|  |
| --- |
| Document de suivi  de l’unité logicielle ULComparateur  ~  Spécification de l’unité de programme  ~  Fiche de test pour validation de l’unité  ~  Bilan de validation  ~  Annexes  (sources, données de test, résultats…)  ~  Code Interne : SPEC-FDT-BV\_Comparateur |

Résumé

|  |  |  |  |
| --- | --- | --- | --- |
| * Présentation des spécifications fonctionnelles du composant * Présentation des spécifications d’interface du composant * Plan de validation du composant * Fiche de test type * Compte rendu de validation et Visa de validation |  | **Niveau de diffusion** | |
|  | Contrôlée |  |
|  | Interne | X |
|  | Libre |  |

Validation

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nom | Entité | Date |
| **Auteur(s)** | Jérémy Guimberteau | **L3 Miage –** Groupe 5 | 26/05/2009 |
| **Vérificateur(s)** | Anaïs Dubourg | **L3 Miage –** Groupe 5 | 27/05/2009 |
|  |  |  |
| **Validation(s)** | dddddddddddd | **L3 Miage - Enseignant** | … |

Historique de mise à jour

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Rédacteur | Date | Objet | Paragraphe(s)  Concerné(s) |
| **S0F0** | Jérémy Guimberteau | 26/05/2009 | Création du document | De §1 à §5 |
| **S0F1** | Anaïs Dubourg | 27/05/2009 | Bilan – Visa de validation | De §1 à §5 |
|  |  |  |  |  |

Documents associés

|  |  |
| --- | --- |
| Titre | Référence du fichier |
|  |  |

**Table des matières**

[1. Spécifications de l’unité de programme 4](#_Toc231655780)

[1.1 Contexte 4](#_Toc231655781)

[1.2 Spécifications fonctionnelles de l’unité logicielle 4](#_Toc231655782)

[1.2.1 Exigences fonctionnelles 4](#_Toc231655783)

[1.2.2 Performances et exigences qualité 4](#_Toc231655784)

[1.3 Spécifications d’interface de l’unité logicielle 5](#_Toc231655785)

[1.3.1 Description détaillée des interfaces 5](#_Toc231655786)

[1.3.2 Schéma de la solution retenue 5](#_Toc231655787)

[1.4 Contraintes de réalisation 5](#_Toc231655788)

[1.4.1 Conditions d’activation 5](#_Toc231655789)

[1.4.2 Conditions d’exception et d’arrêt 5](#_Toc231655790)

[2. Plan de validation de l’unité logicielle 6](#_Toc231655791)

[2.1 Stratégie 6](#_Toc231655792)

[2.2 Exemple de scénario valide 6](#_Toc231655793)

[3. Fiche de test pour la validation de l’unité 7](#_Toc231655794)

[4. Bilan de validation 8](#_Toc231655795)

[4.1 Validation par le programmeur 8](#_Toc231655796)

[4.2 Validation par des testeurs indépendants 8](#_Toc231655797)

[4.3 Visa de validation 8](#_Toc231655798)

[5. Annexes : 9](#_Toc231655799)

[5.1 Les sources du programme 9](#_Toc231655800)

[5.1.1 Chemin de l’arborescence\UL.java 9](#_Toc231655801)

[5.1.2 etc.. 9](#_Toc231655802)

[5.2 Les fichiers d’entrée et de sortie 10](#_Toc231655803)

[5.2.1 [IN] - Chemin de l’arborescence\JDT.stub 10](#_Toc231655804)

[5.2.2 [OUT] - Chemin de l’arborescence \JDT.run 10](#_Toc231655805)

# Spécifications de l’unité de programme

## Contexte

L’unité logicielle « comparateur » s’inscrit dans le cycle de développement du service d’autotest du composant de vérification syntaxique.

## Spécifications fonctionnelles de l’unité logicielle

### Exigences fonctionnelles

L’unité logicielle « comparateur » a pour objectif de tester le service de vérification syntaxique. Les spécifications de l’unité logicielle peuvent se résumer de la manière suivante : l'unité logiciel a pour but de comparer 2 fichiers et de détecter leurs différences.

L’unité logicielle « comparateur » doit rendre possible l’**automatisation des tests** du composant final. Ainsi, il devra être possible de prendre en entrée un **fichier de jeu de données** et de produire en sortie un **fichier résultat** mémorisant les traitements effectués.

### Performances et exigences qualité

RAS

## Spécifications d’interface de l’unité logicielle

### Description détaillée des interfaces

L’unité logicielle **comparateur** a une unique méthode publique : fichier(*final String fichier1, String fichier2, String fichierEvaluation*).

Les fichiers passés en paramètre (fichier1, fichier2) sont comparés, le résultat de cette comparaison est décrit dans un fichier de sortie (fichierEvaluation), grâce à ce composant il est possible de comparer les fichier UL.jdt et JDT.ref ou les fichiers Oracle.run et UL.jdt afin de s'assurer que le service fonctionne correctement.

### Schéma de la solution retenue

Voici le schéma UML représentant la solution retenue pour l’unité logicielle UL1 :

|  |
| --- |
| Comparateur |
|  |
| public static void fichier(String fichier1, String fichier2, String fichierEvaluation) |

## Contraintes de réalisation

### Conditions d’activation

L’unité logicielle **compare** a besoin de 2 fichiers d'entrés à comparer et un fichier de sortie contenant de résultat de cette comparaison.

### Conditions d’exception et d’arrêt

L’unité logicielle ayant besoin de 2 fichiers en entrée, si un de ces derniers n’est pas trouvés, le traitement prend fin. De la même manière, si l’écriture des résultats ne peut se faire dans un fichier valide, l’exécution de la méthode fichierfichier(String fichier1, String fichier2, String fichierEvaluation) sera interrompue et l’unité logicielle ne pourra être considérée comme opérationnelle.

# Plan de validation de l’unité logicielle

## Stratégie

Pour pouvoir affirmer que l’unité logicielle comparateur est 0 défaut, il est nécessaire que les résultats des tests effectués sur cette unité logiciel soient positifs.

L’unité logicielle sera valide si il est écrit dans le fichier de résultat que les fichiers passés en paramètre sont identiques lorsque ces 2 fichiers contiennent les mêmes données.

Voici un exemple de scénario permettant de valider l’unité logicielle.

## Exemple de scénario valide

**Soit le JDT.ref suivant présenté en entrée :**

1 F1 0

2 F2 0 0

3 F3 0 0

4 F1 153

**Soit le JDT.run suivant présenté en entrée :**

1 F1 0

2 F2 0 0

3 F3 0 0

4 F1 153

**Le fichier en sortie devra être :**

Les 2 fichiers sont identiques

Maintenant que nous avons élaboré la stratégie de validation, nous pouvons définir une fiche de test type.

# Fiche de test pour la validation de l’unité

|  |  |  |  |
| --- | --- | --- | --- |
| **FICHE DE TEST** | | | |
| **Date de création de la FDT :** | 20/05/2009 | **Rédacteur :** | Jérémy Guimberteau |
| **Contact :** | jeremy.guimberteau@gmail.com | **Entité :** | **L3 Miage Paris Descartes** |
| **Version de la FDT :** | 1 | **Code interne** | **FDT\_ULComparateur** |
| **Intitulé :** | **Fiche de test unitaire de l’unité logicielle UL1** | | |

|  |  |  |
| --- | --- | --- |
| **DESCRIPTIF** | | |
| **Choisir la typologie du test** | **Test Unitaire** | Test d’intégration |
| **Description du test  - Données en entrée - Données en sortie** | | |
| Ce programme va prendre en entrée un **fichier « jeu de données »** et va produire, en sortie, un **fichier résultat** mémorisant les traitements effectués.  **Données en entrée**  Les données en entrées : exemple de JDT.ref et JDT.run que l’on pourra tester :  1 A0  2 17 B4  3 42EF  4 C2G3    1 A0  2 17 B4  3 42EF  4 C2G3  Les 2 fichiers sont identiques  **Données en sortie**  Les données en sortie : exemple de fichierque l’on pourrait obtenir : | | |

|  |  |  |
| --- | --- | --- |
| **PROCEDURES DE TESTS ET D’EVALUTAION** | | |
| **Mode d’évaluation des résultats :** | **Humain** | Automatique |
| **Procédures de tests :** | | |
| Pour tester la méthode public static void fichier(String fichier1, String fichier2, String fichierEvaluation)  Les données de tests sont présentes dans le répertoire :  Data tests\JDT.ref (ex : fichier JDT.ref)  Les résultats de sorties seront écrits dans le répertoire :  Data tests\JDT.run (ex : fichier JDT.run)  Pour l’évaluation des résultats et la validation de l’unité logicielle, … | | |

# Bilan de validation

## Validation par le programmeur

|  |  |  |  |
| --- | --- | --- | --- |
| **Date de passage du test :** | 19/05/2009 | **Rédacteur :** | Jérémy Guimberteau |
| **Testeur :** | Jérémy Guimberteau | jeremy.guimverteau@gmail.com | |
| **Compte Rendu de validation** | **2 types de test :**   * **Données identiques** * **Données différentes**   **La validation de l’unité logicielle Comparateur est acceptée.** | | |
|  | | | |

## Validation par des testeurs indépendants

|  |  |  |  |
| --- | --- | --- | --- |
| **Date de passage du test :** | 21/05/2009 | **Rédacteur :** | Jérémy Guimberteau |
| **Testeur(s) :**  (indépendant du programmeur) | Anaïs Dubourg | anais.dubourg@gmail.com | |
| **Compte Rendu de validation** | **L’unité logicielle est donc 0 Défaut.** | | |
|  | | | |

## Visa de validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Validation par le programmeur** | | | |
| **Visa Programmeur** | OK | **Date de validation** | 20/05/2008 |
| **Visa Testeur(s) indépendants** | OK | **Date de validation** | 21/05/2008 |
| **Conclusion** | **0-Défaut** | Bug Constaté | |
|  | | | |

# Annexes :

## Les sources du programme

### Code\comparateur\Comparateur.java

**public** **class** Comparateur {

**public** **static** **void** fichier(String fichier1,String fichier2, String fichierEvaluation){

**try**{

//ouverture du premier fichier

BufferedReader br1 = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(fichier1)));

//ouverture du second fichier

BufferedReader br2 = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(fichier2)));

//création du fichier résultat

PrintWriter result = **new** PrintWriter(**new** FileWriter(fichierEvaluation));

//création de variable de type string

String lectureLigneFichier1,lectureLigneFichier2;

//on initialise un booléen 'longueurFichierIdentique' à  true qui passera à  false si les 2 fichiers n'ont pas les mêmes données

**boolean** fichiersIdentique = **true**;

//on crée une variable qui va servir de compteur pour déterminer le nombre de ligne testée.

**int** nbLigneTeste = 0;

//tant que l'on peut lire les 2 fichiers

**while**(br1.ready() && br2.ready()){

nbLigneTeste++;

lectureLigneFichier1 = br1.readLine();

lectureLigneFichier2 = br2.readLine();

//si les lignes sont différentes on l'insert dans le fichier de résultat

**if**(!lectureLigneFichier1.equals(lectureLigneFichier2)){

result.println("Lignes non identiques : ["+ lectureLigneFichier1 + "] [" + lectureLigneFichier2 +"]");

fichiersIdentique = **false**;

}

}

//si il est possible de lire l'un des 2 fichiers, c'est que l'un possède plus de données que l'autre

**if**(br1.readLine() != **null**){

result.println("Le fichier 1 comporte plus de données que le fichier 2");

}

**else**{

**if**(br2.readLine() != **null**){

result.println("Le fichier 2 comporte plus de données que le fichier 1");

}

**else**{

**if**(fichiersIdentique) //si toute les lignes sont identiques et que les fichiers on le mÃªme nombre de donnÃ©es, les 2 fichiers sont identiques

result.println("Les 2 fichiers sont identiques");

}

}

//on précise le nombre de ligne testée

result.println("Nombre de ligne testée : " + nbLigneTeste);

//fermeture des 3 fichiers

br1.close();

br2.close();

result.close();

}

**catch**(Exception e)

{

System.*out*.println(e);

}

}

### Les fichiers d’entrée et de sortie

### [IN] - Data tests\JDT.ref

1 F1 0

2 F2 0 0

3 F3 0 0

4 F1 153

### [IN] - Data tests\JDT.run

1 F1 0

2 F2 0 0

3 F3 0 0

4 F1 153

### [OUT] - Data tests\eval.txt

Les 2 fichiers sont identiques.