Instructions simples et introduction au développement Java sous eclipse

1 Eclipse

Eclipse est un environnement de développement libre et à code ouvert dédié à plusieurs langages de programmation et de modélisation, dont le langage Java. Il est facilement extensible grâce à la notion de plugin que nous n'aborderons pas ici. Vous devez l'installer pour faire le TP, y compris dans les salles informatisées de l'université. La procédure pour l'installation d'Eclipse est décrite ici : `https://moodle.umontpellier.fr/mod/page/view.php?id=131074`

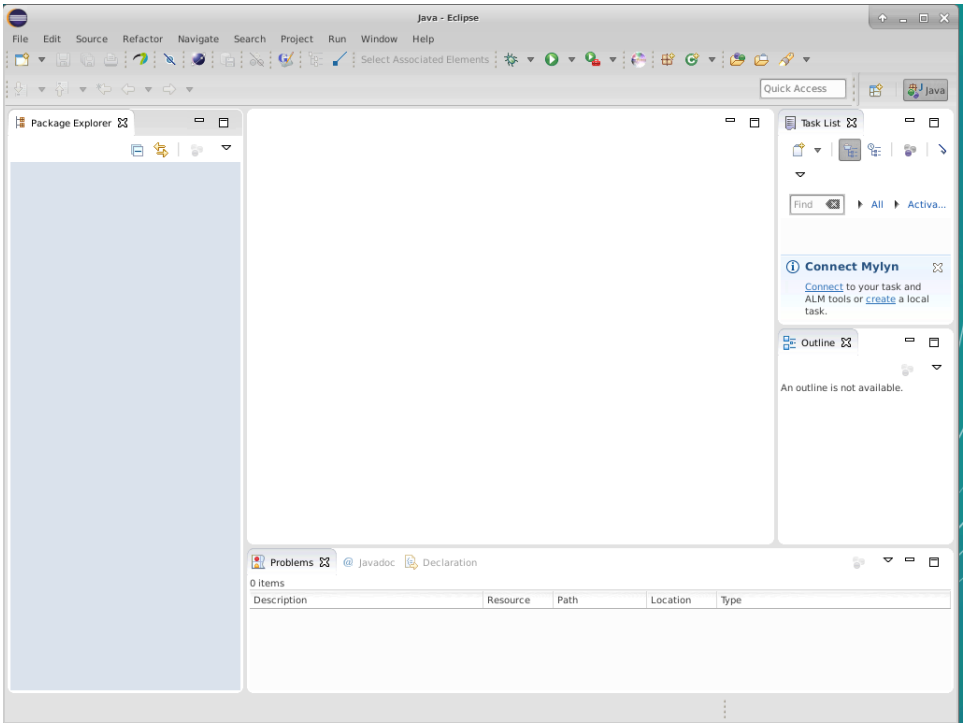
Attention, vous devez installer cette version d'éclipse :

1.1 Eclipse : quelques manipulations de base

Nous indiquons ici des manipulations de base. Si vous connaissez déjà Eclipse, lisez ce qui suit et assurez-vous que vous connaissez toutes les manipulations indiquées. Eclipse existe dans plusieurs versions, donc ce document peut comporter de légères différences avec la version que vous utilisez.

1.2 Lancer Eclipse

Lancez Eclipse par le menu Application de votre bureau.
Vous êtes invité à préciser votre workspace. Le workspace correspond à l'espace de travail d'Eclipse, c'est en fait un répertoire où seront notamment stockés vos fichiers sources. Vous pouvez avoir plusieurs workspaces si vous le souhaitez. Nous allons garder le workspace par défaut, à savoir : /home/*votreLogin*/workspace. Lorsque vous validez, la fenêtre de l'application Eclipse s'ouvre.

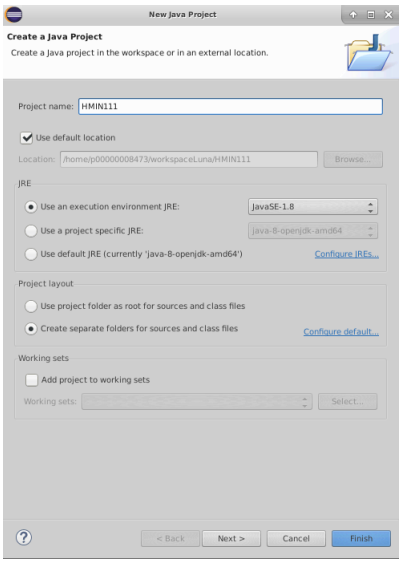


1.2.1 Création d'un projet

La première chose à faire pour écrire un programme Java sous Eclipse est de créer un projet. Pour cela, vous avez deux possibilités :

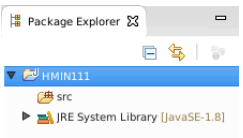
1. Dans le package explorer (partie gauche de la fenêtre) : clic droit → new → Java Project
2. Dans le menu : File → New → Java Project

Un assistant (wizard) s'ouvre.



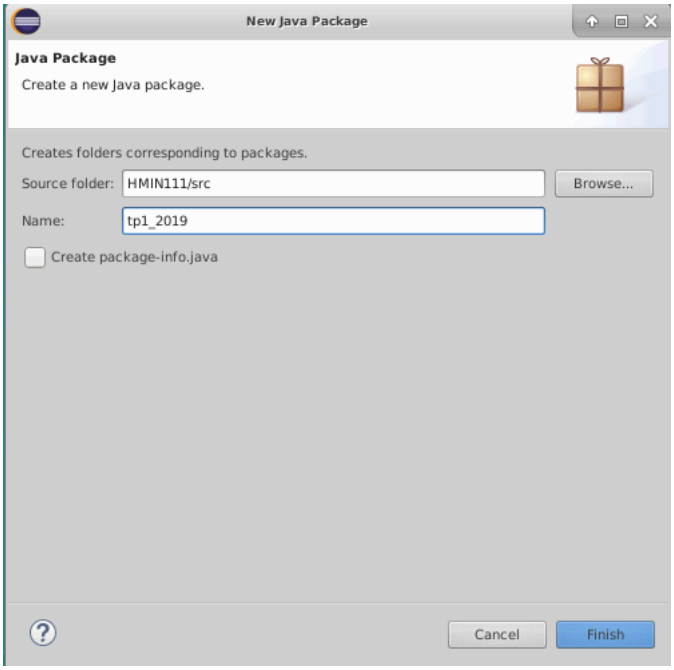
— Nommez le projet `HAI717I`. Eclipse crée dans le workspace un répertoire de ce nom (vous pouvez vérifier cela grâce à votre explorateur de fichiers). Vous utiliserez ce projet pour l'ensemble des TPs de ce module.
— Garder la case Use Default Location cochée
— Choisir Create Separate folders for sources and class files, afin de séparer vos fichiers sources (d'extension .java) des fichiers de bytecode (d'extension .class). Eclipse va créer dans votre projet un répertoire src (sources d'extension .java) et un répertoire bin (bytecodes qui vont être utilisés par Eclipse lors de l'interprétation)
— Dans la partie `JRE`, vérifiez que la version de la JRE est bien 1.8 ou au-dessus.
— Si vous faites Next, vous arrivez à des configurations fines qu'il n'est pas nécessaire de modifier pour l'instant → faites Finish

Dans le package explorer, on voit le nouveau projet `HAI717I` avec un répertoire src, et la référence à la librairie système JRE que vous n'avez pas besoin d'observer pour le moment.

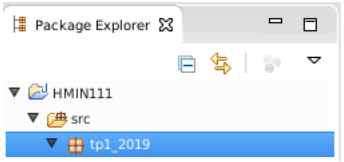


1.2.2 Création d'un package

Ne manquez pas cette étape, qui est très importante pour développer des projets bien structurés. En effet, un programme Java est organisé en plusieurs classes qui sont regroupées dans un même package. Il faut donc le définir pour pouvoir ensuite créer les classes du programme.
Dans le package explorer, faire un clic droit sur l'icône src de votre projet, puis → new → package. Une fenêtre s'ouvre permettant de donner un nom au package. Le nom commence usuellement par une minuscule, ici nous allons le nommer `tp1`.

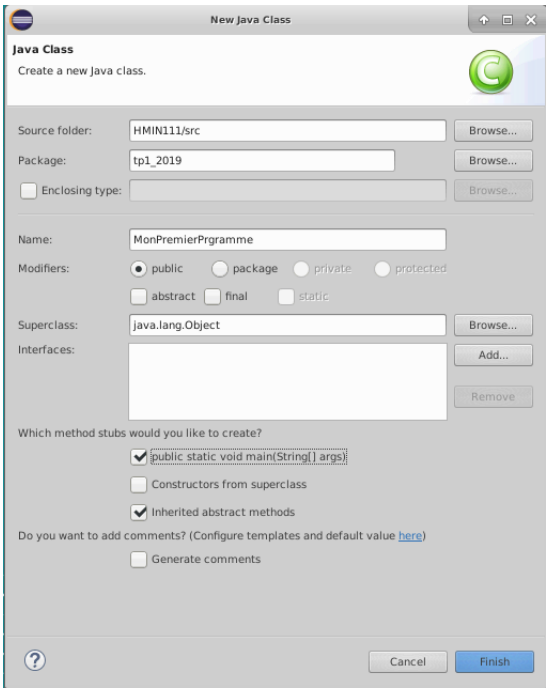


Dans le package explorer, on voit le nouveau package `tp1` à l'intérieur du projet `HAI717I`



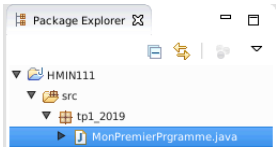
1.2.3 Création d'une classe

Un programme Java est généralement organisé en plusieurs classes de différents types qui peuvent avoir des relations les unes avec les autres. Vous allez ici créer votre première classe Java. Dans le package explorer, sur l'icône du paquetage `tp1`, faire un clic droit → new → class. Une nouvelle fenêtre s'ouvre permettant de donner les informations concernant la classe.

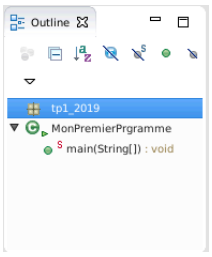


Vous devez :

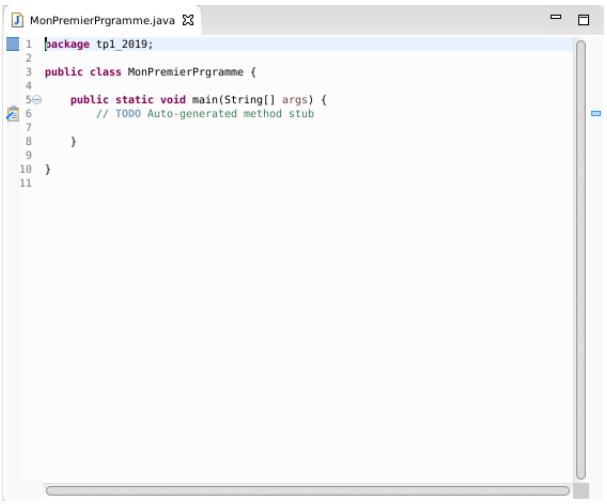
— Vérifier que le source folder est bien `HAI717I/src`, et le nom du package `tp1`
— Nommer la classe `MonPremierProgramme`. En Java, le nom d'une classe commence toujours par une majuscule.
— Cocher la case `public static void main`. C'est cette méthode qui va permettre d'exécuter le programme.
le package explorer, la classe apparaît à l'intérieur du package.



Dans l'outline (à droite), s'ouvre un explorateur sur la classe faisant apparaître ses propriétés (ici on ne voit que la méthode main).



Au centre, se trouve le code source de la classe qui a été généré automatiquement par Eclipse.



— La première instruction spécifie le nom du package auquel appartient la classe
— La seconde instruction définit la classe en elle même
— Enfin, la méthode main est définie. Vous remarquerez le commentaire TODO qui signale qu'il faut compléter la méthode main pour qu'elle puisse exécuter quelque chose (à ce stade, elle ne fait rien...)
— Modifiez la méthode main pour y ajouter :

```
System.out.println("Mon premier programme");
```

1.2.4 Exécution d'un programme

Pour exécuter le programme, choisir : menu run → run as → java application ou utiliser l'icône 
Cela exécute le programme et vous montre les résultats dans une fenêtre console (en bas). Il se peut qu'il n'y ait pas de fenêtre console ouverte, dans ce cas, allez dans le menu "Window", choisissez Show View puis console dans les menus successifs.

Normalement, vous ne devriez pas avoir besoin de le faire, mais si la situation se présente, pour arrêter une exécution en cours de route, appuyer sur le carré rouge vers la droite du bandeau de la fenêtre console.

Nota1. La console sert également à vous indiquer des messages d'erreur lors de l'exécution de votre code. Par exemple, si vous avez oublié le ; à la fin de l'instruction System.out.println("Mon premier programme") et que vous voulez malgré tout exécuter votre code. Eclipse, vous signalera par une fenêtre qu'il y a des erreurs et vous affichera ensuite le type d'erreur dans la console.

Nota2. Si vous perdez des fenêtres et vous trouvez perdu, allez dans le menu "Window" et choisissez "reset perspective", vous retrouverez le package explorer, la fenêtre de code centrale et la fenêtre d'outline à droite.

2 Premiers pas en Java

Vous utiliserez le fichier `MainTP1.java` qui contient des parties pré-définies (à récupérer sur moodle). Pour pouvoir l'utiliser, vous devez importer ce fichier dans Eclipse. Pour cela, vous devez faire un clic droit sur le package `tp1` et faire `import`. Une fenêtre s'ouvre, sélectionnez `FileSystem`, cliquez sur `next` et cherchez le répertoire où vous avez sauvegardé le fichier, sélectionnez `MainTP.java` en cochant la case et cliquez sur `Finish`. La classe devrait apparaître dans le `Package Explorer` à gauche. Vous pouvez maintenant l'ouvrir en double cliquant sur celle-ci. Le code pré-rempli est maintenant visible dans la partie centrale d'Eclipse.

2.1 Une première méthode à étudier et à tester

Dans la partie `Méthodes` du fichier `MainTP1.java`, étudiez le code de la méthode suivante :

— Méthode `celsius2Farenheit` qui convertit les degrés celsius en dégrés farenheit.
Sa signature théorique est `celsius2Farenheit(double tempC) : double`
ce qui signifie qu'elle prend en paramètre un réel qui représente la température en degrés Celsius et retourne un réel qui représente la température en degrés Farenheit.

Dans la partie `Main` du programme (à la fin du fichier), vous allez tester la méthode.

— Regardez l'exemple donné pour la méthode `celsius2Farenheit` avec la valeur 35,
— Utilisez cet exemple pour écrire le code permettant de tester la méthode `celsius2Farenheit` avec la valeur correspondante à la température d'ébullition de l'eau.

2.2 A vous de jouer !

1. Des moyennes...
   — Définir une méthode `moyenne` qui à partir de trois notes d'un étudiant, calcule la moyenne de celui-ci.
   Sa signature est : `moyenne(double n1, double n2, double n3) : double`
   — Dans la partie `Main` du programme, écrire le code permettant de tester la méthode `moyenne` avec les valeurs fournies dans le fichier source
   — Ecrire une méthode `moyenne pondérée` qui est une variante de la méthode `moyenne`. Sa signature est : `moyennePonderee(double n1, double n2, double n3, double c1, double c2, double c3) : double` , où n1, n2, n3 correspondent aux notes de l'étudiant et où c1, c2, c3 sont les coefficients appliqués sur ces notes
   — Dans la partie `Main` du programme, écrire le code permettant de tester la méthode `moyennePonderee` avec les valeurs fournies dans le fichier source
2. Un peu de conjugaison

— Ecrire une méthode `conjugaisonFutur` qui à partir de l'infinitif d'un verbe du premier groupe, retourne une chaîne de caractères contenant la conjugaison du verbe au futur de l'indicatif. Sa signature est la suivante : `conjugaisonFutur(String inf) : String`
— Dans la partie `Main` du programme, écrire le code permettant de tester la méthode `conjugaisonFutur` avec les valeurs fournies dans le fichier source

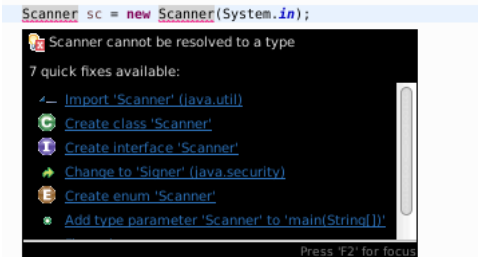
3 Approfondissement sur l'assistance d'Eclipse

3.1 Indentation du code

Vous remarquerez que le code de votre programme est positionné de telle sorte que les sous-instructions soient décalées par rapport à l'instruction principale (on dit alors qu'il est indenté). Si vous copiez du code non indenté ou si vous voulez rafraîchir l'indentation : sélectionner la partie concernée, clic droit → source → correct indentation (ou crtl +I).

3.2 Erreurs de compilation

Eclipse peut corriger quelques erreurs de compilation. Par exemple, dans la partie `Utilisation du scanner` du fichier `MainTP1.java`, décommentez la ligne qui contient l'instruction `Scanner sc = new Scanner(System.in);`. Vous observez qu'Eclipse souligne en rouge le mot clé Scanner. Si vous passez la souris sur ce mot clé, Eclipse vous propose plusieurs solutions pour résoudre ce problème :



La première solution proposée par Eclipse est d'importer la classe `java.util.Scanner` qui va permettre d'utiliser des objets de type Scanner. Cliquer sur `import java.util.Scanner`. Vous remarquez que dans l'entête du fichier (sous la spécification du package), une nouvelle ligne est apparue et que le trait rouge sous le mot clé Scanner a disparu. Eclipse vous a aidé à résoudre cette erreur de compilation !

3.3 Complétion sémantique

Sous l'instruction `Scanner sc = new Scanner(System.in);` tapez `sc.` (c'est-à-dire "sc" suivi de '.'). La liste des méthodes définies pour l'objet sc est proposée ; il suffit de choisir celle qui nous intéresse (par exemple `nextInt` pour saisir un nombre entier) en double-cliquant dessus.
Eclipse nous permettra par la suite d'ajouter automatiquement beaucoup d'autres éléments dans le code et de le modifier facilement, par exemple pour renommer des éléments.

3.4 Renommer un élément

Pour renommer un élément, sélectionnez cet élément (par exemple la classe `MainTP1.java`). Toutes ses occurrences dans les fichiers le contenant sont grisées. Dans le menu contextuel, sélectionnez Refactor puis Rename (ou clic droit Refactor → Rename). Donnez-lui le nom `Maintp1.java` et validez. Cela renomme alors l'élément dans tous les fichiers du même projet.

4 Le Scanner

Le scanner permet à l'utilisateur de saisir des informations au clavier ou de récupérer des informations dans un fichier, afin d'utiliser celles-ci dans une méthode ou pour afficher les informations sur la sortie standard

(l'écran). Pour pouvoir l'utiliser, il faut définir une instance de la classe `Scanner`, qui se trouve dans le paquetage `java.util` de l'API[1] Java fournie avec le `JDK`. Cela implique d'importer la classe avec l'instruction `import java.util.Scanner;`

4.1 Utilisation de la classe `Scanner` et de ses méthodes

Quand on crée une instance de `Scanner`, on passe en paramètre du constructeur (notion que nous verrons à un prochain cours) le texte à analyser : on peut lui passer un nom de fichier, ou bien un flux de texte comme `System.in`, qui permet l'analyse de l'entrée standard (i.e. le clavier).

```
Scanner fichier = new Scanner(new File("cheminVers/nomFichier.txt"));
Scanner clavier = new Scanner(System.in);
```

L'analyseur permet de lire les éléments d'un texte les uns après les autres. Pour cela, il est nécessaire de connaître le caractère séparateur des éléments dans le texte. Par défaut, il s'agit de l'espace. L'analyseur permet aussi de traduire les éléments lus vers le type que l'on lui fournit : on peut demander à lire le prochain entier, et on récupère alors un élément de type entier. Bien sûr, l'analyse échoue si l'on demande la lecture d'un entier et que l'on fournit une chaîne de caractères.
La classe `Scanner` dispose notamment des méthodes :

- `next()` qui retourne la prochaine chaîne de caractères lue dans le texte
- `nextInt()` qui retourne le prochain entier lu dans le texte
- `nextFloat()` qui retourne le prochain flottant lu dans le texte. Le caractère séparant la partie entière de la partie décimale est la virgule.

Voici quelques exemples d'utilisation :

```
System.out.println("entrez un entier ? ");
  int i = sc.nextInt();
  System.out.println("entier lu : "+i+"\n");

  System.out.println("entrez une chaine ? ");
  String s = sc.next();
  System.out.println("chaine lue : "+s+"\n");

  System.out.println("entrez un flottant ? ");
  float f = sc.nextFloat();
  System.out.println("flottant lu : "+f+"\n");

  System.out.println("entrez un double ? ");
  double d = sc.nextDouble();
  System.out.println("double lu : "+d+"\n");

  System.out.println("entrez un booléen ? ");
  boolean b = sc.nextBoolean();
  System.out.println("booléen lu : "+b+"\n");
```

4.2 Utilisation du scanner dans le `TP1`

Normalement vous avez déjà instancié un scanner et réalisé l'import de `java.util.Scanner;` dans la partie 3.2 de ce TP. Vous devez pouvoir retrouver le code correspondant dans la partie `Utilisation du scanner` du fichier `MainTP1.java`. Pour chacune des méthodes que nous avons définies, nous allons maintenant utiliser ce scanner pour donner la possibilité à l'utilisateur de rentrer les valeurs qu'il souhaite.

1. Application Programming Interface

— Un exemple vous est donné dans le TP pour utiliser le scanner avec la méthode `celsius2Farenheit`. Décommentez le code et exécutez-le.
— En vous inspirant de l'exemple, écrire le code pour les autres méthodes, à savoir :
  — `moyenne`
  — `moyennePonderee`,
  — `conjugaisonFutur`