→Clear pour dégager le terrain). Jetez y régulièrement un coup d'œil au cours des manipulations suivantes afin de vous familiariser avec la syntaxe Java associée aux différentes invocations.

instances Créez une instance de la classe `Tv` et visualisez son état.

invocations Effectuez quelques invocations sur cette instance créée, consultez la documentation pour un bon usage des méthodes. Certaines méthodes requièrent que l'on fournisse un argument, la nature de cet argument est alors rappelée dans la fenêtre qui s'ouvre.

À voir dans le source :

- le constructeur,
- les commentaires, "`/* .. */`" et "`// ...`",
- déclaration d'attributs,
- initialisation d'attributs à la déclaration,
- déclaration de méthodes, avec ou sans argument,
- méthode avec valeur de retour, utilisation du `return`,
- la délimitation de bloc par les accolades { ... },
- la syntaxe de la structure conditionnelle `if`, avec ou sans `else`
- les opérateurs `++` et `--`,
- l'opérateur `+` de concaténation des chaînes, et les chaînes notées entre guillemets doubles `"`.
- le `System.out.println` qui permet un affichage.

4 Seconde classe

(Conservez par la suite toujours la fenêtre du terminal ouverte afin de visualisez les invocations JAVA réalisées)

Ouvrez le projet `Livre1`. Deux classes appartiennent cette fois à ce projet. Vous pouvez remarquer que les icônes des deux classes sont reliées par une flèche. Celle-ci indique qu'il existe une relation de dépendance entre les classes : la classe `Bibliotheque` utilise des instances de la classe `Livre`. Cette information est importante. Elle souligne que toute modification de la classe `Livre` peut avoir des répercussions sur le fonctionnement de la classe `Bibliotheque` et d'autant plus si l'on modifie l'interface publique de `Livre`.

Créez une instance de `Bibliotheque`. Si vous n'y arrivez pas, peut être avez-vous oublié une étape ? Référez vousà l'exercice 2.

Crées deux instances de `Livre`. Trois des arguments du constructeur sont des objets de classe `String`. Cette classe est un peu particulière et vous pouvez utiliser comme valeur d'instance de cette classe toutes les constantes chaînes de caractères qui se notent **entre guillemets**. Par exemple, vous pouvez créer un objet par :

```
new Livre("JRR Tolkien","Le Seigneur des Anneaux",1954,"...")
```


Examinez les états de ces deux instances de `Livre`[1]

Invoquez la méthode `ajouteLivre` sur l'objet `Bibliotheque` créé. Cette méthode prend pour argument une instance de la classe `Livre`, vous pouvez donc donner comme argument l'identifiantv de l'une des deux références d'instance dont vous disposez (`livre_1` par exemple). Attention, il s'agit ici d'une référence et non pas d'une chaîne de caractères, et donc pas de guillemets !

Effectuez diverses manipulations avec ce projet.

À voir dans le source :

- un constructeur avec paramètre dans `Livre`,
- l'utilisation de `this` dans le constructeur de `Livre`,
- le modificateur `private` sur les attributs, et `public` sur les méthodes,
- la syntaxe de la structure itérative `while`,
- le `import` en début de code, qui indique que l'on souhaite utiliser une classe définie dans un autre paquetage, `java.util.*` ici, nous reviendrons là-dessus plus tard,
- l'utilisation de la collection `ArrayList` (définie dans le paquetage `java.util`), qui sera également détaillée utltérieurement.

Morale du tp

- L'exploitation se fait par l'intermédiaire d'instances créées en cours d'exécution à partir des classes créées en phase de conception.
- Toute manipulation se fait à travers un objet en lui envoyant un message. On utilise dans ce cas sa référence.

[1] On pourra regretter qu'il n'y ait pas moyen de distinguer les fenêtres des deux instances, mais un double click sur l'instance qui vous intéresse permet de sélectionner sa fenêtre.