

## TP0 : L'environnement de programmation « Eclipse »

Le but de cet exercice est de configurer (sous linux) l'environnement de programmation (Eclipse) qui sera utilisé en TP de Programmation.

Eclipse est un outil qui offre de nombreuses fonctionnalités utiles dans le *cycle de développement* d'un logiciel ; dans le cadre de nos TP, il nous servira principalement à effectuer la *saisie* de programmes en java et c++, à les *compiler*, les *mettre au point* et les *exécuter*.

**Espace de travail :** Avant de débiter, vous devrez créer (avec la commande `mkdir`) un répertoire de nom **prog** dans votre répertoire personnel ; il servira d'*espace de travail* (workspace) pour eclipse : c'est là que vous placerez les différents TP dans des *projets* gérés par eclipse.

### 1 Configuration de Eclipse

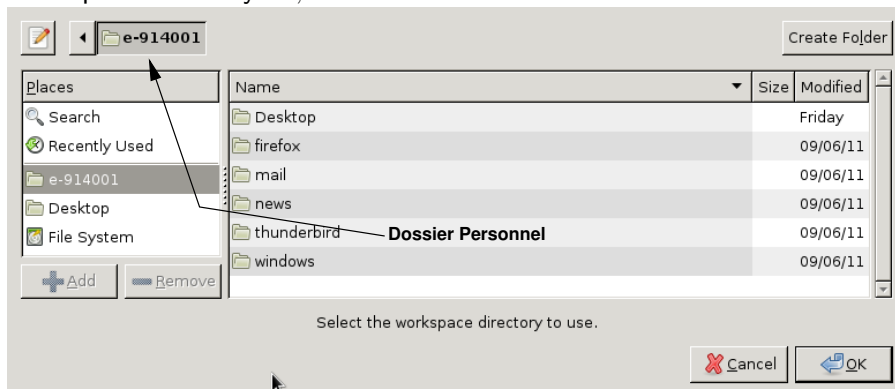
Cette configuration est à faire une fois pour toutes ; lorsqu'il faudra saisir des informations au clavier, pensez à respecter la casse (majuscule/minuscule).

Démarrez l'environnement de programmation eclipse : menu « Logiciels/Programmation/Eclipse » ; prenez la version « Mars Java ».

#### 1.1 Espace de travail par défaut

Vous allez *modifier l'espace de travail par défaut* utilisé par eclipse :

- Au démarrage, eclipse affiche une boîte de dialogue de titre « **Workspace Launcher** » ;
- cliquez « **Browse** » pour naviguer dans les répertoires ; ceci ouvre une fenêtre de titre « **Select Workspace Directory** » ;



- sélectionnez votre répertoire personnel dont le nom est votre numéro d'étudiant puis le répertoire **prog** créé au début ;
- validez ; quand vous revenez dans la boîte de dialogue « **Workspace Launcher** », vérifiez que le répertoire de l'espace de travail est bien celui que vous venez de créer ; il doit être de la forme : `/private/student/x/yy/zzzzzz/prog` où **zzzzzz** est votre numéro d'étudiant ; si ce n'est pas le cas, reprenez au début ;
- cochez la case « **Use this as the default...** » ;
- validez (« **OK** ») ; eclipse finit de démarrer ; ignorez d'éventuels messages de mise-à-jour.

## 1.2 Préférences générales

Ouvrez le panneau des préférences (menu « Window/Preferences ») ; dans la partie gauche de ce panneau se trouvent des rubriques ; celles qui ont une flèche à leur gauche comportent des sous-rubriques, qu'on peut afficher en cliquant sur la flèche ; dans la partie droite sont affichées des options propres à la rubrique sélectionnée.

### 1.2.1 Enregistrement automatique

Sélectionnez « General/Workspace » :

1. dé-cochez la case « Build automatically » (empêche eclipse de recompiler votre programme lors de chaque sauvegarde) ;
2. cochez les cases « Refresh on access » et « Save automatically before build » (oblige eclipse à enregistrer vos fichiers avant toute compilation) ;

### 1.2.2 Espace de travail initial

Sélectionnez « General/Startup and Shutdown/Workspaces » :

1. vérifiez que la case « Prompt for workspace on startup » n'est **pas** cochée ; si elle est cochée, dé-cochez-la : ceci empêche eclipse de vous demander quel espace de travail vous souhaitez utiliser à chaque démarrage.

### 1.2.3 Compilateur

1. sélectionnez « Java/Compiler » ; fixez le niveau de compatibilité du compilateur à 1.5 (« Compiler compliance level ») ;
2. sélectionnez « Java/Compiler/Errors » puis ouvrez la rubrique « Deprecated and Restricted API » puis, en face de la sous-rubrique « Forbidden reference », sélectionnez « Ignore » au lieu de « Error » ;
3. cliquez « Apply » ; eclipse vous demande si vous souhaitez régénérer tous vos projets répondez « Non » .

### 1.2.4 Assertions et bibliothèques

Vous allez ici *activer la vérification des assertions* et indiquer à eclipse d'utiliser les bibliothèques qui vous serviront lors des TP :

1. sélectionnez « Java/Installed JREs » : ceci vous donne la liste des versions de java disponibles ; dans la liste de droite, sélectionnez celle qui est cochée puis cliquez « Edit » ;
2. dans la boîte de dialogue « Edit JRE »
  - (a) dans la rubrique « Default VM arguments », tapez les deux options séparées par un espace :  
-enableassertions -enablesystemassertions
  - (b) cliquez « Add external JARs » puis naviguez dans le système de fichiers jusqu'au répertoire /share/esir1/prog/bibliotheques ; sélectionnez le fichier types.jar et validez : le fichier /share/esir1/prog/bibliotheques/types.jar doit apparaître dans la liste « JRE system libraries ».
  - (c) dans la liste « JRE system libraries », sélectionnez la bibliothèque types.jar :

- i. cliquez le bouton « **Source Attachment** » ;  
dans la boîte de dialogue « **Source Attachment Configuration** » qui s'ouvre, cliquez le bouton « **External File** », naviguez jusqu'au répertoire précédent (`/share/esir1/prog/bibliotheques`) puis sélectionnez le fichier `sourcesTypes.jar` ; validez jusqu'à ce que vous reveniez à la boîte « **Edit JRE** ».

Pour vérifier cette opération, cliquez sur la flèche à gauche du fichier `types.jar` : la sous-rubrique « **Source attachment** » doit indiquer le fichier `/share/esir1/prog/bibliotheques/sourcesTypes.jar`.

Quand tout est correct, cliquez « **Apply** » puis fermez la boîte « **Edit JRE** » en cliquant « **Finish** ».

### 1.2.5 Perspective de travail

Une *perspective de travail* est composée d'un certain nombre de fenêtres et d'onglets qui servent à présenter de façon structurée les informations d'un projet.

Ouvrez la rubrique « **General/Perspectives** » : dans la sous-rubrique « **Open the associated perspective when creating a new project** », cochez la case « **Always open** »

### 1.2.6 Divers

- Ouvrez la rubrique « **General/Editors/Text Editors** » et cochez la case « **Show line numbers** » ;
- Ouvrez « **General/Editors/Text Editors/Spelling** » et dé-cochez la case « **Enable spell checking** ».
- Validez

Enfin, cliquez « **OK** » dans le panneau « **Preferences** ».

La configuration est terminée ; vous pouvez fermer le panneau « **Welcome** » en cliquant sur la croix à droite de l'onglet : l'interface de travail d'eclipse s'affiche.

## 2 Création d'un projet

Vous allez maintenant créer un projet java, y intégrer un fichier source java, le compiler et l'exécuter ; ceci vous permettra de valider votre configuration ; à chaque étape, si vous n'obtenez pas les informations correctes, vérifiez la configuration de l'élément erroné et refaites les opérations nécessaires.

1. Créez un nouveau projet java (menu « **File/New/Java Project** ») ; donnez-lui un nom (par exemple : `tp0`) ; vérifiez, dans le champ « **Location** », que le projet que vous créez est bien situé dans le répertoire créé tout au début de la configuration (section 1.1) ; si tout va bien, cliquez « **Next** » ;
2. Dans la fenêtre suivante (« **Java settings** »), sélectionnez l'onglet « **Libraries** » : cliquez sur la flèche à gauche de l'élément « **JRE System Library** » et vérifiez la présence de la bibliothèque `types.jar` ; si tout est correct, cliquez « **Finish** ».

L'explorateur de paquets (onglet « **Package Explorer** ») s'affiche sur la gauche : il permet de naviguer dans les différents fichiers d'un projet java.

### 2.1 Incorporation d'un fichier source dans le projet

Il existe plusieurs façons d'incorporer un fichier source dans votre projet ; en voici une qui utilise la commande `cp` dans une fenêtre terminal : il suffit de copier le fichier `/share/esir1/prog/tp00/Exemple.java` dans le sous-répertoire `src` de votre projet (`~/prog/tp0/src`).


## 2.2 Explorer le fichier

À l'aide de l'explorateur de paquets, naviguez à l'intérieur du répertoire `src`, puis dans le paquetage par défaut (« `default package` ») : vous devriez voir le fichier `Exemple.java` ; cliquez sur la flèche à sa gauche : la liste qui s'affiche vous montre la structure interne du fichier `Exemple.java`, constitué d'une *classe* (`Exemple`), elle-même composée de plusieurs *fonctions* : `ecrireNombres`, `lireNombres`, `main`, `trierNombres` ;

Faites un double-clic sur la fonction `trierNombres` : ceci ouvre le fichier `Exemple.java` dans un éditeur qui occupe la fenêtre centrale de votre environnement de développement, et sélectionne la fonction `trierNombres` ; faites un double-clic sur un autre nom de fonction (`main`, par exemple) : l'éditeur vous affiche la fonction sélectionnée : c'est un moyen rapide de naviguer directement d'une fonction à une autre au sein de votre fichier.

À tout instant, vous pouvez agrandir un onglet en double-cliquant sur son titre et le ramener à sa taille initiale en faisant de même.

## 2.3 Compiler et exécuter

- Assurez-vous de sélectionner le fichier (`Exemple.java`) dans l'explorateur puis compilez-le (menu « `Project/Build Project` ») ; comme il n'y a aucune erreur, vous ne devriez rien voir de particulier ;
- Pour exécuter le programme *depuis eclipse*, menu « `Run/Run As/Java Application` » : ceci va démarrer le programme et ouvrir l'onglet « `Console` » ; c'est dans cet onglet que se fera l'interaction avec le programme : saisissez-y les données demandées par le programme (validez chaque nombre par l'appui de la touche « `Entrée` ») ; n'hésitez pas à agrandir la console (double-clic sur l'onglet).
- Le programme devrait afficher la suite de nombres telle que vous l'avez saisie, puis la même suite triée par ordre croissant.
- Fermez eclipse (menu « `File/Exit` »).
- Redémarrez eclipse : vous devriez retrouver le projet dans l'état précédent ; si ce n'est pas le cas, c'est probablement que vous n'utilisez pas le bon espace de travail ; changez-en ainsi :
  1. menu « `File/Switch Workspace` » ; choisissez le répertoire créé en section 1.1.
  2. après le redémarrage d'eclipse, faites ce qui est décrit en section 1.2.2.
- Vous pouvez à nouveau exécuter le programme ; cette fois-ci, il suffit de cliquer l'icône  : ceci va démarrer le dernier programme exécuté.

## 2.4 Erreurs

- Cherchez la fonction `initialiserTableau` et placez le curseur sur la ligne qui précède l'instruction `return lesNombres` ; et qui ne contient qu'une accolade fermante `}`.
- Supprimez cette accolade puis recompilez (menu `Project/Build Project`) : vous devriez voir apparaître une croix blanche sur fond rouge à gauche de la ligne erronée ;
- Placez la souris au-dessus de cette croix, sans cliquer : vous devriez voir apparaître un message expliquant l'erreur : corrigez puis recompilez.

## 2.5 Assertions (le retour)

Cette partie a pour but de vérifier le bon fonctionnement des assertions : dans le même projet, créez la classe `Racine` ; dans cette classe, programmez la fonction suivante :

```
/**
 * Calculer une valeur approchée de la racine carrée d'un nombre
 * @param r : réel dont on veut calculer la racine carrée
 * @param epsilon : précision du calcul
 * @pre r ≥ 0
 * @pre epsilon > 0
 * @return un réel a, valeur approchée à epsilon près de  $\sqrt{r}$ 
 * @post a ≥ 0 et  $|a^2 - r| < \epsilon$ 
 */
static double racineCarree(double r, double epsilon)
{
    // vérifier les pré-conditions
    assert r >= 0 : "*** PRÉ-CONDITION NON VÉRIFIÉE : r doit être >= 0";
    assert epsilon > 0 : "*** PRÉ-CONDITION NON VÉRIFIÉE : epsilon doit être > 0";
    // les pré-conditions sont vérifiées
    // programmer la suite de la fonction
}
```

Un algorithme très simple est décrit sur la page [https://fr.wikipedia.org/wiki/Méthode\\_de\\_Héron](https://fr.wikipedia.org/wiki/Méthode_de_Héron) : c'est un cas particulier de la « *Méthode de Newton* » appelé « *algorithme de Babylone* » ou « *méthode de Héron* ».

Programmez ensuite la fonction `main` qui fait appel à la fonction `racineCarree` ; faites un premier test en fournissant dans l'instruction d'appel des valeurs pour `r` et `epsilon` qui respectent la spécification de la fonction.

Faites ensuite un test en donnant à `r` ou `epsilon` une valeur hors spécification (valeur négative) : dans ce deuxième cas, votre programme devrait s'interrompre dans la fonction `RacineCarree` et vous afficher le message d'erreur adéquat. Si ce n'est pas le cas, c'est que l'activation des assertions n'a pas été faite correctement : retournez au paragraphe 1.2.4 et corrigez l'activation des assertions.

## 3 Test Unitaire

Consultez le document « *Test unitaire avec JUnit* », en particulier la section 4 (voir aussi <http://etudiant.istic.univ-rennes1.fr/current/esir1/prog/documentation>).

Copiez le fichier `TestUnitaireRacine.java` dans le répertoire de votre projet. C'est un programme de test unitaire que vous devez utiliser pour tester la fonction `racineCarree`.

Si vous rencontrez des erreurs lorsque vous compilez le programme de test, vérifiez d'abord si vous avez fait ce qui est indiqué dans le document ci-dessus ; vérifiez ensuite que votre fonction respecte bien la spécification imposée.

Votre fonction est considérée comme testée (mais pas obligatoirement correcte), si elle passe avec succès tous les tests.

## 4 Création d'un projet ex nihilo<sup>1</sup>

Pour créer un projet « à partir de rien », il faut :

---

1. [http://fr.wiktionary.org/wiki/ex\\_nihilo](http://fr.wiktionary.org/wiki/ex_nihilo)

- créer un *nouveau projet* (voir section 2)
- éventuellement créer *un ou plusieurs paquetages* (dépendra du problème) (« File/New/Package »)
- créer *une ou plusieurs classes* (« File/New/Class »)
- programmer puis compiler chaque classe comme dans la section 2.

## 5 Conclusion

Vous venez de configurer l'environnement eclipse : si vous avez effectué cette configuration avec soin, vous pourrez travailler dans de bonnes conditions lors des TP.

Vous avez aussi effectué les opérations élémentaires du cycle de vie d'un programme ; conservez ce document avec vous lors des TP et consultez-le quand ce sera nécessaire.