Préparation des TP de programmation

Pour vous aider à rafraîchir vos connaissances de java, vous pouvez consulter le poly N° 118 (« Licence Sciences et Technologies. Module API Approche impérative. Méthodes et outils algorithmiques. ») ; ce poly :

- est disponible à la bibliothèque universitaire ;
- peut être acheté à l'ISTIC (salle 186, premier étage) ;
- peut être téléchargé depuis le site `http://ndc.istic.univ-rennes1.fr` ; c'est un document au format PDF de 292 pages !

1 TP 0 : configuration d'eclipse

Il s'agit d'un TP de configuration et de prise en main de l'environnement de programmation qui sera utilisé par la suite en TP ; il est nécessaire d'avoir validé son sésame pour réaliser ce TP.

Ce TP est à réaliser en autonomie **par chaque** étudiant dans l'une des salles de TP linux (salles E003, E008, E103, E105, E212 dans le bâtiment 2B) **avant** de réaliser le TP 1.

Remarque : La configuration d'eclipse est à faire même si vous avez déjà utilisé eclipse auparavant.

2 Directives de rédaction d'une fonction

Toute fonction doit obligatoirement être précédée d'une *spécification* rédigée à l'intérieur d'un commentaire qui donnera les indications suivantes :

- rôle de chaque paramètre (étiquette @param) avec les *pré-conditions* qu'il doit remplir (étiquette @pre) ;
- effet précis de la fonction ;
- propriété (ou post-condition) du résultat (étiquette @post).

Exemple :

```
/**
   Calculer une valeur approchée de la racine carrée d'un nombre
   @param r : nombre dont on veut calculer la racine carrée
   @param epsilon : précision du calcul
   @pre r ≥ 0
   @pre epsilon > 0
   @return un nombre a, valeur approchée à epsilon près de √r
   @post a ≥ 0 et |a² − r| < epsilon
 */
```

Le principe d'utilisation de la spécification (et des pré- et post-conditions) d'une fonction **f** est le suivant :

1. le programmeur qui écrit *l'appel* de la fonction **f** doit vérifier que les valeurs qu'il donne aux paramètres de **f** respectent la spécification, en particulier les pré-conditions.
2. le programmeur qui écrit *le corps* de la fonction **f** fait l'hypothèse que les paramètres fournis respectent la spécification et vérifient les pré-conditions : il écrit donc sa fonction sans faire de contrôle des paramètres.
   Néanmoins, *afin de protéger sa fonction des appels qui ne respectent pas la spécification*, on insère en début de fonction une *assertion pour chaque pré-condition* à vérifier : si l'assertion est vérifiée (cas normal), le programme continue normalement, si l'assertion n'est pas vérifiée (erreur d'appel), le programme s'arrête et affiche le message d'erreur indiqué dans l'assertion.
   En outre, ce programmeur doit garantir que les post-conditions de sa fonction sont vérifiées.

La fonction racineCarree pourra donc s'écrire ainsi :

```
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
}
```

Et la fonction d'appel :

```
static void testerRacine()
{
  Scanner entree = new Scanner(System.in);
  double x;        // nombre dont on veut calculer la racine carrée
  double epsilon;  // précision du calcul
  // saisir les valeurs de x et epsilon et vérifier leur conformité
  do {
    x = entree.nextDouble();
    epsilon = entree.nextDouble();
  } while(x < 0 || epsilon <= 0);
  // valeurs conformes à la spécification de racineCarree
  // l'appel de racineCarre est possible
  double racine = racineCarre(x, epsilon);
  // suite normale
  // ...
}
```

Par ailleurs, vos programmes devront être correctement indentés : sélectionnez tout le code source puis faites *« Source/Correct indentation »*