TP3 : Eclipse Avancé

Objectif du TP

L'objectif de ce TP est de découvrir les fonctionnalités avancées d'Eclipse tels que le débogage, le Refactoring, ou l'écriture de tests unitaires.

Modèles de Code

1. Saisissez `systrace` dans la méthode `toString()` de la classe `Point` du TP1 puis utilisez la complétion sur ce mot. Que se passe-t-il ?
2. Grâce au menu `Window > Preferences > Java Editor > Templates`, créez un modèle `notif` qui fonctionne suivant le modèle `if` mais avec une négation de la condition.
3. Créez un modèle `sysdate` qui affiche la date d'utilisation.
4. Modifiez `systrace` pour placer le curseur immédiatement après la chaine de caractères.

Déboguage

5. L'objectif de cet exercice est d'utiliser les fonctionnalités de débogage d'Eclipse ***sans*** modifier le code source avec les méthodes `toString()` ou `System.out.println()`. Pour utiliser efficacement la perspective de débogage d'Eclipse, il est fortement conseillé de retenir les raccourcis clavier associés aux actions `Step Into`, `Step Over` et `Step Return`.
    a. Lisez le tutoriel suivant http://www.vogella.com/articles/EclipseDebugging/article.html.
    b. Créez un nouveau projet Puissance2 et importer le fichier puissance2.java. Exécuter et déboguer ce programme.
6. Créer un nouveau projet Eclipse avec le contenu du fichier `Debug.zip` et localiser la méthode `main()` ; elle contient des appels à des méthodes mettant en évidence des bogues dans d'autres classes du projet.
    a. La classe `Algorithms` implémente deux versions de la méthode `swap()` censée échanger les objets passés en arguments. Expliquer pourquoi la première version de cette méthode ne peut fonctionner. Vérifier avec le débogueur votre explication.
    b. La deuxième méthode propose un moyen de contourner le problème. Vérifier son bon fonctionnement en comparant les variables du cadre de pile de `swap()` de la classe `Algorithms` et les variables du cadre de pile de `testSwap()` de la classe Buggy.
    c. La méthode `binarySearch()` de la classe `Algorithms` est boguée (cf. l'affichage de la méthode `testBinarySearch()` de la classe Buggy). Utiliser (efficacement) le débogueur pour comprendre le problème (modifier éventuellement le code pour que les appels à cette méthode soient isolés sur une ligne) et le corriger.
    d. Corriger le ou les problèmes de la classe `GrowableArray` testée dans la méthode `testGrowableArray()` de la classe Buggy (utiliser le bon type de point d'arrêt. . .).
    e. Identifier la cause des bogues de la classe `List` mis en évidence par la méthode `testList()` de la classe Buggy puis les corriger (utiliser en particulier la vue `Variables` pour parcourir les listes en mémoire).

TESTS UNITAIRES AVEC JUNIT

7. Suivez le tutoriel suivant http://www.vogella.com/articles/JUnit/article.html jusqu'à la fin de la question 5.
8. Ajoutez un test à la classe `MyClass` pour que le calcul se fasse en moins de 1 ms.
9. Écrire un test unitaire `StringBufferTest` permettant de tester les méthodes `charAt`, `setCharAt` et `append(String str)` de la classe `StringBuffer`. Définir également les méthodes de test `testAppendWithNullString` et `testCharAtWithInvalidIndexes` testant les exceptions susceptibles d'être levées par la méthode `charAt`.
10. Tests sur les dates :
    a. Construisez un nouveau projet Eclipse contenant les fichiers `MyDate.java` & `MyDateTest.java`.
    b. On se propose de tester et mettre au point la méthode `isValidDate`. Compléter le test unitaire `MyDateTest` pour couvrir les cas suivants :

CE Valides	CE Invalides
1<= jour <=31 *ET* 1<= mois <= 12 *ET* année >= 0 *(a1)*	jour < 1 *(b1)*
	jour > 31 *(b2)*
	mois < 1 *(b3)*
	mois > 12 *(b4)*
	année > 2012 *(b5)*
	année < 1812 *(b6)*
mois € {2,4,6,9,11} => 1<= jour <=30 *(a2)*	mois € {2,4,6,9,11} ET (jour<1 OU jour>30) *(b7)*
(mois = 2) ET (année non bissextile) => (jour <= 28) *(a3)*	(mois = 2) ET (année non bissextile) ET (jour> 28) *(b8)*
(mois = 2) ET (année bissextile) => (jour <= 29) *(a4)*	(mois = 2) ET (année bissextile) ET (jour> 29) *(b9)*

REFACTORING

11. Suivez le tutoriel suivant http://www.vogella.com/articles/Eclipse/article.html#refactoring (questions 18 & 19).
12. Dans cet exercice il faudra utiliser au maximum les opérations de refactoring disponible dans Eclipse (à mettre dans le CR : comment vous avez fait pour obtenir ce qu'on vous demande). L'objectif est de saisir un mimum de ligne de code pour éviter les bugs !!!!
    a. Créez un nouveau projet Eclipse ;
    b. Dans le répertoire source, ajoutez un package `iut.bad` ;
    c. Créez une classe `Homme` ayant comme champs `nom, prenom, age` et des constructeurs ;
    d. Créez une classe `Femme` ayant les mêmes champs et des constructeurs ;
    e. Créez une classe `Humain` et modifiez `Homme/Femme` pour qu'elle soit leur classe parente ;
    f. Déplacez les attributs de `Homme/Femme` pour les placer dans le parent ;
    g. Dans la classe `Humain`, écrivez une méthode `details()` qui affiche le nom, prénom, et âge en utilisant un unique `System.out.println` ;
    h. Extrayez les paramètres affichés de la méthode précédente pour les placer dans une méthode `toString()` ;

i. Que se passe-t-il si vous voulez déplacer la méthode `toString()` vers les sous classes ?

j. Ajoutez les méthodes `manger()` et `boire()` à la classe `Humain` ;

k. Déplacez ces méthodes dans une interface `Consommation` ;

l. Ajoutez une méthode `ami(Humain)` qui permet d'indiquer une amitié entre 2 humains ;

m. Dans la classe `Femme`, créez une méthode `main()` qui instancie un `Homme` et une `Femme`, et déclare que l'un est ami de l'autre ;

n. Modifiez la signature d'`ami` pour qu'elle prenne un `int` en paramètre (la durée de l'amitié en jours) valant 100 par défaut.

GESTION DES TACHES

13. Ouvrez la vue `Tasks` et fermez toutes les fenêtres contenant du code source.
14. Suivez le tutoriel suivant http://www.vogella.com/articles/Eclipse/article.html#tasks (uniquement le paragraphe 34).
15. Créez un nouveau type de tâche `TOTEST` !
16. Il existe un autre gestionnaire de tâche intitulé `MyLyn`. Suivez ce tutoriel http://www.vogella.com/articles/Mylyn/article.html.

Pour aller plus loin

17. Vous allez avoir des cours d'UML. Sachez qu'il est également possible d'installer un plugin UML sous Eclipse : http://www.vogella.com/articles/UML/article.html mais cela risque de ne pas fonctionner. Je vous invite alors à installer Papyrus. Sachez également qu'un nouveau standard de modélisation émerge : SysML à la place de UML (néanmoins SysML est basé sur UML).

Sources pour ce TP

Cours de Fabrice Huet : http://www-sop.inria.fr/oasis/personnel/Fabrice.Huet/Enseignement/IUT/GL/
TPs de Pascal Graffion : http://deptinfo.cnam.fr/~graffion/UES/GLG101/tps/java/index.html
TPs de Patrick Labatut : http://www.normalesup.org/~labatut
Tutoriels Vogella : http://www.vogella.com/eclipseide.html